
3 **Guidelines for the Selection,**
4 **Configuration, and Use of Transport**
5 **Layer Security (TLS) Implementations**

6
7 Kerry McKay
8 David Cooper
9

10
11
12
13
14
15 **C O M P U T E R S E C U R I T Y**
16
17

(DRAFT) NIST Special Publication 800-52
Revision 2

Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations

Kerry McKay
David Cooper
*Computer Security Division
Information Technology Laboratory*

November 2017



U.S. Department of Commerce
Wilbur L. Ross, Jr., Secretary

National Institute of Standards and Technology
Walter Copan, NIST Director and Under Secretary of Commerce for Standards and Technology

49

Authority

50 This publication has been developed by NIST in accordance with its statutory responsibilities under the
51 Federal Information Security Modernization Act (FISMA) of 2014, 44 U.S.C. § 3541 *et seq.*, Public Law
52 (P.L.) 113-283. NIST is responsible for developing information security standards and guidelines, including
53 minimum requirements for federal information systems, but such standards and guidelines shall not apply
54 to national security systems without the express approval of appropriate federal officials exercising policy
55 authority over such systems. This guideline is consistent with the requirements of the Office of Management
56 and Budget (OMB) Circular A-130.

57 Nothing in this publication should be taken to contradict the standards and guidelines made mandatory and
58 binding on federal agencies by the Secretary of Commerce under statutory authority. Nor should these
59 guidelines be interpreted as altering or superseding the existing authorities of the Secretary of Commerce,
60 Director of the OMB, or any other federal official. This publication may be used by nongovernmental
61 organizations on a voluntary basis and is not subject to copyright in the United States. Attribution would,
62 however, be appreciated by NIST.

63 National Institute of Standards and Technology Special Publication 800-52 Revision 2
64 Natl. Inst. Stand. Technol. Spec. Publ. 800-52 Rev. 2, 68 pages (November 2017)
65 CODEN: NSPUE2

66 Certain commercial entities, equipment, or materials may be identified in this document in order to describe an
67 experimental procedure or concept adequately. Such identification is not intended to imply recommendation or
68 endorsement by NIST, nor is it intended to imply that the entities, materials, or equipment are necessarily the best
69 available for the purpose.

70 There may be references in this publication to other publications currently under development by NIST in accordance
71 with its assigned statutory responsibilities. The information in this publication, including concepts and methodologies,
72 may be used by federal agencies even before the completion of such companion publications. Thus, until each
73 publication is completed, current requirements, guidelines, and procedures, where they exist, remain operative. For
74 planning and transition purposes, federal agencies may wish to closely follow the development of these new
75 publications by NIST.

76 Organizations are encouraged to review all draft publications during public comment periods and provide feedback to
77 NIST. Many NIST cybersecurity publications, other than the ones noted above, are available at
78 <https://csrc.nist.gov/publications>.

79

80 **Public comment period: *November 15, 2017 through February 1, 2018***

81 National Institute of Standards and Technology
82 Attn: Computer Security Division, Information Technology Laboratory
83 100 Bureau Drive (Mail Stop 8930) Gaithersburg, MD 20899-8930
84 Email: sp80052-comments@nist.gov

85 All comments are subject to release under the Freedom of Information Act (FOIA).

86

Reports on Computer Systems Technology

87 The Information Technology Laboratory (ITL) at the National Institute of Standards and
88 Technology (NIST) promotes the U.S. economy and public welfare by providing technical
89 leadership for the Nation's measurement and standards infrastructure. ITL develops tests, test
90 methods, reference data, proof of concept implementations, and technical analyses to advance the
91 development and productive use of information technology. ITL's responsibilities include the
92 development of management, administrative, technical, and physical standards and guidelines for
93 the cost-effective security and privacy of other than national security-related information in federal
94 information systems. The Special Publication 800-series reports on ITL's research, guidelines, and
95 outreach efforts in information system security, and its collaborative activities with industry,
96 government, and academic organizations.

97

Abstract

98 Transport Layer Security (TLS) provides mechanisms to protect data during electronic
99 dissemination across the Internet. This Special Publication provides guidance to the selection and
100 configuration of TLS protocol implementations while making effective use of Federal
101 Information Processing Standards (FIPS) and NIST-recommended cryptographic algorithms. It
102 requires that TLS 1.2 configured with FIPS-based cipher suites be supported by all government
103 TLS servers and clients and recommends that agencies develop migration plans to support TLS
104 1.3 by January 1, 2020. This Special Publication also provides guidance on certificates and TLS
105 extensions that impact security.

106

Keywords

107 information security; network security; SSL; TLS; Transport Layer Security

108

109

Acknowledgements

110 The authors, Kerry McKay and David Cooper of the National Institute of Standards and
111 Technology (NIST), would like to thank the many people who assisted with the development of
112 this document. In particular, we would like to acknowledge Tim Polk of NIST and Santosh
113 Chokhani of CygnaCom Solutions, who were co-authors on the first revision of this document.
114 We would also like to acknowledge Matthew J. Fanto and C. Michael Chernick of NIST and
115 Charles Edington III and Rob Rosenthal of Booz Allen and Hamilton who wrote the initial
116 published version of this document.

117

118

Note to Reviewers

119 Several developments have occurred since SP 800-52 Revision 1 regarding the use of RSA key
120 transport for key establishment in TLS. Research has shown that prominent TLS
121 implementations are incorrectly handling RSA key transport, leaving the key establishment
122 vulnerable to Bleichenbacher attacks. In addition, SP 800-131A currently disallows the use of
123 RSA key-transport using PKCS #1 v1.5 padding after December 31, 2017 (see
124 <https://csrc.nist.gov/News/2017/Transition-Plans-for-Key-Establishment-Schemes>). For these
125 reasons, all cipher suites that use RSA key transport to establish the premaster secret have been
126 removed from the recommended cipher suite list.

127 This may be problematic in architectures that currently rely on static RSA keys to support the
128 decryption of TLS sessions by network monitoring devices. For TLS version 1.2 and below, this
129 use case could be supported by switching to cipher suites that use static Diffie-Hellman (or static
130 Elliptic Curve Diffie-Hellman) keys. However, these cipher suites are not widely supported, and
131 this option is not available in TLS 1.3. Enterprise and datacenter monitoring could theoretically
132 be supported through a TLS 1.3 extension, re-architecting data flows with a man-in-the-middle,
133 or other measures outside the scope of TLS. A document proposing a TLS extension has
134 submitted to the Internet Engineering Task Force (IETF). The National Cybersecurity Center of
135 Excellence (NCCoE) plans to prototype this extension and other solutions that agencies and
136 organizations can use a template.

137 The Triple Data Encryption Algorithm (TDEA), also known as 3DES, is no longer approved for
138 use with TLS (see Department of Homeland Security Binding Operational Directive BOD-18-01,
139 <https://cyber.dhs.gov/assets/report/bod-18-01.pdf>). The 64-bit block size does not provide
140 adequate protection in applications such as TLS where large amounts of data are encrypted under
141 the same key.

142 This draft also requires agencies to develop migration plans to support TLS 1.3 by January 1,
143 2020.

144 **Executive Summary**

145 Office of Management and Budget (OMB) Circular A-130, *Managing Information as a Strategic*
146 *Resource*, requires managers of publicly accessible information repositories or dissemination
147 systems that contain sensitive but unclassified data to ensure that sensitive data is protected
148 commensurate with the risk and magnitude of the harm that would result from the loss, misuse,
149 or unauthorized access to or modification of such data. Given the nature of interconnected
150 networks and the use of the Internet to share information, the protection of this sensitive data can
151 become difficult if proper mechanisms are not employed to protect the data. Transport Layer
152 Security (TLS) provides such a mechanism to protect sensitive data during electronic
153 dissemination across the Internet.

154 TLS is a protocol created to provide authentication, confidentiality, and data integrity protection
155 between two communicating applications. TLS is based on a precursor protocol called the Secure
156 Sockets Layer Version 3.0 (SSL 3.0) and is considered to be an improvement to SSL 3.0. SSL
157 3.0 is specified in [33]. The Transport Layer Security version 1 (TLS 1.0) specification is an
158 Internet Request for Comments, RFC 2246 [24]. Each document specifies a similar protocol that
159 provides security services over the Internet. TLS 1.0 has been revised to version 1.1, as
160 documented in RFC 4346 [25], and TLS 1.1 has been further revised to version 1.2, as
161 documented in RFC 5246 [26]. In addition, some extensions have been defined to mitigate some
162 of the known security vulnerabilities in implementations using TLS versions 1.0, 1.1, and 1.2.
163 TLS 1.3, described in [56], is a significant update to previous versions that includes protections
164 against security concerns that arose in previous versions of TLS.

165 This Special Publication provides guidance to the selection and configuration of TLS protocol
166 implementations while making effective use of NIST-approved cryptographic schemes and
167 algorithms. In particular, it requires that TLS 1.2 be configured with cipher suites using NIST-
168 approved schemes and algorithms as the minimum appropriate secure transport protocol.¹ When
169 interoperability with non-government systems is required, TLS 1.1 and TLS 1.0 may be
170 supported. Agencies are required to develop migration plans to support to TLS 1.3 by 2020. This
171 Special Publication also identifies TLS extensions for which mandatory support must be
172 provided and other recommended extensions.

173 The use of the recommendations provided in this Special Publication would promote:

- 174 • More consistent use of authentication, confidentiality and integrity mechanisms for the
175 protection of information transported across the Internet;
- 176 • Consistent use of the recommended cipher suites that encompass NIST-approved
177 algorithms and open standards;
- 178 • Protection against known and anticipated attacks on the TLS protocol; and

¹ While SSL 3.0 is the most secure of the SSL protocol versions, it is not approved for use in the protection of Federal information because it relies in part on the use of cryptographic algorithms that are not NIST-approved. TLS 1.2 is approved for the protection of Federal information when properly configured. TLS versions 1.1 and 1.0 are approved only when it is required for interoperability with non-government systems and is configured according to these guidelines.

- 179 • Informed decisions by system administrators and managers in the integration of TLS
180 implementations.

181 While these guidelines are primarily designed for Federal users and system administrators to
182 adequately protect sensitive but unclassified U.S. Federal Government data against serious
183 threats on the Internet, they may also be used within closed network environments to segregate
184 data. (The client-server model and security services discussed also apply in these situations).
185 This Special Publication supersedes NIST Special Publication 800-52 Revision 1. This Special
186 Publication should be used in conjunction with existing policies and procedures.

187

188
 189
 190
 191
 192
 193
 194
 195
 196
 197
 198
 199
 200
 201
 202
 203
 204
 205
 206
 207
 208
 209
 210
 211
 212
 213
 214
 215
 216
 217
 218
 219
 220

Table of Contents

Executive Summary iv

1 Introduction 1

 1.1 Background..... 1

 1.2 History of TLS 1

 1.3 Scope..... 2

 1.3.1 Alternative Configurations 3

 1.4 Document Conventions..... 3

2 TLS Overview 4

 2.1 Handshake Protocol..... 4

 2.2 Shared Secret Negotiation 5

 2.3 Confidentiality 5

 2.4 Integrity 5

 2.5 Authentication 6

 2.6 Anti-Replay 6

 2.7 Key Management..... 7

3 Minimum Requirements for TLS Servers 8

 3.1 Protocol Version Support 8

 3.2 Server Keys and Certificates 9

 3.2.1 Server Certificate Profile..... 10

 3.2.2 Obtaining Revocation Status Information for the Client Certificate 12

 3.2.3 Server Public-Key Certificate Assurance 13

 3.3 Cryptographic Support 13

 3.3.1 Cipher Suites..... 14

 3.3.2 Implementation Considerations 19

 3.3.3 Validated Cryptography 20

 3.4 TLS Extension Support..... 21

 3.4.1 Mandatory TLS Extensions 21

 3.4.2 Conditional TLS Extensions 22

 3.4.3 Discouraged TLS Extensions 26

 3.5 Client Authentication 27

 3.5.1 Path Validation 27

221 3.5.2 Trust Anchor Store 28

222 3.5.3 Checking the Client Key Size 28

223 3.5.4 Server Hints List 28

224 3.6 Session Resumption 29

225 3.7 Compression Methods 29

226 3.8 Operational Considerations 29

227 **4 Minimum Requirements for TLS Clients 31**

228 4.1 Protocol Version Support 31

229 4.2 Client Keys and Certificates 31

230 4.2.1 Client Certificate Profile 31

231 4.2.2 Obtaining Revocation Status Information for the Server Certificate ... 33

232 4.2.3 Client Public-Key Certificate Assurance 34

233 4.3 Cryptographic Support 34

234 4.3.1 Cipher Suites 34

235 4.3.2 Validated Cryptography 35

236 4.4 TLS Extension Support 35

237 4.4.1 Mandatory TLS Extensions 35

238 4.4.2 Conditional TLS Extensions 36

239 4.4.3 Discouraged TLS Extension 38

240 4.5 Server Authentication 38

241 4.5.1 Path Validation 39

242 4.5.2 Trust Anchor Store 39

243 4.5.3 Checking the Server Key Size 39

244 4.5.4 User Interface 40

245 4.6 Session Resumption 40

246 4.7 Compression Methods 40

247 4.8 Operational Considerations 40

248

249 **List of Appendices**

250 **Appendix A— Acronyms 42**

251 **Appendix B— Interpreting Cipher Suite Names 44**

252 B.1 Interpreting Cipher Suites Names in TLS 1.0, 1.1, and 1.2 44

253 B.2 Interpreting Cipher Suites Names in TLS 1.3 45

254 **Appendix C— Pre-shared Keys 46**

255 **Appendix D— Future Capabilities..... 48**

256 D.1 U.S. Federal Public Trust PKI 48

257 D.2 DANE 48

258 **Appendix E— Determining the Need for TLS 1.0 and 1.1 50**

259 **Appendix F— References 51**

260 **Appendix G— Revision History 58**

261 G.1 Original 58

262 G.2 Revision 1 58

263 G.3 Revision 2 58

264

265 **1 Introduction**

266 Many networked applications rely on the Secure Sockets Layer (SSL) and Transport Layer
267 Security (TLS) protocols to protect data transmitted over insecure channels. The Internet's
268 client-server model and communication protocol design principles have been described in many
269 books, such as [54], [19], and [37]. TLS often works with a public-key infrastructure (PKI) that
270 generates public-key certificates in compliance with [20]. Books such as [1] and [40], as well as
271 technical journal articles (e.g., [53]) and NIST publications (e.g., SP 800-32 [44]), describe how
272 PKI can be used to protect information.

273 This document assumes that the reader of these guidelines is familiar with TLS protocols and
274 public-key infrastructure concepts, including, for example, X.509 certificates. The references
275 cited above and in Appendix F further explain the background concepts that are not fully
276 explained in these guidelines.

277 **1.1 Background**

278 The TLS protocol is used to secure communications in a wide variety of online transactions such
279 as financial transactions (e.g., banking, trading stocks, e-commerce), healthcare transactions
280 (e.g., viewing medical records or scheduling medical appointments), and social transactions (e.g.,
281 email or social networking). Any network service that handles sensitive or valuable data,
282 whether it is personally identifiable information (PII), financial data, or login information, needs
283 to adequately protect that data. TLS provides a protected channel for sending data between the
284 server and the client. The client is often, but not always, a web browser.

285 TLS is a layered protocol that runs on top of a reliable transport protocol – typically the
286 Transmission Control Protocol (TCP). Application protocols, such as the Hypertext Transfer
287 Protocol (HTTP) and the Internet Message Access Protocol (IMAP), can run above TLS. TLS is
288 application independent, and used to provide security to any two communicating applications
289 that transmit data over a network via an application protocol. It can be used to create a virtual
290 private network (VPN) that connects an external system to an internal network, allowing that
291 system to access a multitude of internal services and resources as if it were in the network.

292 Memorandum M-15-13² requires all publicly accessible Federal websites and web services only
293 provide service through a secure connection.³ The initiative to secure connections will enhance
294 privacy and prevent modification of the data from government sites in transit.

295 **1.2 History of TLS**

296 The SSL protocol was designed by the Netscape Corporation to meet security needs of client and
297 server applications. Version 1 of SSL was never released. SSL 2.0 was released in 1995, but had
298 well-known security vulnerabilities, which were addressed by the 1996 release of SSL 3.0.

² <https://obamawhitehouse.archives.gov/sites/default/files/omb/memoranda/2015/m-15-13.pdf>

³ See <https://https.cio.gov/> for more details on this initiative.

299 During this timeframe, the Microsoft Corporation released a protocol known as Private
300 Communications Technology (PCT), and later released a higher performance protocol known as
301 the Secure Transport Layer Protocol (STLP). PCT and STLP never commanded the market share
302 that SSL 2.0 and SSL 3.0 commanded. The Internet Engineering Task Force (IETF), a technical
303 working group responsible for developing Internet standards to ensure communications
304 compatibility across different implementations, attempted to resolve security engineering and
305 protocol incompatibility issues between the protocols as best it could. The IETF standards track
306 Transport Layer Security protocol Version 1.0 (TLS 1.0) emerged and was codified by the IETF
307 as RFC 2246 [24]. While TLS 1.0 is based on SSL 3.0, and the differences between them are not
308 dramatic, they are significant enough that TLS 1.0 and SSL 3.0 do not interoperate.

309 TLS 1.1, specified in RFC 4346 [25], was developed to address weaknesses discovered in TLS
310 1.0, primarily in the areas of initialization vector selection and padding error processing.
311 Initialization vectors were made explicit⁴ to prevent a certain class of attacks on the Cipher
312 Block Chaining (CBC) mode of operation used by TLS. The handling of padding errors was
313 altered to treat a padding error as a bad message authentication code, rather than a decryption
314 failure. In addition, the TLS 1.1 RFC acknowledges attacks on CBC mode that rely on the time
315 to compute the message authentication code (MAC). The TLS 1.1 specification states that to
316 defend against such attacks, an implementation must process records in the same manner
317 regardless of whether padding errors exist. Further implementation considerations for CBC
318 modes (which were not included in RFC 4346 [25]) are discussed in Section 3.3.2.

319 TLS 1.2, specified in RFC 5246 [26], made several cryptographic enhancements, particularly in
320 the area of hash functions, with the ability to use or specify the SHA-2 family algorithms for
321 hash, MAC, and Pseudorandom Function (PRF) computations. TLS 1.2 also adds authenticated
322 encryption with associated data (AEAD) cipher suites.

323 TLS 1.3, specified in [56], represents a significant change to TLS that aims to address threats
324 that have arisen over the years. Among the changes are a new handshake protocol, a new key
325 derivation process that uses the HMAC-based Extract-and-Expand Key Derivation Function
326 (HKDF) [43], and the removal of cipher suites that use static RSA or DH key exchanges, the
327 CBC mode of operation, or SHA-1. The list of extensions that can be used with TLS 1.3 has
328 been reduced considerably.

329 1.3 Scope

330 Security is not a single property possessed by a single protocol. Rather, security includes a
331 complex set of related properties that together provide the required information assurance
332 characteristics and information protection services. Security requirements are usually derived
333 from a risk assessment of the threats or attacks that an adversary is likely to mount against a
334 system. The adversary is likely to take advantage of implementation vulnerabilities found in
335 many system components, including computer operating systems, application software systems,
336 and the computer networks that interconnect them. Thus, in order to secure a system against a

⁴ The initialization vector (IV) must be sent; it cannot be derived from a state known by both parties, such as the previous message.

337 myriad of threats, security must be judiciously placed in the various systems and network layers.

338 These guidelines focus only on network security, and they focus directly on the small portion of
339 the network communications stack that is referred to as the transport layer. Several other NIST
340 publications address security requirements in the other parts of the system and network layers.
341 Adherence to these guidelines only protects the data in transit. Other applicable NIST standards
342 and guidelines should be used to ensure protection of systems and stored data.

343 These guidelines focus on the common use cases where clients and servers must interoperate
344 with a wide variety of implementations, and authentication is performed using public-key
345 certificates. To promote interoperability, implementations often support a wide array of
346 cryptographic options. However, there are much more constrained TLS implementations where
347 security is needed but broad interoperability is not required, and the cost of implementing unused
348 features may be prohibitive. For example, minimal servers are often implemented in embedded
349 controllers and network infrastructure devices such as routers, and then used with browsers to
350 remotely configure and manage the devices. There are also cases where both the client and server
351 for an application's TLS connection are under the control of the same entity, and therefore
352 allowing a variety of options for interoperability is not necessary. The use of an appropriate
353 subset of the capabilities specified in these guidelines may be acceptable in such cases.

354 The scope is further limited to TLS when used in conjunction with TCP/IP. For example,
355 Datagram TLS (DTLS) is outside the scope of these guidelines. NIST may issue separate
356 guidelines for DTLS at a later date.

357 1.3.1 Alternative Configurations

358 TLS may be used to secure the communications of a wide variety of applications in a diverse set
359 of operating environments. As such, there is not a single configuration that will work well for all
360 scenarios. These guidelines attempt to provide general-use recommendations. However, the
361 needs of an agency or application may differ from general needs. **Deviations from these**
362 **guidelines are acceptable, provided that agencies and system administrators assess and**
363 **accept the risks associated with alternative configurations in terms of both security and**
364 **interoperability.**

365 1.4 Document Conventions

366 Throughout this document, key words are used to identify requirements. The key words “**shall**,”
367 “**shall not**,” “**should**,” and “**should not**” are used. These words are a subset of the IETF Request
368 for Comments (RFC) 2119 key words, and have been chosen based on convention in other
369 normative documents [15]. In addition to the key words, the words “need,” “can,” and “may” are
370 used in this document, but are not intended to be normative. The key word “NIST-approved” is
371 used to indicate that a scheme or algorithm is described in a Federal Information Processing
372 Standard (FIPS) or is recommended by NIST.

373 The recommendations in this document are grouped by server recommendations and client
374 recommendations. Section 3 provides detailed guidance for the selection and configuration of
375 TLS servers. Section 4 provides detailed guidance for the selection, configuration, and use of
376 TLS clients.

377 2 TLS Overview

378 TLS exchanges records via the TLS record protocol. A TLS record contains several fields,
379 including version information, application protocol data, and the higher-level protocol used to
380 process the application data. TLS protects the application data by using a set of cryptographic
381 algorithms to ensure the confidentiality, integrity, and authenticity of exchanged application data.
382 TLS defines several protocols for connection management that sit on top of the record protocol,
383 where each protocol has its own record type. These protocols, discussed in Section 2.1, are used
384 to establish and change security parameters, and to communicate error and warning conditions to
385 the server and client. Sections 2.2 through 2.6 describe the security services provided by the TLS
386 protocol and how those security services are provisioned. Section 2.7 discusses key management.

387 2.1 Handshake Protocol

388 There are three subprotocols in the TLS protocol that are used to control the session connection:
389 the handshake, change cipher spec, and alert protocols. The TLS handshake protocol is used to
390 negotiate the session parameters. The alert protocol is used to notify the other party of an error
391 condition. The change cipher spec protocol is used in TLS 1.0, 1.1, and 1.2 to change the
392 cryptographic parameters of a session. In addition, the client and the server exchange application
393 data that is protected by the security services provisioned by the negotiated cipher suite. These
394 security services are negotiated and established with the handshake. The handshake protocol is
395 similar in TLS 1.0, 1.1, and 1.2, whereas the handshake of TLS 1.3 is different than in previous
396 TLS versions.

397 The handshake protocol consists of a series of message exchanges between the client and the
398 server. The handshake protocol initializes both the client and server to use cryptographic
399 capabilities by negotiating a cipher suite of algorithms and functions, including key
400 establishment, digital signature, confidentiality and integrity algorithms. Clients and servers can
401 be configured so that one or more of the following security services are negotiated during the
402 handshake: confidentiality, message integrity, authentication, and replay protection. A
403 confidentiality service provides assurance that data is kept secret, preventing eavesdropping. A
404 message integrity service provides confirmation that unauthorized data modification is detected,
405 thus preventing undetected deletion, addition, or modification of data. An authentication service
406 provides assurance of the sender or receiver's identity, thereby detecting forgery. Replay
407 protection ensures that an unauthorized user does not capture and successfully replay previous
408 data. In order to comply with these guidelines, both the client and the server must be configured
409 for data confidentiality and integrity services.

410 The handshake protocol is used to optionally exchange X.509 public-key certificates⁵ to
411 authenticate the server and the client to each other.

412 The handshake protocol is responsible for establishing the session parameters. The client and
413 server negotiate algorithms for authentication, confidentiality and integrity, as well as derive

⁵ The use of X.509 public-key certificates is fundamental to TLS. For a comprehensive explanation of X.509 public-key certificates see [1] or [40]. In these guidelines, the terms "certificate" and "public-key certificate" are used interchangeably.

414 symmetric keys and establish other session parameters, such as extensions. The negotiated set of
415 cryptographic algorithms is called the cipher suite.

416 Alerts are used to convey information about the session, such as errors or warnings. For example,
417 an alert can be used to signal a decryption error (`decrypt_error`) or that access has been denied
418 (`access_denied`). Some alerts are used for warnings, and others are considered fatal and lead to
419 immediate termination of the session. A `close_notify` alert message is used to signal normal
420 termination of a session. Like all other messages after the handshake protocol is completed, alert
421 messages are encrypted and optionally compressed.

422 Details of the handshake, change cipher spec (in TLS versions prior to 1.3) and alert protocols
423 are outside the scope of these guidelines; they are described in RFC 5246 [26] and [56].

424 **2.2 Shared Secret Negotiation**

425 The client and server establish keying material during the TLS handshake protocol. The
426 derivation of the premaster secret depends on the key exchange method that is agreed upon and
427 the version of TLS used. For example, when Diffie-Hellman is used as the key-exchange
428 algorithm in TLS 1.2 and earlier versions, the client and server send each other their parameters,
429 and the resulting key is used as the premaster secret. The premaster secret, along with random
430 values exchanged by the client and server in the hello messages, is used to compute the master
431 secret. In TLS 1.3, the master secret is derived by iteratively invoking an extract-then-expand
432 function with previously derived secrets. The master secret is used to derive session keys, which
433 are used by the negotiated security services to protect the data exchanged between the client and
434 the server, thus providing a secure channel for the client and the server to communicate.

435 The establishment of these secrets is secure against eavesdroppers. When the TLS protocol is
436 used in accordance with these guidelines, the application data, as well as the secrets, are not
437 vulnerable to attackers who place themselves in the middle of the connection. The attacker
438 cannot modify the handshake messages without being detected by the client and the server
439 because the Finished message, which is exchanged after security parameter establishment,
440 provides integrity protection to the entire exchange. In other words, an attacker cannot modify or
441 downgrade the security of the connection by placing itself in the middle of the negotiation.

442 **2.3 Confidentiality**

443 Confidentiality is provided for a communication session by the negotiated encryption algorithm
444 for the cipher suite and the encryption keys derived from the master secret and random values,
445 one for encryption by the client (the client write key), and another for encryption by the server
446 (the server write key). The sender of a message (client or server) encrypts the message using a
447 derived encryption key; the receiver uses the same (independently derived) key to decrypt the
448 message. Both the client and server know these keys, and decrypt the messages using the same
449 key that was used for encryption. The encryption keys are derived from the shared master secret.

450 **2.4 Integrity**

451 The keyed MAC algorithm, specified by the negotiated cipher suite, provides message integrity.
452 Two MAC keys are derived: 1) a MAC key to be used when the client is the message sender and

453 the server is the message receiver (the client write MAC key), and 2) a second MAC key to be
454 used when the server is the message sender and the client is the message receiver (the server
455 write MAC key). The sender of a message (client or server) calculates the MAC for the message
456 using the appropriate MAC key, and encrypts both the message and the MAC using the
457 appropriate encryption key. The sender then transmits the encrypted message and MAC to the
458 receiver. The receiver decrypts the received message and MAC, and calculates its own version of
459 the MAC using the MAC algorithm and sender's MAC key. The receiver verifies that the MAC
460 that it calculates matches the MAC sent by the sender.

461 Two types of constructions are used for MAC algorithms in TLS. TLS versions 1.0, 1.1 and 1.2
462 support the use of the Keyed-Hash Message Authentication Code (HMAC) using the hash
463 algorithm specified by the negotiated cipher suite. With HMAC, MACs for server-to-client
464 messages are keyed by the server write MAC key, while MACs for client-to-server messages
465 are keyed by the client write MAC key. These MAC keys are derived from the shared master
466 secret.

467 TLS 1.2 added AEAD cipher modes of operation, such as Counter with CBC-MAC (CCM) [47]
468 and Galois Counter Mode (GCM) [55, 59], as an alternative way of providing integrity and
469 confidentiality. In AEAD modes, the sender uses its write key for both encryption and integrity
470 protection. The client and server write MAC keys are not used. The recipient decrypts the
471 message and verifies the integrity information using the sender's write key. In TLS 1.3, only
472 AEAD symmetric algorithms are used for confidentiality and integrity.

473 **2.5 Authentication**

474 Server authentication is performed by the client using the server's public-key certificate, which
475 the server presents during the handshake. The exact nature of the cryptographic operation for
476 server authentication is dependent on the negotiated security parameters and extensions. In most
477 cases, authentication is performed explicitly by verifying digital signatures using public keys that
478 are present in certificates, and implicitly by the use of the server public key by the client during
479 the establishment of the master secret. A successful Finished message implies that both parties
480 calculated the same master secret and thus, the server must have known the private key
481 corresponding to the public key used for key establishment.

482 Client authentication is optional, and only occurs at the server's request. Client authentication is
483 based on the client's public-key certificate. The exact nature of the cryptographic operation for
484 client authentication depends on the negotiated cipher suite's key-exchange algorithm and the
485 negotiated extensions. For example, when the client's public-key certificate contains an RSA
486 public key, the client signs a portion of the handshake message using the private key
487 corresponding to that public key, and the server verifies the signature using the public key to
488 authenticate the client.

489 **2.6 Anti-Replay**

490 TLS provides inherent protection against replay attacks, except when 0-RTT data (optionally

491 sent in the first flight of handshake messages) is sent in TLS 1.3.⁶ The integrity-protected
492 envelope of the message contains a monotonically increasing sequence number. Once the
493 message integrity is verified, the sequence number of the current message is compared with the
494 sequence number of the previous message. The sequence number of the current message must be
495 greater than the sequence number of the previous message in order to further process the
496 message.

497 **2.7 Key Management**

498 The security of the server's private key is critical to the security of TLS. If the server's private
499 key is weak or can be obtained by a third party, the third party can masquerade as the server to
500 all clients. Similarly, if a third party can obtain a public-key certificate for a public key
501 corresponding to its own private key in the name of a legitimate server from a certification
502 authority (CA) trusted by the clients, the third party can masquerade as the server to the clients.
503 Requirements and recommendations to mitigate these concerns are addressed later in these
504 guidelines.

505 Similar threats exist for clients. If a client's private key is weak or can be obtained by a third
506 party, the third party can masquerade as the client to a server. Similarly, if a third party can
507 obtain a public-key certificate for a public key corresponding to his own private key in the name
508 of a client from a CA trusted by the server, the third party can masquerade as that client to the
509 server. Requirements and recommendations to mitigate these concerns are addressed later in
510 these guidelines.

511 Since the random numbers generated by the client and server contribute to the randomness of the
512 session keys, the client and server must be capable of generating random numbers with at least
513 112 bits of security⁷ each. The various TLS session keys derived from these random values and
514 other data are valid for the duration of the session. Because the session keys are only used to
515 protect messages exchanged during an active TLS session, and are not used to protect any data at
516 rest, there is no requirement for recovering TLS session keys. However, all versions of TLS
517 provide mechanisms to store a key related to a session, which allows sessions to be resumed in
518 the future. Keys for a resumed session are derived during an abbreviated handshake that uses the
519 stored key as a form of authentication.

520

⁶ While TLS 1.3 does not inherently provide replay protection with 0-RTT data, the TLS 1.3 specification does recommend mechanisms to protect against replay attacks (see Section 8 of [56]).

⁷ Bits of security provided by NIST-approved algorithms are described in SP 800-57 part 1 [6], Section 5.6.

521 **3 Minimum Requirements for TLS Servers**

522 This section provides a minimum set of requirements that a server must implement in order to
523 meet these guidelines. Requirements are organized in the following sections: TLS protocol
524 version support; server keys and certificates; cryptographic support; TLS extension support;
525 client authentication; session resumption; compression methods; and operational considerations.

526 Specific requirements are stated as either implementation requirements or configuration
527 requirements. Implementation requirements indicate that Federal agencies **shall not** procure TLS
528 server implementations unless they include the required functionality, or can be augmented with
529 additional commercial products to meet requirements. Configuration requirements indicate that
530 TLS server administrators are required to verify that particular features are enabled or disabled,
531 or in some cases, configured appropriately, if present.

532 **3.1 Protocol Version Support**

533 Servers that support government-only applications⁸ **shall** be configured to use TLS 1.2, and
534 **should** be configured to use TLS 1.3. These servers **should not** be configured to use TLS 1.1,
535 and **shall not** use TLS 1.0, SSL 3.0, or SSL 2.0. TLS versions 1.2 and 1.3 are represented by
536 major and minor number tuples (3, 3) and (3, 4), respectively, and may appear in that format
537 during configuration.⁹ Agencies **shall** develop migration plans to support TLS 1.3 by January 1,
538 2020. After this date, use of TLS 1.3 **shall** be supported in the government's servers.

539 Servers that support citizen or business-facing applications (i.e., the client may not be part of a
540 government IT system)¹⁰ **shall** be configured to negotiate TLS 1.2, **should** be configured to
541 negotiate TLS 1.3, and may be configured to negotiate TLS versions 1.1 and 1.0 in order to
542 enable interaction with citizens and businesses. See Appendix E for discussion on determining
543 whether to support TLS 1.0 and TLS 1.1. These servers **shall not** allow the use of SSL 2.0 or
544 SSL 3.0.

545 Some server implementations are known to implement version negotiation incorrectly. For
546 example, there are TLS 1.0 servers that terminate the connection when the client offers a version
547 newer than TLS 1.0. Servers that incorrectly implement TLS version negotiation **shall not** be
548 used.

⁸ A government-only application is an application where the intended users are exclusively government employees or contractors working on behalf of the government. This includes applications that are accessed on a government employee's bring-your-own-device (BYOD) system. This is in contrast to applications that are publicly accessible.

⁹ Historically TLS 1.0 was assigned major and minor tuple (3,1) to align it as SSL 3.1. TLS 1.1 is represented by the major and minor tuple (3,2).

¹⁰ For the purposes of this document, clients that reside on "bring your own device" (BYOD) systems, or privately-owned systems used to perform telework, are considered to be part of the government IT system, as they access services that are not available to the public.

549 3.2 Server Keys and Certificates

550 The TLS server **shall** be configured with one or more public-key certificates and the associated
551 private keys. TLS server implementations **should** support the use of multiple server certificates
552 with their associated private keys to support algorithm and key size agility.

553 Several options for TLS server certificates meet the requirement for NIST-approved
554 cryptography: an RSA signature certificate; an Elliptic Curve Digital Signature Algorithm
555 (ECDSA) signature certificate; a Digital Signature Algorithm (DSA)¹¹ signature certificate; a
556 Diffie-Hellman (DH) certificate; and an Elliptic Curve Diffie-Hellman (ECDH) certificate.

557 At a minimum, TLS servers conforming to this specification **shall** be configured with an RSA
558 signature certificate or an ECDSA signature certificate. If the server is configured with an
559 ECDSA signature certificate, a Suite B named curve **should** be used for the public key in the
560 certificate.¹²

561 TLS servers that are accessible to systems residing on a different network (e.g., connected to the
562 Internet) **shall** be configured with certificates issued by a CA, rather than self-signed certificates.
563 Furthermore, TLS server certificates **shall** be issued by a CA that publishes revocation
564 information in Online Certificate Status Protocol (OCSP) [61] responses. The CA may
565 additionally publish revocation information in a certificate revocation list (CRL) [20]. The
566 source(s) for the revocation information **shall** be included in the CA-issued certificate in the
567 appropriate extension to promote interoperability.

568 A TLS server that has been issued certificates by multiple CAs can select the appropriate
569 certificate based on the client specified “Trusted CA Keys” TLS extension, as described in
570 Section 3.4.2.7. A TLS server that has been issued certificates for multiple server names can
571 select the appropriate certificate based on the client specified “Server Name” TLS extension, as
572 described in Section 3.4.1.2. A TLS server certificate may also contain multiple names in the
573 Subject Alternative Name extension in order to allow the use of multiple server names of the
574 same name form (e.g., DNS name) or multiple server names of multiple name forms (e.g., DNS
575 names, IP address, etc.).

576 Application processes for obtaining certificates differ and require different levels of proof when
577 associating certificates to domains. An applicant can obtain a domain-validated (DV) certificate
578 by proving control over a DNS domain. An Organization Validation (OV) certificate requires
579 further vetting, such as verifying the entity’s details. An Extended Validation (EV) certificate has
580 the most thorough identity vetting process. This recommendation does not provide guidance on
581 which verification level to use.

582 Section 3.2.1 specifies a detailed profile for server certificates. Basic guidelines for RSA,
583 ECDSA, DSA, DH, and ECDH certificates are provided. Section 3.2.2 specifies requirements for

¹¹ In the names for the TLS cipher suites, DSA is referred to as DSS (Digital Signature Standard), for historical reasons.

¹² The Suite B curves are known as P-256 and P-384. These curves are defined in FIPS 186-4 [66], and their inclusion in Suite B is documented in [60].

584 revocation checking. Section 3.5.4 specifies requirements for the “hints list.”

585 3.2.1 Server Certificate Profile

586 The server certificate profile, described in this section, provides requirements and
 587 recommendations for the format of the server certificate. To comply with these guidelines, the
 588 TLS server certificate **shall** be an X.509 version 3 certificate; both the public key contained in
 589 the certificate and the signature **shall** provide at least 112 bits of security. Prior to TLS 1.2, the
 590 server Certificate message required that the signing algorithm for the certificate be the same as
 591 the algorithm for the certificate key (see Section 7.4.2 of [25]). If the server supports TLS
 592 versions prior to TLS 1.2, the certificate **should** be signed with an algorithm consistent with the
 593 public key:^{13,14}

- 594 • Certificates containing RSA, ECDSA, or DSA public keys **should** be signed with those
- 595 same signature algorithms, respectively;
- 596 • Certificates containing Diffie-Hellman public keys **should** be signed with DSA; and
- 597 • Certificates containing ECDH public keys **should** be signed with ECDSA.

598 The extended key usage extension limits how the keys in a certificate are used. There is a key
 599 purpose specifically for server authentication, and the server **should** be configured to allow its
 600 use. The use of the extended key usage extension will facilitate successful server authentication,
 601 as some clients may require the presence of an extended key usage extension. The use of the
 602 server DNS name in the Subject Alternative Name field ensures that any name constraints on the
 603 certification path will be properly enforced.

604 The server certificate profile is listed in Table 3-1. In the absence of agency-specific certificate
 605 profile requirements, this certificate profile **should** be used for the server certificate.

606

Table 3-1: TLS Server Certificate Profile

Field	Critical	Value	Description
Version	N/A	2	Version 3
Serial Number	N/A	Unique positive integer	Must be unique

¹³ This recommendation is an artifact of requirements in TLS 1.0 and 1.1.

¹⁴ Algorithm-dependent guidelines exist for the generation of public and private key pairs. For guidance on the generation of DH and ECDH key pairs, see SP 800-56A [8]. For guidance regarding the generation of RSA, DSA and ECDSA key pairs, see [66].

Field	Critical	Value	Description
Issuer Signature Algorithm	N/A	<i>Values by CA key type:</i>	
		sha256WithRSAEncryption {1 2 840 113549 1 1 11}, or stronger	CA with RSA key
		ecdsa-with-SHA256 {1 2 840 10045 4 3 2}, or stronger	CA with elliptic curve key
		id-dsa-with-sha256 {2 16 840 1 101 3 4 3 2}, or stronger	CA with DSA key
Issuer Distinguished Name (DN)	N/A	Unique X.500 issuing CA DN	A single value shall be encoded in each Relative Distinguished Name (RDN). All attributes that are of DirectoryString type shall be encoded as a PrintableString.
Validity Period	N/A	3 years or less	Dates through 2049 expressed in UTCTime
Subject Distinguished Name	N/A	Unique X.500 subject DN per agency requirements	A single value shall be encoded in each RDN. All attributes that are of DirectoryString type shall be encoded as a PrintableString. CN={host IP address host DNS name}
Field	Critical	Value	Description
Subject Public Key Information	N/A	<i>Values by certificate type:</i>	
		rsaEncryption {1 2 840 113549 1 1 1}	RSA signature certificate 2048-bit RSA key modulus, or other approved lengths as defined in [66] and [6] Parameters: NULL
		ecPublicKey {1 2 840 10045 2 1}	ECDSA signature certificate or ECDH certificate Parameters: namedCurve OID for named curve specified in [66]. The curve should be P-256 or P-384 SubjectPublic Key: Uncompressed EC Point.
		id-dsa {1 2 840 10040 4 1}	DSA signature certificate Parameters: p, q, g (2048-bit large prime, i.e., p)
		dhpublicnumber {1 2 840 10046 2 1}	DH certificate Parameters: p, g, q (2048-bit large prime, i.e., p)
Issuer's Signature	N/A	Same value as in Issuer Signature Algorithm	
Extensions			

Field	Critical	Value	Description
Authority Key Identifier	No	Octet String	Same as subject key identifier in issuing CA certificate Prohibited: Issuer DN, Serial Number tuple
Subject Key Identifier	No	Octet String	Same as in PKCS-10 request or calculated by the issuing CA
Key Usage	Yes	<i>Values by certificate type:</i>	
		digitalSignature	RSA signature certificate, ECDSA signature certificate, or DSA signature certificate
		keyAgreement	ECDH certificate, DH certificate
Extended Key Usage	No	id-kp-serverAuth {1 3 6 1 5 5 7 3 1}	Required
		id-kp-clientAuth {1 3 6 1 5 5 7 3 2}	Optional
			Prohibited: anyExtendedKeyUsage; all others unless consistent with key usage extension
Certificate Policies	No		Optional
Subject Alternative Name	No	DNS host name, or IP address if there is no DNS name assigned	Multiple SANs are permitted, e.g., for load balanced environments.
Authority Information Access	No	id-ad-caIssuers	Required. Access method entry contains HTTP URL for certificates issued to issuing CA
		id-ad-ocsp	Required. Access method entry contains HTTP URL for the issuing CA OCSP responder
CRL Distribution Points	No	See comments	Optional. HTTP value in distributionPoint field pointing to a full and complete CRL. Prohibited: reasons and cRLIssuer fields, and nameRelativetoCRLIssuer CHOICE
Signed Certificate Timestamps List	No	See comments	Optional. This extension contains a sequence of Signed Certificate Timestamps, which provide evidence that the certificate has been submitted to Certificate Transparency logs.

607

608 **3.2.2 Obtaining Revocation Status Information for the Client Certificate**

609 The server **shall** perform revocation checking of the client certificate when client authentication
 610 is used. Revocation information **shall** be obtained by the server from one or more of the
 611 following locations:

- 612 1. Certificate Revocation List (CRL) or OCSP [61] response in the server’s local store;
- 613 2. OCSP response from a locally configured OCSP responder;
- 614 3. OCSP response from the OCSP responder location identified in the OCSP field in the
 615 Authority Information Access extension in the client certificate; or
- 616 4. CRL from the CRL Distribution Points extension in the client certificate.

617 When the local store does not have the current or a cogent¹⁵ CRL or OCSP response, and the
618 OCSP responder and the CRL distribution point are unavailable or inaccessible at the time of
619 TLS session establishment, the server will either deny the connection or accept a potentially
620 revoked or compromised certificate. The decision to accept or reject a certificate in this situation
621 **should** be made according to agency policy.

622 **3.2.3 Server Public-Key Certificate Assurance**

623 The policies, procedures, and security controls under which a public-key certificate is issued by a
624 CA are documented in a certificate policy. The use of a certificate policy that is designed with
625 the secure operation of PKI in mind and adherence to the stipulated certificate policy mitigates
626 the threat that the issuing CA can be compromised or that the registration system, persons or
627 process can be compromised to obtain an unauthorized certificate in the name of a legitimate
628 entity, and thus compromise the clients. With this in mind, the CA Browser Forum, a private-
629 sector organization, has carried out some efforts in this area by writing the Extended Validation
630 guideline [17]. Under another effort, the CA Browser Forum published requirements for issuing
631 certificates from publicly trusted CAs in order for those CAs and their trust anchor to remain in
632 browser trust stores [16].

633 Several concepts are under development that further mitigate the risks associated with the
634 compromise of a CA or X.509 certificate registration system, process or personnel. These
635 include the Certificate Transparency project (see Section 3.4.2.11) and other emerging concepts,
636 which are discussed in Appendix D.

637 The policy under which a certificate has been issued may optionally be represented in the
638 certificate using the certificatePolicies extension, specified in [20] and updated in [73]. When
639 used, one or more certificate policy object identifiers (OID) are asserted in this extension, with
640 each OID representing a specific certificate policy. Many TLS clients (e.g., browsers), however,
641 do not offer the ability to accept or reject certificates based on the policies under which they
642 were issued. Therefore, it is generally necessary for TLS server certificates to be issued by CAs
643 that only issue certificates in accordance with a certificate policy that specifies adequate security
644 controls.

645 When an agency is obtaining a certificate for a TLS server for which all the clients are under the
646 agency's control, the agency may issue the certificate from its own CA if it can configure the
647 clients to trust that CA. In other cases, the agency should obtain a certificate from a publicly-
648 trusted CA; a CA that clients that will be connecting to the server have already been configured
649 to trust.

650 **3.3 Cryptographic Support**

651 Cryptographic support in TLS is provided through the use of various cipher suites. A cipher suite

¹⁵ A CRL is considered "cogent" when the "CRL Scope" [20] is appropriate for the certificate in question.

652 specifies a collection of algorithms for key exchange (in TLS 1.2 and earlier only), and for
653 providing confidentiality and integrity services to application data. The cipher suite negotiation
654 occurs during the TLS handshake protocol. The client presents cipher suites that it supports to
655 the server, and the server selects one of them to secure the session data.

656 In addition to the selection of appropriate cipher suites, system administrators may also have
657 additional considerations specific to the implementation of the cryptographic algorithms, as well
658 as cryptographic module validation requirements. Acceptable cipher suites are listed in Section
659 3.3.1, grouped by certificate type and protocol version. Implementation considerations are
660 discussed in Section 3.3.2, and recommendations regarding cryptographic module validation are
661 described in Section 3.3.3.

662 3.3.1 Cipher Suites

663 Cipher suites specify the cryptographic algorithms that will be used for a session. Cipher suites
664 in TLS 1.0 through TLS 1.2 have the form:

665 *TLS_KeyExchangeAlg_WITH_EncryptionAlg_MessageAuthenticationAlg*

666 For example, the cipher suite *TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA* uses ephemeral
667 ECDH key establishment, with parameters signed using RSA, confidentiality is provided by
668 AES-128 in cipher block chaining mode, and message authentication is performed using
669 HMAC_SHA.¹⁶ For further information on cipher suite interpretation, see Appendix B.

670 Cipher suites are formatted differently in TLS 1.3. These cipher suites do not specify the key
671 exchange algorithm, and have the form:

672 *TLS_AEAD_HASH*

673 For example, the cipher suite *TLS_AES_128_GCM_SHA256* uses AES-128 in Galois Counter
674 Mode for confidentiality and message authentication, and uses SHA-256 for the HKDF. TLS 1.3
675 cipher suites cannot be used for TLS 1.2 connections, and TLS 1.2 cipher suites cannot be
676 negotiated with TLS 1.3.

677 When negotiating a cipher suite, the client sends a handshake message with a list of cipher suites
678 it will accept. The server chooses from the list and sends a handshake message back indicating
679 which cipher suite it will accept. Although the client may order the list with the strongest cipher
680 suites listed first, the server may choose *any* of the cipher suites proposed by the client.
681 Therefore, there is *no* guarantee that the negotiation will settle on the strongest common suite. If
682 no cipher suites are common to the client and server, the connection is aborted.

683 The server **shall** be configured to only use cipher suites that are composed entirely of NIST-
684 approved algorithms (i.e., [7, 8, 11, 27-29, 65-67, 69]). A complete list of acceptable cipher
685 suites for general use is provided in this section, grouped by certificate type and TLS protocol

¹⁶ SHA indicates the use of the SHA-1 hash algorithm.

686 version. The Internet Assigned Numbers Authority (IANA) value for each cipher suite is given
687 after its text description, in parentheses.¹⁷

688 In some situations, such as closed environments, it may be appropriate to use pre-shared keys.
689 Pre-shared keys are symmetric keys that are already in place prior to the initiation of a TLS
690 session, which are used in the derivation of the premaster secret. For cipher suites that are
691 acceptable in pre-shared key environments, see Appendix C.

692 The following cipher suite listings are grouped by certificate type and TLS protocol version. The
693 cipher suites in these lists include the cipher suites that contain NIST-approved cryptographic
694 algorithms. Cipher suites that do not appear in this section or in Appendix C **shall not** be used.

695 Cipher suites using ephemeral DH and ephemeral ECDH (i.e., those with DHE or ECDHE in the
696 second mnemonic) provide perfect forward secrecy.¹⁸ When ephemeral keys are used to establish
697 the master secret, each ephemeral key-pair (i.e., the server ephemeral key-pair and the client
698 ephemeral key-pair) **shall** have at least 112 bits of security.

699 3.3.1.1 Cipher Suites for TLS 1.2 and Earlier Versions

700 The first revision of this guidance required support for a small set of cipher suites to promote
701 interoperability and align with TLS specifications. There are no longer any mandatory cipher
702 suite requirements. Cipher suites that comprise AES and other NIST-approved algorithms are
703 acceptable to use, although they are not necessarily equal in terms of security. Cipher suites that
704 use TDEA (3DES) are no longer allowed, due to the limited amounts of data that can be
705 processed under a single key. The server **shall** be configured to only use cipher suites for which
706 it has a valid certificate containing a signature providing at least 112 bits of security.

707 By removing requirements that specific cipher suites be supported, system administrators have
708 more freedom to meet the needs of their environment and applications. It also increases agility
709 by allowing administrators to immediately disable cipher suites when attacks are discovered
710 without breaking compliance.

711 If a subset of the cipher suites that are acceptable for the server certificate(s) are supported, the
712 following list gives general guidance on choosing the strongest options:

- 713 1. Prefer ephemeral keys over static keys (i.e., prefer DHE over DH, and prefer ECDHE
714 over ECDH). Ephemeral keys provide perfect forward secrecy.
- 715 2. Prefer GCM or CCM modes over CBC mode. The use of an authenticated encryption
716 mode prevents several attacks (see Section 3.3.2 for more information). Note that these
717 are not available in versions prior to TLS 1.2.

¹⁷ The full list of IANA values for TLS parameters can be found at <https://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml>.

¹⁸ Perfect forward secrecy is the condition in which the compromise of a long-term private key used in deriving a session key subsequent to the derivation does not cause the compromise of the session key.

718 3. Prefer CCM over CCM_8. The latter contains a shorter authentication tag, which
719 provides a lower authentication strength.

720 This list does not have to be strictly followed, as some environments or applications may
721 have special circumstances. Note that this list may become outdated if an attack emerges on
722 one of the preferred components. If an attack significantly impacts the recommended cipher
723 suites, NIST will address the issue in an announcement on the NIST Computer Security
724 Resource Center.

725 3.3.1.1.1 Cipher Suites for ECDSA Certificates

726 TLS version 1.2 includes authenticated encryption modes, and support for the SHA-256 and
727 SHA-384 hash algorithms, which are not supported in prior versions of TLS. These cipher suites
728 are described in [59] and [55]. TLS 1.2 servers that are configured with ECDSA certificates may
729 be configured to support the following cipher suites, which are only supported by TLS 1.2:

- 730 • TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xC0, 0x2B)
- 731 • TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xC0, 0x2C)
- 732 • TLS_ECDHE_ECDSA_WITH_AES_128_CCM (0xC0, 0xAC)
- 733 • TLS_ECDHE_ECDSA_WITH_AES_256_CCM (0xC0, 0xAD)
- 734 • TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 (0xC0, 0xAE)
- 735 • TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8 (0xC0, 0xAF)
- 736 • TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 (0xC0, 0x23)
- 737 • TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 (0xC0, 0x24)

738 TLS servers may be configured to support the following cipher suites when ECDSA certificates
739 are used with TLS versions 1.2, 1.1, or 1.0:

- 740 • TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA¹⁹ (0xC0, 0x09)
- 741 • TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xC0, 0x0A)

742 3.3.1.1.2 Cipher Suites for RSA Certificates

743 TLS 1.2 servers that are configured with RSA certificates may be configured to support the
744 following cipher suites:

- 745 • TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xC0, 0x2F)
- 746 • TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xC0, 0x30)
- 747 • TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 (0x00, 0x9E)

¹⁹ In TLS versions 1.0 and 1.1, DHE and ECDHE cipher suites use SHA-1 for signature generation on the ephemeral parameters (including keys) in the ServerKeyExchange message. While the use of SHA-1 for digital signature generation is generally disallowed by [10], exceptions can be granted by protocol-specific guidance. SHA-1 is allowed for generating digital signatures on ephemeral parameters in TLS. Due to the random nature of the ephemeral keys, a third party is unlikely to cause effective collision. The server and client do not have anything to gain by causing a collision for the connection. Because of the client random and server random values, the server, the client, or a third party cannot use a colliding set of messages to masquerade as the client or server in future connections. Any modification to the parameters by a third party during the handshake will ultimately result in a failed connection.

- 748 • TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (0x00, 0x9F)
- 749 • TLS_DHE_RSA_WITH_AES_128_CCM (0xC0, 0x9E)
- 750 • TLS_DHE_RSA_WITH_AES_256_CCM (0xC0, 0x9F)
- 751 • TLS_DHE_RSA_WITH_AES_128_CCM_8 (0xC0, 0xA2)
- 752 • TLS_DHE_RSA_WITH_AES_256_CCM_8 (0xC0, 0xA3)
- 753 • TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xC0, 0x27)
- 754 • TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xC0, 0x28)
- 755 • TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 (0x00, 0x67)
- 756 • TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 (0x00, 0x6B)

757 TLS servers may be configured to support the following cipher suites when RSA certificates are
758 used with TLS versions 1.2, 1.1, or 1.0:

- 759 • TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xC0, 0x13)
- 760 • TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xC0, 0x14)
- 761 • TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x00, 0x33)
- 762 • TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x00, 0x39)

763 3.3.1.1.3 Cipher Suites for DSA Certificates

764 TLS 1.2 servers that are configured with DSA certificates may be configured to support the
765 following cipher suites:

- 766 • TLS_DHE_DSS_WITH_AES_128_GCM_SHA256 (0x00, 0xA2)
- 767 • TLS_DHE_DSS_WITH_AES_256_GCM_SHA384 (0x00, 0xA3)
- 768 • TLS_DHE_DSS_WITH_AES_128_CBC_SHA256 (0x00, 0x40)
- 769 • TLS_DHE_DSS_WITH_AES_256_CBC_SHA256 (0x00, 0x6A)

770 TLS servers may be configured to support the following cipher suites when DSA certificates are
771 used with TLS versions 1.2, 1.1, or 1.0:

- 772 • TLS_DHE_DSS_WITH_AES_128_CBC_SHA (0x00, 0x32)
- 773 • TLS_DHE_DSS_WITH_AES_256_CBC_SHA (0x00, 0x38)

774 3.3.1.1.4 Cipher Suites for DH Certificates

775 DH certificates contain a static key, and are signed using either DSA or RSA. Unlike cipher
776 suites that use ephemeral DH, these cipher suites contain static DH parameters. While the use of
777 static keys is technically acceptable, the use of ephemeral key cipher suites is encouraged and
778 preferred over the use of the cipher suites listed in this section.

779 TLS 1.2 servers that are configured with DSA-signed DH certificates may be configured to
780 support the following cipher suites:

- 781 • TLS_DH_DSS_WITH_AES_128_GCM_SHA256 (0x00, 0xA4)
- 782 • TLS_DH_DSS_WITH_AES_256_GCM_SHA384 (0x00, 0xA5)
- 783 • TLS_DH_DSS_WITH_AES_128_CBC_SHA256 (0x00, 0x3E)

- 784 • TLS_DH_DSS_WITH_AES_256_CBC_SHA256 (0x00, 0x68)

785 TLS servers may be configured to support the following cipher suites when DSA-signed DH
786 certificates are used with TLS versions 1.2, 1.1, or 1.0:

- 787 • TLS_DH_DSS_WITH_AES_128_CBC_SHA (0x00, 0x30)
788 • TLS_DH_DSS_WITH_AES_256_CBC_SHA (0x00, 0x36)

789 TLS 1.2 servers that are configured with RSA-signed DH certificates may be configured to
790 support the following cipher suites:

- 791 • TLS_DH_RSA_WITH_AES_128_GCM_SHA256 (0x00, 0xA0)
792 • TLS_DH_RSA_WITH_AES_256_GCM_SHA384 (0x00, 0xA1)
793 • TLS_DH_RSA_WITH_AES_128_CBC_SHA256 (0x00, 0x3F)
794 • TLS_DH_RSA_WITH_AES_256_CBC_SHA256 (0x00, 0x69)

795 TLS servers may be configured to support the following cipher suites when RSA-signed DH
796 certificates are used with TLS versions 1.2, 1.1, or 1.0:

- 797 • TLS_DH_RSA_WITH_AES_128_CBC_SHA (0x00, 0x31)
798 • TLS_DH_RSA_WITH_AES_256_CBC_SHA (0x00, 0x37)

799 **3.3.1.1.5 Cipher Suites for ECDH Certificates**

800 ECDH certificates contain a static key, and are signed using either ECDSA or RSA. Unlike
801 cipher suites that use ephemeral ECDH, these cipher suites contain static ECDH parameters. The
802 use of ephemeral key cipher suites is encouraged and preferred over the use of the cipher suites
803 listed in this section.

804 TLS 1.2 servers that are configured with ECDSA-signed ECDH certificates may be configured
805 to support the following cipher suites:

- 806 • TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256 (0xC0, 0x2D)
807 • TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384 (0xC0, 0x2E)
808 • TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256 (0xC0, 0x25)
809 • TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384 (0xC0, 0x26)

810 TLS servers may be configured to support the following cipher suites when ECDSA-signed
811 ECDH certificates are used with TLS versions 1.2, 1.1, or 1.0:

- 812 • TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA (0xC0, 0x04)
813 • TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA (0xC0, 0x05)

814 TLS 1.2 servers that are configured with RSA-signed ECDH certificates may be configured to
815 support the following cipher suites:

- 816 • TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256 (0xC0, 0x31)

- 817 • TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384 (0xC0, 0x32)
- 818 • TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256 (0xC0, 0x29)
- 819 • TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384 (0xC0, 0x2A)

820 TLS servers may be configured to support the following cipher suites when RSA-signed ECDH
821 certificates are used with TLS versions 1.2, 1.1, or 1.0:

- 822 • TLS_ECDH_RSA_WITH_AES_128_CBC_SHA (0xC0, 0x0E)
- 823 • TLS_ECDH_RSA_WITH_AES_256_CBC_SHA (0xC0, 0x0F)

824 3.3.1.2 Cipher Suites for TLS 1.3

825 TLS 1.3 servers may be configured to support the following cipher suites:

- 826 • TLS_AES_128_GCM_SHA256 (013x, 0x01)
- 827 • TLS_AES_256_GCM_SHA384 (0x13, 0x02)
- 828 • TLS_AES_128_CCM_SHA256 (0x13, 0x04)
- 829 • TLS_AES_128_CCM_8_SHA256 (0x13, 0x05)

830 These cipher suites may be used with either RSA or ECDSA server certificates; DSA and DH
831 certificates cannot be used with TLS 1.3. These cipher suites may also be used with pre-shared
832 keys, as specified in Appendix C.

833 3.3.2 Implementation Considerations

834 System administrators need to fully understand the ramifications of selecting cipher suites and
835 configuring applications to support only those cipher suites. The security guarantees of the
836 cryptography are limited to the weakest cipher suite supported by the configuration. When
837 configuring an implementation, there are several factors that affect the selection of supported
838 cipher suites.

839 RFC 4346 [25] describes timing attacks on CBC cipher suites, as well mitigation techniques.
840 TLS implementations **shall** use the bad_record_mac error to indicate a padding error when
841 communications are secured using a CBC cipher suite. Implementations **shall** compute the MAC
842 regardless of whether padding errors exist.

843 In addition to the CBC attacks addressed in RFC 4346 [25], the Lucky 13 attack [2]
844 demonstrates that a constant-time decryption routine is also needed to prevent timing attacks.
845 TLS implementations **should** support constant-time decryption, or near constant-time
846 decryption.

847 The POODLE attack exploits nondeterministic padding in SSL 3.0 [49]. The vulnerability does
848 not exist in the TLS protocols, but the vulnerability can exist in a TLS implementation when the
849 SSL decoder code is reused to process TLS data [45]. TLS implementations **shall** correctly
850 decode the CBC padding bytes.

851 Note that CBC-based attacks can be prevented by using AEAD cipher suites (e.g., GCM, CCM),
852 which are supported in TLS 1.2.

853 3.3.2.1 Algorithm Support

854 Many TLS servers and clients support cipher suites that are not composed of only NIST-
855 approved algorithms. If the server were configured to support cipher suites that are not
856 recommended in this document, they may be chosen during the handshake. Therefore, it is
857 important that the server is configured to only use recommended cipher suites. This is
858 particularly important for server implementations that do not allow the server administrator to
859 specify preference order. In such servers, the only way to ensure that a server uses NIST-
860 approved algorithms for encryption is to disable cipher suites that use other encryption
861 algorithms.

862 If the server implementation does allow the server administrator to specify a preference, the
863 system administrator is encouraged to use the preference recommendations listed in Section
864 3.3.1.1.

865 3.3.2.2 Cipher Suite Scope

866 The selection of a cryptographic algorithm may be system-wide and not application specific for
867 some implementations. For example, disabling an algorithm for one application on a system
868 might disable that algorithm for all applications on that system.

869 3.3.3 Validated Cryptography

870 The cryptographic module used by the server **shall** be a FIPS 140-validated cryptographic
871 module [70]. All cryptographic algorithms that are included in the configured cipher suites **shall**
872 be within the scope of the validation, as well as the random number generator. Note that the TLS
873 1.1 pseudorandom function (PRF) uses MD5 and SHA-1 in parallel so that if one hash function
874 is broken, security is not compromised. While MD5 is not a NIST-approved algorithm, the TLS
875 1.1 PRF is specified as acceptable in SP 800-135 [22]. TLS 1.3 uses the HMAC-based Extract-
876 and-Expand Key Derivation Function (HKDF), described in RFC 5869 [43], to derive the
877 session keys. Note that in TLS 1.1, the use of SHA-1 is found acceptable for specific cases of
878 signing ephemeral keys and for signing for client authentication. This is acceptable due the
879 difficulty for a third party to cause a collision that is not detected, and the client and server
880 cannot exploit the collision they can cause, as further explained in footnote 19. In TLS 1.2, the
881 default hash function in the PRF is SHA-256. Other than the SHA-1 exception listed for specific
882 instances above, all cryptography used **shall** provide at least 112 bits of security. All server and
883 client certificates **shall** contain public keys that offer at least 112 bits of security. All server and
884 client certificates and certificates in their certification paths **shall** be signed using key pairs that
885 offer at least 112 bits of security and SHA-224 or a stronger hashing algorithm. All ephemeral
886 keys used by the client and server **shall** offer at least 112 bits of security. All symmetric
887 algorithms used to protect the TLS data **shall** use keys that offer at least 112 bits of security.

888 The random number generator **shall** be tested and validated in accordance with SP 800-90A [9]
889 under the NIST Cryptographic Algorithm Validation Program (CAVP) and successful results of
890 this testing **shall** be indicated on the cryptographic module's FIPS 140 validation certificate.

891 The server random value, sent in the ServerHello message, contains a 4-byte timestamp²⁰ value
892 and 28-byte random value in TLS version 1.0, 1.1, and 1.2, and contains a 32-byte random value
893 in TLS 1.3. The validated random number generator **shall** be used to generate the random bytes
894 of the server random value.²¹ The validated random number generator **should** be used to
895 generate the 4-byte timestamp of the server random value.

896 **3.4 TLS Extension Support**

897 Several TLS extensions are described in RFCs. This section contains recommendations for a
898 subset of the TLS extensions that the Federal agencies **shall**, **should**, or **should not** use as they
899 become prevalent in commercially available TLS servers and clients.

900 System administrators must carefully consider the risks of supporting extensions that are not
901 listed as mandatory. Only extensions whose specification have an impact on security are
902 discussed here, but the reader is advised that supporting any extension can have unintended
903 security consequences. In particular, enabling extensions increases the potential for
904 implementation flaws and could leave a system vulnerable. For example, the Heartbleed bug [72]
905 was a flaw in an implementation of the heartbeat extension [62]. Although the extension has no
906 inherent security implications, the implementation flaw exposed server data, including private
907 keys, to attackers.

908 In general, it is advised that servers only be configured to support extensions that are required by
909 the application or enhance security. Extensions that are not needed **should not** be enabled.

910 **3.4.1 Mandatory TLS Extensions**

911 The server **shall** support the use of the following TLS extensions.

- 912 1. Renegotiation Indication
- 913 2. Server Name Indication
- 914 3. Session Hash and Extended Master Secret
- 915 4. Signature Algorithms
- 916 5. Certificate Status Request extension

917 **3.4.1.1 Renegotiation Indication**

918 In TLS versions 1.0 to 1.2, session renegotiation is vulnerable to an attack in which the attacker
919 forms a TLS connection with the target server, injects content of its choice, and then splices in a
920 new TLS connection from a legitimate client. The server treats the legitimate client's initial TLS
921 handshake as a renegotiation of the attacker's negotiated session and thus believes that the initial

²⁰ The timestamp value does not need to be correct in TLS. It can be any 4-byte value, unless otherwise restricted by higher-level or application protocols.

²¹ TLS 1.3 implementations include a downgrade protection mechanism embedded in the random value that overwrites the last eight bytes of the server random value with a fixed value. When negotiating TLS 1.2, the last eight bytes of the server random will be set to 44 4F 57 4E 47 52 44 01. When TLS 1.1 or below is negotiated, the last eight bytes of the random value will be set to 44 4F 57 4E 47 52 44 00. This overwrite is separate from the validated random bit generator.

922 data transmitted by the attacker is from the legitimate client. The session renegotiation extension
923 is defined to prevent such a session splicing or session interception. The extension uses the
924 concept of cryptographically binding the initial session negotiation and session renegotiation.

925 Server implementations **shall** perform initial and subsequent renegotiations in accordance with
926 RFC 5746 [57] and [56].

927 **3.4.1.2 Server Name Indication**

928 Multiple virtual servers may exist at the same network address. The server name indication
929 extension allows the client to specify which of the servers located at the address it is trying to
930 connect with. This extension is available in all versions of TLS. The server **shall** be able to
931 process and respond to the server name indication extension received in a ClientHello message
932 as described in [30].

933 **3.4.1.3 Session Hash and Extended Master Secret**

934 Bhargavan et al. have shown that an active attacker can synchronize two TLS sessions such that
935 they share the same master secret, thus allowing the attacker to perform a man-in-the-middle
936 attack [13]. The Session Hash and Extended Master Secret extension, specified in RFC 7627
937 [42], prevents such attacks by binding the master secret to a hashed log of the full handshake.
938 The server **shall** support the use of this extension.

939 **3.4.1.4 Signature Algorithms**

940 Servers **shall** support the processing of the signature algorithms extension received in a
941 ClientHello message. The extension, its syntax, and processing rules are described in Sections
942 7.4.1.4.1, 7.4.2, and 7.4.3 of RFC 5246 [26] and Section 4.2.3 of the TLS 1.3 specification [56].
943 Note that the extension described in the TLS 1.3 specification updates the extension described in
944 RFC 5246 by adding an additional signature scheme.

945 **3.4.1.5 Certificate Status Request**

946 When the client wishes to receive the revocation status of the TLS server certificate from the
947 TLS server, the client includes the Certificate Status Request (status_request) extension in the
948 ClientHello message. Upon receipt of the status_request extension, a server with a certificate
949 issued by a CA that supports OCSP **shall** include the certificate status along with its certificate
950 by sending a CertificateStatus message immediately following the Certificate message.²² While
951 the extension itself is extensible, only OCSP-type certificate status is defined in [30]. This
952 extension is also called OCSP stapling.

953 **3.4.2 Conditional TLS Extensions**

954 Support the use of the following TLS extensions under the circumstances described in the

²² In TLS 1.3 the server includes the certificate status in the Certificate message.

955 following paragraphs:

- 956 1. The Fallback Signaling Cipher Suite Value (SCSV) extension **shall** be supported if the
957 server supports versions of TLS prior to TLS 1.2 and does not support TLS 1.3.
- 958 2. The Encrypt-then-MAC extension **shall** be supported if the server is configured to
959 negotiate CBC cipher suites.
- 960 3. The Negotiated Groups extension **shall be** supported if the server supports ephemeral
961 ECDH cipher suites or if the server supports TLS 1.3.
- 962 4. The EC Point Format extension **shall** be supported if the server supports EC cipher
963 suites.
- 964 5. The Multiple Certificate Status extension **should** be supported if status information for
965 the server's certificate is available via OCSP, and the extension is supported by the server
966 implementation.
- 967 6. The Trusted CA Indication extension **shall** be supported if the server communicates with
968 memory-constrained clients (e.g., low-memory client devices in the Internet of Things),
969 and the server has been issued certificates by multiple CAs.
- 970 7. The Truncated HMAC extension may be supported if the server communicates with
971 constrained device clients and the server implementation does not support variable-length
972 padding.
- 973 8. The Signed Certificate Timestamps extension **should** be supported if the server's
974 certificate was issued by a publicly trusted CA, and the certificate does not include a
975 Signed Certificate Timestamps List extension.
- 976 9. The Supported Versions, Cookie, and Key Share extensions **shall** be supported if the
977 server supports TLS 1.3.
- 978 10. The Pre-Shared Key extension may be supported if the server supports TLS 1.3.
- 979 11. The Pre-Shared Key Exchange Modes extension **shall** be supported if the server supports
980 TLS 1.3 and the Pre-Shared Key extension.

981 **3.4.2.1 Fallback Signaling Cipher Suite Value (SCSV)**

982 TLS 1.3 includes a downgrade protection mechanism that previous versions do not. In versions
983 prior to TLS 1.3, an attacker can use an external version negotiation means to force unnecessary
984 protocol downgrades on a connection. In particular, the attacker can make it appear that the
985 connection failed with the requested TLS version, and some client implementations will try the
986 connection again with a downgraded protocol version. This extension, described in RFC 7507
987 [48], provides a mechanism to prevent unintended protocol downgrades. Clients signal when a
988 connection is a fallback, and if the server deems it inappropriate (i.e., the server supports a higher
989 TLS version), the server returns a fatal alert.

990 When TLS versions prior to TLS 1.2 are supported by the server, and TLS version 1.3 is not
991 supported, the fallback SCSV extension **shall** be supported.

992 **3.4.2.2 Encrypt-then-MAC**

993 Several attacks on CBC cipher suites have been possible due to the MAC-then-encrypt order of
994 operations used in TLS versions 1.0, 1.1, and 1.2. The Encrypt-then-MAC extension alters the
995 order that the encryption and MAC operations are applied to the data. This is believed to provide

996 stronger security, and mitigate or prevent several known attacks on CBC cipher suites. Servers
997 that are configured to negotiate CBC cipher suites **shall** support this extension as described in
998 [\[36\]](#).

999 **3.4.2.3 Negotiated Groups**

1000 The Negotiated Groups extension²³ (supported_groups) allows the client to indicate the groups
1001 that it supports to the server. The extension was originally called the Supported Elliptic Curves
1002 extension (elliptic_curves), and was only used for elliptic curve groups, but it may now also be
1003 used to negotiate finite field groups. In TLS 1.3, the Negotiated Groups extension must be used
1004 to negotiate both elliptic curve and finite field groups. Servers that support either ephemeral
1005 ECDH cipher suites or TLS 1.3 **shall** support this extension. When elliptic curve cipher suites
1006 are configured, at least one of the NIST-approved curves, P-256 (secp256r1) and P-384
1007 (secp384r1), **shall** be supported as described in RFC 4492 [\[14\]](#). The finite field groups
1008 ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, and ffdhe8192 may be supported (see RFC 7919
1009 [\[35\]](#)).

1010 **3.4.2.4 Key Share**

1011 The Key Share extension is used in TLS 1.3 to send cryptographic parameters. Servers that
1012 support TLS 1.3 **shall** support this extension as described in Section 4.2.7 of the TLS 1.3
1013 specification [\[56\]](#).

1014 **3.4.2.5 EC Point Format**

1015 Servers that support EC cipher suites **shall** be able to process the supported EC point format
1016 received in the ClientHello message by the client. The servers **shall** process this extension in
1017 accordance with Section 5.1 of RFC 4492 [\[14\]](#).

1018 Servers that support EC cipher suites **shall** also be able to send the supported EC point format in
1019 the ServerHello message as described in Section 5.2 of RFC 4492 [\[14\]](#).

1020 **3.4.2.6 Multiple Certificate Status**

1021 The multiple certificate status extension improves on the Certificate Status Request extension
1022 described in Section 3.4.1.5 by allowing the client to request the status of all certificates provided
1023 by the server in the TLS handshake. When the server returns the revocation status of all the
1024 certificates in the server certificate chain, the client does not need to query any revocation service
1025 providers, such as OCSP responders. This extension is documented in RFC 6961 [\[51\]](#). Servers
1026 that have this capability and that have certificates issued by CAs that support OCSP **should** be
1027 configured to support this extension.

1028 **3.4.2.7 Trusted CA Indication**

1029 The trusted CA indication (trusted_ca_keys) extension allows a client to specify which CA root

²³ Called “Supported Groups” in RFC 7919.

1030 keys it possesses. This is useful for sessions where the client is memory-constrained and
1031 possesses a small number of root CA keys. Servers that communicate with memory-constrained
1032 clients and that have been issued certificates by multiple CAs **shall** be able to process and
1033 respond to the trusted CA indication extension received in a ClientHello message as described in
1034 [\[30\]](#).

1035 **3.4.2.8 Truncated HMAC**

1036 The Truncated HMAC extension allows a truncation of the HMAC output to 80 bits for use as a
1037 MAC tag. An 80-bit MAC tag complies with the recommendations in SP 800-107 [\[21\]](#), but
1038 reduces the security provided by the integrity algorithm. Because forging a MAC tag is an online
1039 attack, and the TLS session will terminate immediately when an invalid MAC tag is encountered,
1040 the risk introduced by using this extension is low. However, truncated MAC tags **shall not** be
1041 used in conjunction with variable-length padding, due to attacks described by Paterson et al.
1042 [\[50\]](#). This extension cannot be used with TLS 1.3.

1043 **3.4.2.9 Pre-Shared Key**

1044 The Pre-Shared Key extension (`pre_shared_key`), available in TLS 1.3, is used to indicate the
1045 identity of the pre-shared key to be used for PSK key establishment. In TLS 1.3 pre-shared keys
1046 may either be established out-of-band, as in TLS 1.2 are below, or in a previous connection, in
1047 which case they are used for session resumption. Servers that support TLS 1.3 may be
1048 configured to support this extension in order to support session resumption or to support the use
1049 of pre-shared keys that are established out-of-band.

1050 **3.4.2.10 Pre-Shared Key Exchange Modes**

1051 A TLS 1.3 client must send the Pre-Shared Key Exchange Modes extension
1052 (`psk_key_exchange_modes`) if it sends the Pre-Shared Key extension. TLS 1.3 servers use the
1053 list of key exchange modes present in the extension to select an appropriate key exchange
1054 method. TLS servers that support TLS 1.3 and the Pre-Shared Key extension **shall** support this
1055 extension.

1056 **3.4.2.11 Signed Certificate Timestamps**

1057 The Certificate Transparency project (described in RFC 6962 [\[46\]](#)) strives to reduce the impact
1058 of certificate-based threats by making the issuance of CA-signed certificates more transparent.
1059 This is done through the use of public logs of certificates, public log monitoring, and public
1060 certificate auditing. Certificate logs are cryptographically assured records of certificates that are
1061 open to public scrutiny. Certificates may be appended to logs, but they cannot be removed,
1062 modified, or inserted into the middle of a log. Monitors watch certificate logs for suspicious
1063 certificates, such as those that were not authorized by the domain they claim to represent.
1064 Auditors have the ability to check the membership of a particular certificate in a log, as well as
1065 verify the integrity and consistency of logs.

1066 Evidence that the server's certificate has been submitted to Certificate Transparency logs may be
1067 provided to clients either in the certificate itself or in a Signed Certificate Timestamps TLS
1068 extension (`signed_certificate_timestamp`). Servers with certificates issued by publicly trusted

1069 CAs that do not include a Signed Certificate Timestamps List extension **should** support the
1070 Signed Certificate Timestamps TLS extension.

1071 **3.4.2.12 Supported Versions**

1072 The supported versions extension was added in TLS 1.3. The extension is sent in the ClientHello
1073 message to indicate which versions of TLS the client supports. A TLS 1.3 server **shall** be able to
1074 process this extension. When it is absent from the ClientHello message, the server **shall** use the
1075 version negotiation specified in TLS 1.2 and earlier.

1076 **3.4.2.13 Cookie**

1077 The cookie extension was added in TLS 1.3. It allows the server to force the client to prove that
1078 it is reachable at its apparent network address, and offload state to the client. Servers that support
1079 TLS 1.3 may support the cookie extension in accordance with the TLS 1.3 specification [56].

1080 **3.4.3 Discouraged TLS Extensions**

1081 The following extension **should not** be used:

- 1082 1. Client Certificate URL
- 1083 2. Early Data Indication

1084 **3.4.3.1 Client Certificate URL**

1085 The Client Certificate URL extension allows a client to send a URL pointing to a certificate,
1086 rather than sending a certificate to the server during mutual authentication. This can be very
1087 useful for mutual authentication with constrained clients. However, this extension can be used
1088 for malicious purposes. The URL could belong to an innocent server on which the client would
1089 like to perform a denial of service attack, turning the TLS server into an attacker. A server that
1090 supports this extension also acts as a client while retrieving a certificate, and therefore becomes
1091 subject to additional security concerns. For these reasons, the Client Certificate URL extension
1092 **should not** be supported. However, if an agency determines that the risks are minimal, and this
1093 extension is needed for environments where clients are in constrained devices, the extension may
1094 be supported. If the client certificate URL extension is supported, the server **shall** be configured
1095 to mitigate the security concerns described above and in Section 11.3 of [30].

1096 **3.4.3.2 Early Data Indication**

1097 In TLS 1.3, the Early Data Indication extension (early_data) allows the client to send application
1098 data in the ClientHello message when pre-shared keys are used. This includes pre-shared keys
1099 that are established out-of-band, as well those used for session resumption. TLS does not protect
1100 this early data against replay attacks. Servers **should not** process early data received in the
1101 ClientHello message. If the server is configured to send the Early Data Indication extension, the
1102 server **shall** use methods of replay protection, such as those described in Section 8 of the TLS
1103 1.3 specification [56].

1104 3.5 Client Authentication

1105 Where strong cryptographic client authentication is required, TLS servers may use the TLS
1106 protocol client authentication option to request a client certificate to cryptographically
1107 authenticate the client.²⁴ For example, the Personal Identity Verification (PIV) Authentication
1108 certificate [68] (and the associated private key) provides a suitable option for strong
1109 authentication of Federal employees and contractors. To ensure that agencies are positioned to
1110 take full advantage of the PIV Card, all TLS servers that perform client authentication **shall**
1111 implement certificate-based client authentication.

1112 The client authentication option requires the server to implement the X.509 path validation
1113 mechanism and a trust anchor store. Requirements for these mechanisms are specified in
1114 Sections 3.5.1 and 3.5.2, respectively. To ensure that cryptographic authentication actually
1115 results in strong authentication, client keys **shall** contain at least 112 bits of security. Section
1116 3.5.3 describes mechanisms that can contribute, albeit indirectly, to enforcing this requirement.
1117 Section 3.5.4 describes the client's use of the server hints list.

1118 The TLS server **shall** be configurable to terminate the connection with a fatal "handshake
1119 failure" alert when a client certificate is requested, and the client does not have a suitable
1120 certificate.

1121 3.5.1 Path Validation

1122 The client certificate **shall** be validated in accordance with the certification path validation rules
1123 specified in Section 6 of [20]. In addition, the revocation status of each certificate in the
1124 certification path **shall** be validated using the Online Certificate Status Protocol (OCSP) or a
1125 certificate revocation list (CRL). OCSP checking **shall** be in compliance with RFC 6960 [61].

1126 Revocation information **shall** be obtained as described in Section 3.2.2.

1127 The server **shall** be able to determine the certificate policies that the client certificate is trusted
1128 for by using the certification path validation rules specified in Section 6 of [20]. Server and
1129 backend applications may use this determination to accept or reject the certificate. Checking
1130 certificate policies assures the server that only client certificates that have been issued with
1131 acceptable assurance, in terms of CA and registration system and process security, are accepted.

1132 Not all commercial products may support the public-key certification path validation and
1133 certificate policy processing rules listed and cited above. When implementing client
1134 authentication, the Federal agencies **shall** either use the commercial products that meet these

²⁴ The CertificateVerify message is sent to explicitly verify a client certificate that has a signing capability. In TLS 1.1 (and TLS 1.0), this message uses SHA-1 to generate a signature on all handshake messages that came before it. SP 800-131A [10] states that the use of SHA-1 for digital signature generation is disallowed after 2013. Even if a collision is found, the client must use its private key to authenticate itself by signing the hash. Due to the client random and server random values, the server, the client, or a third party cannot use a colliding set of messages to masquerade as the client or server in future connections. Any modification to this message, preceding messages, or subsequent messages will ultimately result in a failed connection. Therefore, SHA-1 is allowed for generating digital signatures in the TLS CertificateVerify message.

1135 requirements or augment commercial products to meet these requirements.

1136 The server **shall** be able to provide the client certificate, and the certificate policies for which the
1137 client certification path is valid, to the applications in order to support access control decisions.

1138 **3.5.2 Trust Anchor Store**

1139 Having an excessive number of trust anchors installed in the TLS application can expose the
1140 application to all the PKIs emanating from those trust anchors. The best way to minimize the
1141 exposure is to only include the trust anchors in the trust anchor store that are absolutely
1142 necessary for client public-key certificate authentication.

1143 The server **shall** be configured with only the trust anchors that the server trusts, and of those,
1144 only the ones that are required to authenticate the clients, in the case where the server supports
1145 client authentication in TLS. These trust anchors are typically a small subset of the trust anchors
1146 that may be included on the server by default. Also, note that this trust anchor store is distinct
1147 from the machine trust anchor store. Thus, the default set of trust anchors **shall** be examined to
1148 determine if any of them are required for client authentication. Some specific enterprise and/or
1149 PKI service provider trust anchor may need to be added.

1150 In the U.S. Federal environment, in most situations, the Federal Common Policy Root or the
1151 agency root (if cross certified with the Federal Bridge Certification Authority or the Federal
1152 Common Policy Root) should be sufficient to build a certification path to the client certificates.

1153 System administrators of a TLS server that supports certificate-based client authentication **shall**
1154 perform an analysis of the client certificate issuers and use that information to determine the
1155 minimum set of trust anchors required for the server. The server **shall** be configured to only use
1156 those trust anchors.

1157 **3.5.3 Checking the Client Key Size**

1158 The only direct mechanism for a server to check whether the key size and algorithms presented
1159 in a client public-key certificate are acceptable is for the server to examine the public key and
1160 algorithm in the client's certificate. An indirect mechanism is to check that the certificate
1161 policies extension in the client public-key certificate indicates the minimum cryptographic
1162 strength of the signature and hashing algorithms used, and for the server to perform certificate
1163 policy processing and checking. The server **shall** check the client key length if client
1164 authentication is performed, and the server implementation provides a mechanism to do so.
1165 Federal Agencies **shall** use the key size guidelines provided in [\[10\]](#) to check the client key size.

1166 **3.5.4 Server Hints List**

1167 Clients may use the list of trust anchors sent by the server in the CertificateRequest message to
1168 determine if the client's certification path terminates at one of these trust anchors. The list sent
1169 by the server is known as a "hints list." When the server and client are in different PKI domains,
1170 and the trust is established via direct cross-certification between the two PKI domains (i.e., the
1171 server PKI domain and the client PKI domain) or via transitive cross-certification (i.e., through
1172 cross-certifications among multiple PKI domains), the client may erroneously decide that its

1173 certificate will not be accepted by the server since the client's trust anchor is not sent in the hints
1174 list. To mitigate this failure, the server **shall** either 1) maintain the trust anchors of the various
1175 PKIs whose subscribers are the potential clients for the server, and include them in the hints list,
1176 or 2) be configured to send an empty hints list so that the client can always provide a certificate it
1177 possesses. The hints list **shall** be distinct from the server's trust anchor store.²⁵ In other words,
1178 the server **shall** continue to only populate its trust anchor store with the trust anchor of the
1179 server's PKI domain and the domains it needs to trust directly for client authentication. Note that
1180 the distinction between the server hints list and the server's own trust store is as follows: 1) the
1181 hints list is the list of trust anchors that a potential client might trust; and 2) the server's trust
1182 store is the list of trust anchors that the server explicitly trusts.

1183 **3.6 Session Resumption**

1184 Previous TLS sessions can be resumed, allowing for a connection to be established using an
1185 abbreviated handshake. All versions of TLS offer session resumption, although the mechanism
1186 for performing resumption differs. A server may be configured to ignore requests to resume a
1187 session, if the implementation allows it.

1188 TLS 1.3 allows the client to send data in the first flight of handshake, known as 0-RTT data. This
1189 practice may provide opportunities for attackers, such as replay attacks.²⁶ The TLS 1.3
1190 specification describes two mechanisms to mitigate threats introduced by 0-RTT data. One of
1191 these mechanisms is single-use tickets, which allows each session ticket to be used only once. It
1192 may be difficult to implement this mechanism in an environment with distributed servers, as a
1193 session database must be shared between servers. ClientHello recording is a second mechanism
1194 that defends against replay attacks by recording a unique value derived from the ClientHello and
1195 rejecting duplicates. To limit the size of the list, the server can maintain a list only within a
1196 specified time window. In general, 0-RTT data **should not** be accepted by the server. If the
1197 server does allow 0-RTT data, then the server **should** use the single-use ticket mechanism in
1198 accordance with the TLS 1.3 specification (see Section 8 of [56]).

1199 **3.7 Compression Methods**

1200 The use of compression may enable attackers to perform attacks using compression-based side
1201 channels (e.g., [58], [12]). Because of this, only the null compression method, which disables
1202 TLS compression, **should** be used. If compression is used, the methods defined in RFC 3749
1203 [39] or RFC 3943 [34] may be used.

1204 **3.8 Operational Considerations**

1205 The sections above specify TLS-specific functionality. This functionality is necessary, but is not
1206 sufficient, to achieve security in an operational environment.

²⁵ Depending on the server and client trust anchors, the two lists could be identical, could have some trust anchors in common, or have no trust anchors in common.

²⁶ TLS does not inherently provide replay protection for 0-RTT data.

1207 Federal agencies **shall** ensure that TLS servers include appropriate network security protections
1208 as specified in other NIST guidelines, such as SP 800-53 [\[41\]](#).

1209 The server **shall** operate on a secure operating system.²⁷ Where the server relies on a FIPS 140
1210 Level 1 cryptographic module, the software and private key **shall** be protected using the
1211 operating system identification, authentication and access control mechanisms. In some highly
1212 sensitive applications, server private keys may require protection using a FIPS 140 Level 2 or
1213 higher hardware cryptographic module.

1214 The server and associated platform **shall** be kept up-to-date in terms of security patches. This is
1215 critical to various aspects of security.

1216

²⁷ A secure operating system contains and uses the following features: operating system protection from applications and processes; operating system mediated isolation among applications and processes; user identification and authentication; access control based on authenticated user identity, and event logging of security-relevant activities.

1217 **4 Minimum Requirements for TLS Clients**

1218 This section provides a minimum set of requirements that a TLS client must meet in order to
 1219 adhere to these guidelines. Requirements are organized as follows: TLS protocol version
 1220 support; client keys and certificates; cryptographic support; TLS extension support; server
 1221 authentication; session resumption; compression methods; and operational considerations.

1222 Specific requirements are stated as either implementation requirements or configuration
 1223 requirements. Implementation requirements indicate that Federal agencies **shall not** procure TLS
 1224 client implementations unless they include the required functionality. Configuration
 1225 requirements indicate that system administrators are required to verify that particular features are
 1226 enabled, or in some cases, configured appropriately if present.

1227 **4.1 Protocol Version Support**

1228 The client **shall** be configured to use TLS 1.2 and **should** be configured to use TLS 1.3. The
 1229 client may be configured to use TLS 1.1 and TLS 1.0 to facilitate communication with private
 1230 sector servers. The client **shall not** be configured to use SSL 2.0 or SSL 3.0. Agencies **shall**
 1231 develop migration plans to support TLS 1.3 by January 1, 2020.

1232 **4.2 Client Keys and Certificates**

1233 Some applications may require client authentication. For TLS, this can be achieved by
 1234 performing mutual authentication using certificates.

1235 **4.2.1 Client Certificate Profile**

1236 When certificate-based client authentication is needed, the client **shall** be configured with a
 1237 certificate that adheres to the recommendations presented in this section. A client certificate may
 1238 be configured on the system or located on an external device (e.g., a PIV Card). For this
 1239 specification, the TLS client certificate **shall** be an X.509 version 3 certificate; both the public
 1240 key contained in the certificate and the signature **shall** provide at least 112 bits of security. If the
 1241 client supports TLS versions prior to TLS 1.2, the certificate **should** be signed with an algorithm
 1242 that is consistent with the public key:²⁸

- 1243 • Certificates containing RSA (signature), ECDSA, or DSA public keys **should** be signed
 1244 with those same signature algorithms, respectively;
- 1245 • Certificates containing Diffie-Hellman certificates **should** be signed with DSA; and
- 1246 • Certificates containing ECDH public keys **should** be signed with ECDSA.

1247 The client certificate profile is listed in Table 4-1. In the absence of an agency-specific client
 1248 certificate profile, this profile **should** be used for client certificates.

²⁸ This recommendation is an artifact of requirements in TLS 1.0 and 1.1.

1249

Table 4-1: TLS Client Certificate Profile

Field	Critical	Value	Description
Version	N/A	2	Version 3
Serial Number	N/A	Unique positive integer	Must be unique
Issuer Signature Algorithm	N/A	<i>Values by CA key type:</i>	
		sha256WithRSAEncryption {1 2 840 113549 1 1 11}, or stronger	CA with RSA key
		ecdsa-with-SHA256 {1 2 840 10045 4 3 2}, or stronger	CA with elliptic curve key
		id-dsa-with-sha256 {2 16 840 1 101 3 4 3 2}, or stronger	CA with DSA key
Issuer Distinguished Name	N/A	Unique X.500 Issuing CA DN	A single value shall be encoded in each RDN. All attributes that are of directoryString type shall be encoded as a printable string.
Validity Period	N/A	3 years or less	Dates through 2049 expressed in UTCTime
Subject Distinguished Name	N/A	Unique X.500 subject DN per agency requirements	A single value shall be encoded in each RDN. All attributes that are of directoryString type shall be encoded as a printable string.
Subject Public Key Information	N/A	<i>Values by certificate type:</i>	
		rsaEncryption {1 2 840 113549 1 1 1}	RSA signature certificate 2048-bit RSA key modulus, or other approved lengths as defined in [FIPS186-4] and [6] Parameters: NULL
		ecPublicKey {1 2 840 10045 2 1}	ECDSA signature certificate or ECDH certificate Parameters: namedCurve OID for names curve specified in FIPS 186-4. The curve shall be P-256 or P-384 SubjectPublic Key: Uncompressed EC Point.
		id-dsa {1 2 840 10040 4 1}	DSA signature certificate Parameters: p, q, g
		dhpublicnumber {1 2 840 10046 2 1}	DH certificate Parameters: p, g, q
Issuer's Signature	N/A	Same value as in Issuer Signature Algorithm	
Extensions			
Authority Key Identifier	No	Octet String	Same as subject key identifier in issuing CA certificate Prohibited: Issuer DN, Serial Number tuple

Field	Critical	Value	Description
Subject Key Identifier	No	Octet String	Same as in PKCS-10 request or calculated by the issuing CA
Key Usage	Yes	digitalSignature	RSA certificate, DSA certificate, ECDSA certificate
		keyAgreement	ECDH certificate, DH certificate
Extended Key Usage	No	id-kp-clientAuth {1 3 6 1 5 5 7 3 2}	Required
		anyExtendedKeyUsage {2 5 29 37 0}	The anyExtendedKeyUsage OID should be present if the extended key usage extension is included, but there is no intention to limit the types of applications with which the certificate may be used (e.g., the certificate is a general-purpose authentication certificate).
			Prohibited: all others unless consistent with key usage extension
Certificate Policies	No	Per issuer's X.509 certificate policy	
Subject Alternative Name	No	RFC 822 e-mail address, Universal Principal Name (UPN), DNS Name, and/or others	Optional
Authority Information Access	No	id-ad-caIssuers	Required. Access method entry contains HTTP URL for certificates issued to issuing CA
		id-ad-ocsp	Optional. Access method entry contains HTTP URL for the issuing CA OCSP responder
CRL Distribution Points	No	See comments	Optional: HTTP value in distributionPoint field pointing to a full and complete CRL. Prohibited: reasons and cRLIssuer fields, and nameRelativetoCRLIssuer CHOICE

1250

1251 If a client has multiple certificates that meet the requirements of the TLS server, the TLS client
 1252 (e.g., a browser) may ask the user to select from a list of certificates. The extended key usage
 1253 (EKU) extension limits the operations for which the keys in a certificate may be used, and so the
 1254 use of the EKU extension in client certificates may eliminate this request. If the EKU extension
 1255 is included in client certificates, then the id-kp-client-auth key purpose OID **should** be included
 1256 in the certificates to be used for TLS client authentication and **should** be omitted from any other
 1257 certificates.

1258 Client certificates are also filtered by TLS clients on the basis of an ability to build a path to one
 1259 of the trust anchors in the hints list sent by the server, as described in Section 3.5.4.

1260 **4.2.2 Obtaining Revocation Status Information for the Server Certificate**

1261 The client **shall** perform revocation checking of the server certificate. Revocation information
 1262 can be obtained by the client from one of the following locations:

- 1263 1. OCSP response or responses in the server's CertificateStatus message ([30], [51]) (or
 1264 Certificate message in TLS 1.3);

- 1265 2. Certificate Revocation List (CRL) or OCSP response in the client's local certificate store;
- 1266 3. OCSP response from a locally configured OCSP responder;
- 1267 4. OCSP response from the OCSP responder location identified in the OCSP field in the
- 1268 Authority Information Access extension in the server certificate; or
- 1269 5. CRL from the CRL Distribution Point extension in the server certificate.

1270 When the server does not provide the revocation status, the local certificate store does not have
1271 the current or a cogent CRL or OCSP response, and the OCSP responder and the CRL
1272 distribution point are unavailable or inaccessible at the time of TLS session establishment, the
1273 client will either terminate the connection or accept a potentially revoked or compromised
1274 certificate. The decision to accept or reject a certificate in this situation **should** be made
1275 according to agency policy.

1276 Other emerging concepts that can be useful in lieu of revocation checking are further discussed
1277 in Appendix D.2.

1278 4.2.3 Client Public-Key Certificate Assurance

1279 The client public-key certificate may be trusted by the servers on the basis of the policies,
1280 procedures and security controls used to issue the client public-key certificate as described in
1281 Section 3.5.1. For example, these guidelines recommend that the PIV Authentication certificate
1282 be the norm for authentication of Federal employees and long-term contractors. PIV
1283 Authentication certificate policy is defined in the Federal PKI Common Policy Framework [32],
1284 and PIV-I Authentication certificate policy is defined in the X.509 Certificate Policy for the
1285 Federal Bridge Certification Authority [64]. Depending on the requirements of the server-side
1286 application, other certificate policies may also be acceptable. Guidance regarding other
1287 certificate policies is outside the scope of these guidelines.

1288 4.3 Cryptographic Support

1289 4.3.1 Cipher Suites

1290 The acceptable cipher suites for a TLS client are the same as those for a TLS server. General-
1291 purpose cipher suites are listed in Section 3.3.1, and cipher suites appropriate for pre-shared key
1292 environments for TLS 1.2 and prior versions are listed in Appendix C. When ephemeral keys are
1293 used to establish the master secret, each ephemeral key-pair (i.e., the server ephemeral key-pair
1294 and the client ephemeral key-pair) **shall** have at least 112 bits of security.

1295 The client **should not** be configured to use cipher suites other than those listed in Section 3.3.1
1296 or Appendix C.

1297 To mitigate attacks against CBC mode, TLS implementations that support versions prior to TLS
1298 1.3 **shall** use the bad_record_mac error to indicate a padding error. Implementations **shall**
1299 compute the MAC regardless of whether padding errors exist. TLS implementations **should**
1300 support constant-time decryption, or near constant-time decryption. This does not apply to TLS
1301 1.3 implementations, as they do not support cipher suites that use CBC mode.

1302 **4.3.2 Validated Cryptography**

1303 The client **shall** use validated cryptography, as described for the server in Section 3.3.3.

1304 The validated random number generator **shall** be used to generate the random bytes (32 bytes in
1305 TLS 1.3; 28 bytes in prior TLS versions) of the client random value. The validated random
1306 number generator **should** be used to generate the 4-byte timestamp of the client random value for
1307 TLS versions prior to TLS 1.3.

1308 **4.4 TLS Extension Support**

1309 Some servers will refuse the connection if any TLS extensions are included in the ClientHello
1310 message. Interoperability with servers that do not properly handle TLS extensions may require
1311 multiple connection attempts by the client.

1312 **4.4.1 Mandatory TLS Extensions**

1313 The client **shall** be configured to use the following extensions:

- 1314 1. Renegotiation Indication
- 1315 2. Server Name Indication
- 1316 3. Session Hash and Extended Master Secret
- 1317 4. Signature Algorithms
- 1318 5. Certificate Status Request

1319 **4.4.1.1 Renegotiation Indication**

1320 The Renegotiation Indication extension is required by these guidelines as described in Section
1321 3.4.1.1. Clients **shall** perform the initial and subsequent renegotiations in accordance with RFC
1322 5746 [57].

1323 **4.4.1.2 Server Name Indication**

1324 The server name indication extension is described in Section 3.4.1.2. The client **shall** be capable
1325 of including this extension in a ClientHello message, as described in RFC 6066 [30].

1326 **4.4.1.3 Session Hash and Extended Master Secret**

1327 The Session Hash and Extended Master Secret extension, described in Section 3.4.1.3, prevents
1328 man-in-the-middle attacks by binding the master secret to a hashed log of the full handshake.
1329 The client **shall** support this extension.

1330 **4.4.1.4 Signature Algorithms**

1331 The clients **shall** assert acceptable hashing and signature algorithm pairs in this extension in TLS
1332 1.2 and TLS 1.3 ClientHello messages. The extension, its syntax, and processing rules are
1333 described in Sections 7.4.1.4.1, 7.4.4, 7.4.6 and 7.4.8 of RFC 5246 [26] and in Section 4.2.3 of
1334 the TLS 1.3 specification [56]. Note that the extension described in the TLS 1.3 specification
1335 updates the extension described in RFC 5246 by adding an additional signature scheme.

1336 4.4.1.5 Certificate Status Request

1337 The client **shall** include the “status_request” extension in the ClientHello message.

1338 4.4.2 Conditional TLS Extensions

1339 A TLS client supports the following TLS extensions under the circumstances described:

- 1340 1. The Fallback Signaling Cipher Suite Value (SCSV) extension **shall** be supported if the
1341 client supports versions of TLS prior to TLS 1.2 and does not support TLS 1.3.
- 1342 2. The Negotiated Groups extension **shall** be supported if the client supports ephemeral
1343 ECDH cipher suites or if the client supports TLS 1.3.
- 1344 3. The EC Point Format TLS extension **shall** be supported if the client supports EC cipher
1345 suite(s).
- 1346 4. The Multiple Certificate Status extension **should** be enabled if the extension is supported
1347 by the client implementation.
- 1348 5. The Trusted CA Indication extension **should** be supported by clients that run on memory-
1349 constrained devices where only a small number of CA root keys are stored.
- 1350 6. The Encrypt-then-MAC extension **shall** be supported when CBC mode cipher suites are
1351 configured.
- 1352 7. The Truncated HMAC extension may be supported by clients that run on constrained
1353 devices when variable-length padding is not supported.
- 1354 8. The Supported versions, Cookie, and Key Share extensions **shall** be supported by TLS
1355 1.3 clients.
- 1356 9. The Pre-Shared Key extension may be supported by TLS 1.3 clients.
- 1357 10. The Pre-Shared Key Exchange Modes extension **shall** be supported by TLS 1.3 clients
1358 that support the Pre-Shared Key extension.

1359 4.4.2.1 Fallback Signaling Cipher Suite Value (SCSV)

1360 This extension, described in Section 3.4.2.1, provides a mechanism to prevent unintended
1361 protocol downgrades in TLS versions prior to TLS 1.3. Clients signal when a connection is a
1362 fallback, and if the server supports a higher TLS version, the server returns a fatal alert. If the
1363 client does not support TLS 1.3, and is attempting to connect with a TLS version prior to TLS
1364 1.2, the client **shall** include TLS_FALLBACK_SCSV at the end of the cipher suite list in the
1365 ClientHello message.

1366 4.4.2.2 Negotiated Groups

1367 The Negotiated Groups extension (supported_groups) is described in Section 3.4.2.3. Client
1368 implementations **shall** send this extension in TLS 1.3 ClientHello messages and in ClientHello
1369 messages that include ephemeral ECDH cipher suites. When elliptic curve cipher suites are
1370 configured, at least one of the NIST-approved curves, P-256 (secp256r1) and P-384 (secp384r1),
1371 **shall** be supported as described in RFC 4492 [14]. The finite field groups ffdhe2048, ffdhe3072,
1372 ffdhe4096, ffdhe6144, and ffdhe8192 may be supported (see RFC 7919 [35]).

1373 4.4.2.3 Key Share

1374 The Key Share extension is used to send cryptographic parameters. Clients that support TLS 1.3
1375 **shall** support this extension as described in Section 4.2.7 of the TLS 1.3 specification [56].

1376 4.4.2.4 EC Point Format

1377 The clients that support EC cipher suites **shall** be capable of specifying supported EC point
1378 formats in the ClientHello message, in accordance with Section 5.1 of [14].

1379 Clients that support EC cipher suites **shall** support the processing of at least one²⁹ of the EC
1380 point formats received in the ServerHello message, as described in Section 5.2 of [14].

1381 4.4.2.5 Multiple Certificate Status

1382 The multiple certificate status extension is described in Section 3.4.2.6. This extension improves
1383 on the Certificate Status Request extension described in Section 3.4.1.5 by allowing the client to
1384 request the status of all certificates provided by the server in the TLS handshake. This extension
1385 is documented in RFC 6961 [51]. Client implementations that have this capability **should** be
1386 configured to include this extension in the ClientHello message.

1387 4.4.2.6 Trusted CA Indication

1388 Clients that run on memory-constrained devices where only a small number of CA root keys are
1389 stored **should** be capable of including the trusted CA indication (trusted_ca_keys) extension in a
1390 ClientHello message as described in [30].

1391 4.4.2.7 Encrypt-then-MAC

1392 The Encrypt-then-MAC extension, described in Section 3.4.2.2, can mitigate or prevent several
1393 known attacks on CBC cipher suites. In order for this modified order of operations to be applied,
1394 both server and client need to implement the Encrypt-then-MAC extension and negotiate its use.
1395 When CBC mode cipher suites are configured, clients **shall** support this extension as described
1396 in RFC 7366 [36]. The client **shall** include this extension in the ClientHello message whenever
1397 the ClientHello message includes CBC cipher suites.

1398 4.4.2.8 Truncated HMAC

1399 The Truncated HMAC extension is described in Section 3.4.2.8. Clients running on constrained
1400 devices may support this extension. The Truncated HMAC extension **shall not** be used in
1401 conjunction with variable-length padding, due to attacks described by Paterson et al. [50]. This
1402 extension cannot be used with TLS 1.3.

²⁹ The uncompressed point format must be supported, as described in Sections 5.1.2 and 5.2 of [14].

1403 4.4.2.9 Supported Versions

1404 The supported versions extension was added in TLS 1.3. The client sends this extension in the
1405 ClientHello message to indicate which versions of TLS it is able to negotiate. A TLS 1.3 client
1406 **shall** send this extension in the ClientHello message.

1407 4.4.2.10 Cookie

1408 The cookie extension, added in TLS 1.3, allows the server to force the client to prove that it is
1409 reachable at its apparent network address, and offload state to the client. Clients that support TLS
1410 1.3 **shall** support the cookie extension in accordance with the TLS 1.3 specification [56].

1411 4.4.2.11 Pre-shared Key

1412 The Pre-Shared Key extension (`pre_shared_key`), available in TLS 1.3, is used to indicate the
1413 identity of the pre-shared key to be used for PSK key establishment. In TLS 1.3 pre-shared keys
1414 may either be established out-of-band, as in TLS 1.2 and prior versions, or in a previous
1415 connection, in which case they are used for session resumption. Clients that support TLS 1.3 may
1416 be configured to use this extension in order to allow session resumption or to allow the use of
1417 pre-shared keys that are established out-of-band.

1418 4.4.2.12 Pre-Shared Key Exchange Modes

1419 A TLS 1.3 client must send the Pre-Shared Key Exchange Modes extension
1420 (`psk_key_exchange_modes`) if it sends the Pre-Shared Key extension, otherwise the server will
1421 abort the handshake. TLS clients that support TLS 1.3 and the Pre-Shared Key extension **shall**
1422 implement this extension.

1423 4.4.3 Discouraged TLS Extension

1424 The following extensions **should not** be used:

- 1425 1. Client Certificate URL
- 1426 2. Early Data Indication

1427 The reasons for discouraging the use of these extensions can be found in Section 3.4.3.

1428 4.5 Server Authentication

1429 The client **shall** be able to build the certification path for the server certificate presented in the
1430 TLS handshake with at least one of the trust anchors in the client trust store, if an appropriate
1431 trust anchor is present in the store. The client may use all or a subset of the following resources
1432 to build the certification path: local certificate store, certificates received from the server during
1433 the handshake, LDAP, resources declared in CA Repository field of the Subject Information
1434 Access extension in various CA certificates, and resources declared in the CA Issuers field of the
1435 Authority Information Access extension in various certificates.

1436 4.5.1 Path Validation

1437 The client **shall** validate the server certificate in accordance with the certification path validation
1438 rules specified in Section 6 of [20]. The revocation status of each certificate in the certification
1439 path **shall** be checked using Online Certificate Status Protocol (OCSP) or a certificate revocation
1440 list (CRL). OCSP checking **shall** be in compliance with [61]. Revocation information **shall** be
1441 obtained as described in Section 4.2.2.

1442 Not all clients support name constraint checking. The Federal agencies **should** only procure
1443 clients that perform name constraint checking in order to obtain assurance that unauthorized
1444 certificates are properly rejected. As an alternative, the Federal agency may procure clients that
1445 use one or more of the features discussed in Appendix D.1.

1446 The client **shall** terminate the TLS connection if path validation fails.

1447 Federal agencies **shall** only use clients that check that the DNS name or IP address, whichever is
1448 presented in the client TLS request, matches a DNS name or IP address contained in the server
1449 certificate. The client **shall** terminate the TLS connection if the name check fails.

1450 4.5.2 Trust Anchor Store

1451 Having an excessive number of trust anchors installed in the TLS client can increase the chances
1452 for the client to be spoofed. As the number of trust anchors increase, the number of CAs that the
1453 client trusts increases, and the chances that one of these CAs or its registration system or process
1454 will be compromised to issue TLS server certificates also increases.

1455 Clients **shall not** overpopulate their trust stores with various CA certificates that can be verified
1456 via cross-certification. Direct trust of these certificates can expose the clients unduly to a variety
1457 of situations, including but not limited to, revocation or compromise of these trust anchors.
1458 Direct trust also increases the operational and security burden on the clients to promulgate
1459 addition and deletion of trust anchors. Instead, the client **shall** rely on the server overpopulating
1460 or not providing the hints list to mitigate the client certificate selection and path-building
1461 problem as discussed in Section 3.5.4.

1462 4.5.3 Checking the Server Key Size

1463 The only direct mechanism for a client to check if the key size presented in a server public
1464 certificate is acceptable is for the client to examine the server public key in the certificate. An
1465 indirect mechanism is to ensure that the server public-key certificate was issued under a policy
1466 that indicates the minimum cryptographic strength of the signature and hashing algorithms used.
1467 In some cases, this can be done by the client performing certificate policy processing and
1468 checking. However, since many TLS clients cannot be configured to accept or reject certificates
1469 based on the policies under which they were issued, this may require ensuring that the trust
1470 anchor store only contains CAs that issue certificates under acceptable policies. The client **shall**
1471 check the server public key length if the client implementation provides a mechanism to do so.
1472 The client **shall** also check the server public key length if the server uses ephemeral keys for the
1473 creation of the master secret, and the client implementation provides a mechanism to do so.

1474 The length of each write key is determined by the negotiated cipher suite. Restrictions on the
1475 length of the shared session keys can be enforced by configuring the client to only support cipher
1476 suites that meet the key length requirements.

1477 **4.5.4 User Interface**

1478 When the TLS client is a browser, the browser interface can be used to determine if a TLS
1479 session is in effect. The indication that a TLS session is in effect varies by browser. Examples of
1480 indicators include a padlock in the URL bar, the word “secure” preceding the URL, or a different
1481 color for the URL bar. Some clients, such as browsers, may allow further investigation of the
1482 server certificate and negotiated session parameters by clicking on the lock (or other indicator).
1483 Users **should** examine the interface for the presence of the indicator to ensure that the TLS
1484 session is in force and **should** also visually examine web site URLs to ensure that the user
1485 intended to visit the indicated web site. Users **should** be aware that URLs can appear to be
1486 legitimate, but still not be valid. For example, the numeric “1” and the letter “l” appear quite
1487 similar or the same to the human eye.

1488 Client authentication keys may be located outside of the client (e.g., PIV Cards). Users **shall**
1489 follow the policies and procedures for protecting client authentication keys outside of the client.

1490 **4.6 Session Resumption**

1491 Session resumption considerations and server recommendations were given in Section 3.6. There
1492 are no specific recommendations for clients regarding session resumption when using TLS 1.2,
1493 1.1, or 1.0. Clients typically will not know if any anti-replay mechanisms are in place to prevent
1494 replay attacks on 0-RTT data in TLS 1.3. Therefore, clients using TLS 1.3 **should not** send 0-
1495 RTT data.

1496 **4.7 Compression Methods**

1497 The client **shall** follow the same compression recommendations as the server, which are
1498 described in Section 3.7.

1499 **4.8 Operational Considerations**

1500 The client and associated platform **shall** be kept up-to-date in terms of security patches. This is
1501 critical to various aspects of security.

1502 Once the TLS-protected data is received at the client, and decrypted and authenticated by the
1503 TLS layer of the client system, the unencrypted data is available to the applications on the client
1504 platform.

1505 These guidelines do not mitigate the threats against the misuse or exposure of the client
1506 credentials that resides on the client machine. These credentials could contain the private key
1507 used for client authentication or other credentials (e.g., a one-time password (OTP) or user ID
1508 and password) for authenticating to a server-side application.

1509 For these reasons, the use of TLS does not obviate the need for the client to use appropriate

1510 security measures, as described in applicable Federal Information Processing Standards and
1511 NIST Special Publications, to protect computer systems and applications. Users **shall** operate
1512 client systems in accordance with agency and administrator instructions.

1513

1514 **Appendix A—Acronyms**

1515 Selected acronyms and abbreviations used in this paper are defined below.

3DES	Triple Data Encryption Algorithm (TDEA)
AEAD	Authenticated Encryption with Associated Data
AES	Advanced Encryption Standard
CA	Certification Authority
CBC	Cipher Block Chaining
CCM	Counter with CBC-MAC
CRL	Certificate Revocation List
DES	Data Encryption Standard
DH	Diffie-Hellman key exchange
DHE	Ephemeral Diffie-Hellman key exchange
DNS	Domain Name System
DNSSEC	DNS Security Extensions
DSA	Digital Signature Algorithm
DSS	Digital Signature Standard (implies DSA)
EC	Elliptic Curve
ECDHE	Ephemeral Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
FIPS	Federal Information Processing Standard
GCM	Galois Counter Mode
HKDF	HMAC-based Extract-and-Expand Key Derivation Function
HMAC	Keyed-hash Message Authentication Code
IETF	Internet Engineering Task Force
KDF	Key derivation function
MAC	Message Authentication Code
OCSP	Online Certificate Status Protocol
OID	Object Identifier
PIV	Personal Identity Verification
PKI	Public Key Infrastructure
PRF	Pseudo-random Function
PSK	Pre-shared Key

RFC	Request for Comments
SHA	Secure Hash Algorithm
SSL	Secure Sockets Layer
TLS	Transport Layer Security
URL	Uniform Resource Locator

1516

1517 **Appendix B—Interpreting Cipher Suite Names**

1518 TLS cipher suite names consist of a set of mnemonics separated by underscores (i.e., “_”). The
1519 naming convention in TLS 1.3 differs from the convention shared in TLS 1.0, 1.1, and 1.2.
1520 Section B.1 provides guidance for interpreting the names of cipher suites that are recommended
1521 in these guidelines for TLS versions 1.0, 1.1, and 1.2. Section B.2 provides guidance for
1522 interpreting the names of cipher suites for TLS 1.3. In all TLS cipher suites, the first mnemonic
1523 is the protocol name, i.e., “TLS”.

1524 **B.1 Interpreting Cipher Suites Names in TLS 1.0, 1.1, and 1.2**

1525 One or two mnemonics follow the protocol name, indicating the key-exchange algorithm. If
1526 there is only one mnemonic, it must be PSK, based on the recommendations in these guidelines.
1527 The single mnemonic PSK indicates that the premaster secret is established using only
1528 symmetric algorithms with pre-shared keys, as described in RFC 4279 [31]. Pre-shared key
1529 cipher suites that are approved for use with TLS 1.2 are listed in Appendix C. If there are two
1530 mnemonics following the protocol name, the first key exchange mnemonic should be DH,
1531 ECDH, DHE, or ECDHE. When the first key exchange mnemonic is DH or ECDH, it indicates
1532 that the server’s public key in its certificate is for either DH or ECDH key exchange, and the
1533 second mnemonic indicates the signature algorithm that was used by the issuing CA to sign the
1534 server certificate. When the first key exchange mnemonic is DHE or ECDHE, it indicates that
1535 ephemeral DH or ECDH will be used for key exchange, with the second mnemonic indicating
1536 the server signature public key type that will be used to authenticate the server’s ephemeral
1537 public key.³⁰

1538 Next is the word WITH, followed by the mnemonic for the symmetric encryption algorithm and
1539 associated mode of operations.

1540 The last mnemonic is generally the hashing algorithm to be used for HMAC, if applicable.³¹ In
1541 cases where HMAC is not applicable (e.g., AES-GCM), or the cipher suite was defined after the
1542 release of the TLS 1.2 RFC, this mnemonic represents the hashing algorithm used with the PRF.

1543 The following examples illustrate how to interpret the cipher suite names:

- 1544 • TLS_DH_DSS_WITH_AES_256_CBC_SHA256: The server is using a DH certificate. If
1545 the signature algorithms extension is provided by the client, then the certificate is signed
1546 using one of the algorithms specified by the extension. Otherwise, the certificate is signed
1547 using DSA. Once the handshake is completed, the messages are encrypted using AES-
1548 256 in CBC mode. SHA-256 is used for both the PRF and HMAC computations. Cipher

³⁰ In this case, the signature algorithm used by the CA to sign the certificate is not articulated in the cipher suite.

³¹ HMAC is not applicable when the symmetric encryption mode of operation is authenticated encryption. Note that the CCM mode cipher suites do not specify the last mnemonic and require that SHA-256 be used for the PRF.

1549 suites that specify secure hash algorithms other than SHA-1 are not supported prior to
1550 TLS 1.2.

1551 • **TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384**: Ephemeral ECDH is used for
1552 key exchange. The server's ephemeral public key is authenticated using the server's
1553 ECDSA public key. The CA signature algorithm used to certify the server's ECDSA
1554 public key is not specified by the cipher suite. Once the handshake is completed, the
1555 messages are encrypted and authenticated using AES-256 in GCM mode, and SHA-384
1556 is used for the PRF. Since an authenticated encryption mode is used, messages neither
1557 have nor require an HMAC message authentication code.

1558 **B.2 Interpreting Cipher Suites Names in TLS 1.3**

1559 TLS 1.3 cipher suites are formatted differently from those described in the previous section. In
1560 particular, these cipher suites only specify an authenticated encryption algorithm (which provides
1561 confidentiality, integrity, and message authentication) and a hash algorithm for use with the
1562 HKDF. The negotiation of the key exchange method is handled elsewhere in the TLS handshake.

1563 The following examples illustrate how to interpret TLS 1.3 cipher suite names.

- 1564 • **TLS_AES_256_GCM_SHA384**: messages are encrypted and authenticated with AES-
1565 256 in GCM mode, and SHA-384 is used with the HKDF.
- 1566 • **TLS_AES_128_CCM_SHA256**: messages are encrypted and authenticated with AES-
1567 128 in CCM mode, and SHA-256 is used with the HKDF.

1568 Appendix C—Pre-shared Keys

1569 Pre-shared keys (PSK) are symmetric keys that are already in place prior to the initiation of a
 1570 TLS session (e.g., as the result of a manual distribution). The use of PSKs in TLS versions prior
 1571 to TLS 1.3 is described in RFC 4279 [31], RFC 5487 [4], and RFC 5489 [5]. Pre-shared keys are
 1572 used for session resumption in TLS 1.3. In general, pre-shared keys **should not** be used in TLS
 1573 versions prior to TLS 1.3, or for initial session establishment in TLS 1.3. However, the use of
 1574 pre-shared keys may be appropriate for some closed environments that have adequate key
 1575 management support. For example, they might be appropriate for constrained environments with
 1576 limited processing, memory, or power. If PSKs are appropriate and supported, then the following
 1577 additional guidelines **shall** be followed.

1578 Recommended pre-shared key (PSK) cipher suites for TLS 1.2 are listed below. Cipher suites for
 1579 TLS 1.3 (see Section 3.3.1.2) can all be used with pre-shared keys. Pre-shared keys **shall** be
 1580 distributed in a secure manner, such as a secure manual distribution or using a key-establishment
 1581 certificate. These cipher suites employ a pre-shared key for entity authentication (for both the
 1582 server and the client) and may also use ephemeral Diffie-Hellman (DHE) or ephemeral Elliptic
 1583 Curve Diffie-Hellman (ECDHE) algorithms for key establishment. For example, when DHE is
 1584 used, the result of the Diffie-Hellman computation is combined with the pre-shared key and
 1585 other input to determine the premaster secret.

1586 The pre-shared key **shall** have a minimum security strength of 112 bits. Because these cipher
 1587 suites require pre-shared keys, these suites are not generally applicable to common secure web
 1588 site applications and are not expected to be widely supported in TLS clients or TLS servers.
 1589 NIST suggests that these suites be considered for infrastructure applications, particularly if
 1590 frequent authentication of the network entities is required.

1591 Pre-shared key cipher suites may only be used in networks where both the client and server
 1592 belong to an organization. Cipher suites using pre-shared keys **shall not** be used with TLS 1.0 or
 1593 TLS 1.1, and **shall not** be used when a government client or server communicates with non-
 1594 government systems.

1595 TLS 1.2 servers and clients using pre-shared keys may support the following cipher suites:

- 1596 • TLS_DHE_PSK_WITH_AES_128_GCM_SHA256 (0x00, 0xAA)
- 1597 • TLS_DHE_PSK_WITH_AES_256_GCM_SHA384 (0x00, 0xAB)
- 1598 • TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256 (0xC0, 0x37)
- 1599 • TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA384 (0xC0, 0x38)
- 1600 • TLS_DHE_PSK_WITH_AES_128_CCM (0xC0, 0xA6)
- 1601 • TLS_DHE_PSK_WITH_AES_256_CCM (0xC0, 0xA7)
- 1602 • TLS_PSK_DHE_WITH_AES_128_CCM_8 (0xC0, 0xAA)
- 1603 • TLS_PSK_DHE_WITH_AES_256_CCM_8 (0xC0, 0xAB)
- 1604 • TLS_DHE_PSK_WITH_AES_128_CBC_SHA256 (0x00, 0xB2)
- 1605 • TLS_DHE_PSK_WITH_AES_256_CBC_SHA384 (0x00, 0xB3)
- 1606 • TLS_PSK_WITH_AES_128_GCM_SHA256 (0x00, 0xA8)
- 1607 • TLS_PSK_WITH_AES_256_GCM_SHA384 (0x00, 0xA9)
- 1608 • TLS_PSK_WITH_AES_128_CCM (0xC0, 0xA4)

- 1609 • TLS_PSK_WITH_AES_256_CCM (0xC0, 0xA5)
 - 1610 • TLS_PSK_WITH_AES_128_CCM_8 (0xC0, 0xA8)
 - 1611 • TLS_PSK_WITH_AES_256_CCM_8 (0xC0, 0xA9)
 - 1612 • TLS_PSK_WITH_AES_128_CBC_SHA256 (0x00, 0xAE)
 - 1613 • TLS_PSK_WITH_AES_256_CBC_SHA384 (0x00, 0xAF)
 - 1614 • TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA (0xC0, 0x35)
 - 1615 • TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA (0xC0, 0x36)
 - 1616 • TLS_DHE_PSK_WITH_AES_128_CBC_SHA (0x00, 0x90)
 - 1617 • TLS_DHE_PSK_WITH_AES_256_CBC_SHA (0x00, 0x91)
 - 1618 • TLS_PSK_WITH_AES_128_CBC_SHA (0x00, 0x8C)
 - 1619 • TLS_PSK_WITH_AES_256_CBC_SHA (0x00, 0x8D)
- 1620

1621 **Appendix D—Future Capabilities**

1622 This section identifies emerging concepts and capabilities that are applicable to TLS. As these
1623 concepts mature, and commercial products are available to support them, these guidelines will be
1624 revised to provide specific recommendations.

1625 **D.1 U.S. Federal Public Trust PKI**

1626 The Identity, Credential, and Access Management (ICAM) Subcommittee of the Federal CIO
1627 Council’s Information Security and Identity Management Committee is developing a new public
1628 trust root and issuing CA infrastructure to issue TLS server certificates for Federal web services
1629 on the public Internet. The intent is for this new root to be included in all of the commonly used
1630 trust stores so that Federal agencies can obtain their TLS server certificates from this PKI rather
1631 than from commercial CAs. The certificate policy for this PKI is being developed at
1632 <https://devicepki.idmanagement.gov>.

1633 Once this PKI is operational and is included in the commonly used trust stores, Federal agencies
1634 should consider obtaining their TLS server certificates from this PKI.

1635 **D.2 DANE**

1636 DNS-based Authentication of Named Entities (DANE) leverages DNS security extensions
1637 (DNSSEC) to provide mechanisms for securely obtaining information about TLS server
1638 certificates from the DNS. RFC 6698 [38] specifies a resource record that may be made available
1639 in DNS that includes a certificate (or the public key of a certificate), along with an indicator of
1640 how the certificate is to be used. There are four options:

- 1641 1. The DNS record contains an end-entity certificate. In addition to the server public-key
1642 certificate validation as specified in Section 4.5, the client verifies that the TLS server
1643 certificate matches the certificate provided in the DNS records.
- 1644 2. The DNS record contains a domain-issued end-entity certificate.³² The client can use the
1645 certificate if it verifies that the TLS server certificate matches the one provided in the
1646 DNS records (i.e., the client forgoes server public-key certificate validation as specified
1647 in Section 4.5).
- 1648 3. The DNS record contains a CA certificate. In addition to the server public-key certificate
1649 validation as specified in Section 4.5, the client verifies that the certification path for the
1650 TLS server certificate includes the CA certificate provided in the DNS records.
- 1651 4. The DNS record contains a certificate that is to be used as a trust anchor. The client
1652 validates the TLS server certificate as specified in Section 4.5 using the trust anchor
1653 provided in the DNS records instead of the trust anchors in the client’s local trust anchor
1654 store.

³² In this context, a “domain-issued” certificate is one that is issued by the domain name administrator without involving a third-party CA. It corresponds to usage case 3 in Section 2.1.1 of RFC 6698.

1655 In each case, the client verifies the digital signatures on the DNS records in accordance with the
1656 DNSSEC, as described in RFC 4033 [\[3\]](#).

Appendix E—Determining the Need for TLS 1.0 and 1.1

1657
1658 Enabling TLS 1.0 when it is not needed may leave systems and users vulnerable to attacks (such
1659 as the BEAST attack and the Klima attack [63]). However, disabling TLS 1.0 when there is a
1660 need may deny access to users who are unable to install or upgrade to a browser that is capable
1661 of TLS 1.3, 1.2 or 1.1.

1662 The system administrator must consider the benefits and risks of using TLS 1.0, in the context of
1663 applications supported by the server, and decide whether the benefits of using TLS 1.0 outweigh
1664 the risks. This decision should be driven by the service(s) running on the server and the versions
1665 supported by clients accessing the server. Services that do not access high-value information
1666 (such as personally identifiable information or financial data) may benefit from using TLS 1.0 by
1667 increasing accessibility with little increased risk. On the other hand, services that do access high-
1668 value data may increase the likelihood of a breach for relatively little gain in terms of
1669 accessibility. The decision to support TLS 1.0 must be assessed on a case-by-case basis by the
1670 system administrator.

1671 These guidelines do not give specific recommendations on steps that can be taken to make this
1672 determination. There are tools available (such as the Data Analytics Program [71]) that can
1673 provide information to system administrators that can be used to assess the impact of supporting,
1674 or not supporting, TLS 1.0. For example, DAP data on visitor OS and browser versions can help
1675 administrators determine what percentage of visitors to agency websites cannot negotiate TLS
1676 1.2 (or TLS 1.1) by default.

1677 Many products that implement TLS 1.1 also implement TLS 1.2. Because of this, it may be
1678 unnecessary for servers to support TLS 1.1. Administrators can determine whether TLS 1.1 is
1679 needed by assessing whether it must support connections with clients where 1.1 is the highest
1680 TLS version available.

1681 **Appendix F—References**

- 1682 [1] Adams, C., and Lloyd, S., *Understanding PKI: Concepts, Standard, and Deployment*
1683 *Considerations* (Macmillan Technology Publishing. ISBN 1-57870-166-X, 1999)
- 1684 [2] AlFardan, N.J., and Paterson, K.G., *Lucky Thirteen: Breaking the TLS and DTLS Record*
1685 *Protocols*, February 2013, <http://www.isg.rhul.ac.uk/tls/TLStiming.pdf>
- 1686 [3] Arends, R., Austein, R., Larson, M., Massey, D., and Rose, S., *DNS Security Introduction*
1687 *and Requirements*, Internet Engineering Task Force (IETF) Request for Comments (RFC) 4033,
1688 March 2005, <https://doi.org/10.17487/RFC4033>
- 1689 [4] Badra, M., *Pre-Shared Key Cipher Suites for TLS with SHA-256/384 and AES Galois*
1690 *Counter Mode*, Internet Engineering Task Force (IETF) Request for Comments (RFC) 5487,
1691 March 2009, <https://doi.org/10.17487/RFC5487>
- 1692 [5] Badra, M., and Hajjeh, I., *ECDHE_PSK Cipher Suites for Transport Layer Security*
1693 *(TLS)*, Internet Engineering Task Force (IETF) Request for Comments (RFC) 5489, March
1694 2009, <https://doi.org/10.17487/RFC5489>
- 1695 [6] Barker, E., *Recommendation for Key Management Part 1: General*, NIST Special
1696 Publication (SP) 800-57 Part 1 Revision 4, National Institute of Standards and Technology,
1697 Gaithersburg, Maryland, January 2016, <https://doi.org/10.6028/NIST.SP.800-57pt1r4>
- 1698 [7] Barker, E., Chen, L., and Moody, D., *Recommendation for Pair-Wise Key-Establishment*
1699 *Schemes Using Integer Factorization Cryptography*, NIST Special Publication (SP) 800-56B
1700 Revision 1, National Institute of Standards and Technology, Gaithersburg, Maryland September
1701 2014, <https://doi.org/10.6028/NIST.SP.800-56Br1>
- 1702 [8] Barker, E., Chen, L., Roginsky, A., and Smid, M., *Recommendation for Pair-Wise Key*
1703 *Establishment Schemes Using Discrete Logarithm Cryptography*, Special Publication (SP) 800-
1704 56A Revision 2, National Institute of Standards and Technology, Gaithersburg, Maryland, May
1705 2013, <https://doi.org/10.6028/NIST.SP.800-56Ar2>
- 1706 [9] Barker, E., and Kelsey, J., *Recommendation for Random Number Generation Using*
1707 *Deterministic Random Bit Generators*, NIST Special Publication (SP) 800-90A Revision 1,
1708 National Institute of Standards and Technology, Gaithersburg, Maryland June 2015,
1709 <https://doi.org/10.6028/NIST.SP.800-90Ar1>
- 1710 [10] Barker, E., and Roginsky, A., *Transitions: Recommendation for Transitioning the Use of*
1711 *Cryptographic Algorithms and Key Lengths*, NIST Special Publication (SP) 800-131A Revision
1712 1, National Institute of Standards and Technology, Gaithersburg, Maryland November 2015,
1713 <https://doi.org/10.6028/NIST.SP.800-131Ar1>
- 1714 [11] Barker, W.C., and Barker, E., *Recommendation for the Triple Data Encryption Algorithm*
1715 *(TDEA) Block Cipher*, NIST Special Publication (SP) 800-67 Revision 1, National Institute of
1716 Standards and Technology, Gaithersburg, Maryland, January 2012,
1717 <https://doi.org/10.6028/NIST.SP.800-67r1>

- 1718 [12] Be'ery, T., and Shulman, A., *A Perfect CRIME? Only TIME Will Tell*, Blackhat Europe,
1719 2013, <https://media.blackhat.com/eu-13/briefings/Beery/bh-eu-13-a-perfect-crime-beery-wp.pdf>
- 1720 [13] Bhargavan, K., Lavaud, A.D., Fournet, C., Pironi, A., and Strub, P.Y., *Triple*
1721 *Handshakes and Cookie Cutters: Breaking and Fixing Authentication over TLS*, 2014 IEEE
1722 Symposium on Security and Privacy, May 2014, pp. 98-113, <https://doi.org/10.1109/SP.2014.14>
- 1723 [14] Blake-Wilson, S., Bolyard, N., Gupta, V., Hawk, C., and Moeller, B., *Elliptic Curve*
1724 *Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)*, Internet Engineering
1725 Task Force (IETF) Request for Comments (RFC) 4492, May 2006,
1726 <https://doi.org/10.17487/RFC4492>
- 1727 [15] Bradner, S., *Key words for use in RFCs to Indicate Requirement Levels*, Internet
1728 Engineering Task Force (IETF) Request for Comments (RFC) 2119, March 1997,
1729 <https://doi.org/10.17487/RFC2119>
- 1730 [16] CA/Browser Forum, *Baseline Requirements Certificate Policy for the Issuance and*
1731 *Management of Publicly-Trusted Certificates*, Version 1.4.1, September 2016,
1732 <https://cabforum.org/wp-content/uploads/CA-Browser-Forum-BR-1.4.1.pdf>
- 1733 [17] CA/Browser Forum, *Guidelines For The Issuance And Management Of Extended*
1734 *Validation Certificates*, Version 1.6.0, July 2016, [https://cabforum.org/wp-content/uploads/EV-](https://cabforum.org/wp-content/uploads/EV-V1_6_0.pdf)
1735 [V1_6_0.pdf](https://cabforum.org/wp-content/uploads/EV-V1_6_0.pdf)
- 1736 [18] Chernick, C.M., III, C.E., Fanto, M.J., and Rosenthal, R., *Guidelines for the Selection*
1737 *and Use of Transport Layer Security (TLS) Implementations*, NIST Special Publication (SP) 800-
1738 52, National Institute of Standards and Technology, Gaithersburg, Maryland, June 2005,
- 1739 [19] Comer, D.E., *Internetworking with TCP/IP, Principles, Protocols, and Architectures*
1740 (Prentice Hall, fourth edn., ISBN 0-13- 018380-6, 2000)
- 1741 [20] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and Polk, W., *Internet*
1742 *X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*,
1743 Internet Engineering Task Force (IETF) Request for Comments (RFC) 5280, 2008,
1744 <https://doi.org/10.17487/RFC5280>
- 1745 [21] Dang, Q., *Recommendation for Applications Using Approved Hash Algorithms*, NIST
1746 Special Publication (SP) 800-107 Revision 1, National Institute of Standards and Technology,
1747 Gaithersburg, Maryland August 2012, <https://doi.org/10.6028/NIST.SP.800-107r1>
- 1748 [22] Dang, Q., *Recommendation for Existing Application-Specific Key Derivation Functions*,
1749 NIST Special Publication (SP) 800-135 Revision 1, National Institute of Standards and
1750 Technology, Gaithersburg, Maryland, December 2011, [https://doi.org/10.6028/NIST.SP.800-](https://doi.org/10.6028/NIST.SP.800-135r1)
1751 [135r1](https://doi.org/10.6028/NIST.SP.800-135r1)
- 1752 [23] Dang, Q., and Barker, E., *Recommendation for Key Management, Part 3: Application-*
1753 *Specific Key Management Guidance*, NIST Special Publication (SP) 800-57 Part 3 Revision 1,
1754 National Institute of Standards and Technology, Gaithersburg, Maryland, January 2015,

- 1755 <https://doi.org/10.6028/NIST.SP.800-57pt3r1>
- 1756 [24] Dierks, T., and Allen, C., *The TLS Protocol Version 1.0*, Internet Engineering Task Force
1757 (IETF) Request for Comments (RFC) 2246, January 1999, <https://doi.org/10.17487/RFC2246>
- 1758 [25] Dierks, T., and Rescorla, E., *The Transport Layer Security (TLS) Protocol Version 1.1*,
1759 Internet Engineering Task Force (IETF) Request for Comments (RFC) 4346, 2006,
1760 <https://doi.org/10.17487/RFC4346>
- 1761 [26] Dierks, T., and Rescorla, E., *The Transport Layer Security (TLS) Protocol Version 1.2*,
1762 Internet Engineering Task Force (IETF) Request for Comments (RFC) 5246, August 2008,
1763 <https://doi.org/10.17487/RFC5246>
- 1764 [27] Dworkin, M., *Recommendation for Block Cipher Modes of Operation: Galois/Counter
1765 Mode (GCM) and GMAC*, NIST Special Publication (SP) 800-38D, National Institute of
1766 Standards and Technology, Gaithersburg, Maryland, November 2007,
1767 <https://doi.org/10.6028/NIST.SP.800-38D>
- 1768 [28] Dworkin, M., *Recommendation for Block Cipher Modes of Operation: Methods and
1769 Techniques*, NIST Special Publication (SP) 800-38A, National Institute of Standards and
1770 Technology, Gaithersburg, Maryland, December 2001, <https://doi.org/10.6028/NIST.SP.800-38A>
1771
- 1772 [29] Dworkin, M., *Recommendation for Block Cipher Modes of Operation: the CCM Mode
1773 for Authentication and Confidentiality*, NIST Special Publication (SP) 800-38C, National
1774 Institute of Standards and Technology, Gaithersburg, Maryland, May 2004,
1775 <https://doi.org/10.6028/NIST.SP.800-38C>
- 1776 [30] Eastlake, D., 3rd, *Transport Layer Security (TLS) Extensions: Extension Definitions*,
1777 Internet Engineering Task Force (IETF) Request for Comments (RFC) 6066, January 2011,
1778 <https://doi.org/10.17487/RFC6066>
- 1779 [31] Eronen, P., and Tschofenig, H., *Pre-Shared Key Ciphersuites for Transport Layer
1780 Security (TLS)*, Internet Engineering Task Force (IETF) Request for Comments (RFC) 4279,
1781 December 2005, <https://doi.org/10.17487/RFC4279>
- 1782 [32] Federal Public Key Infrastructure Authority, *X.509 Certificate Policy For The U.S.
1783 Federal PKI Common Policy Framework*, Version 1.27, June 2017,
1784 [https://www.idmanagement.gov/wp-content/uploads/sites/1171/uploads/fpki-x509-cert-common-
1785 policy.pdf](https://www.idmanagement.gov/wp-content/uploads/sites/1171/uploads/fpki-x509-cert-common-policy.pdf)
- 1786 [33] Freier, A., Karlton, P., and Kocher, P., *The Secure Sockets Layer (SSL) Protocol Version
1787 3.0*, Internet Engineering Task Force (IETF) Request for Comments (RFC) 6101, August 2011,
1788 <https://doi.org/10.17487/RFC6101>
- 1789 [34] Friend, R., *Transport Layer Security (TLS) Protocol Compression Using Lempel-Ziv-Stac
1790 (LZS)*, Internet Engineering Task Force (IETF) Request for Comments (RFC) 3943, November
1791 2004, <https://doi.org/10.17487/RFC3943>

- 1792 [35] Gillmor, D., *Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for*
1793 *Transport Layer Security (TLS)*, Internet Engineering Task Force (IETF) Request for Comments
1794 (RFC) 7919, August 2016, <https://doi.org/10.17487/RFC7919>
- 1795 [36] Gutmann, P., *Encrypt-then-MAC for Transport Layer Security (TLS) and Datagram*
1796 *Transport Layer Security (DTLS)*, Internet Engineering Task Force (IETF) Request for
1797 Comments (RFC) 7366, September 2014, <https://doi.org/10.17487/RFC7366>
- 1798 [37] Hall, E., *Internet Core Protocols: The Definitive Guide* (O'Reilly & Associates, Inc.,
1799 ISBN 1-56592-572-6, 2000)
- 1800 [38] Hoffman, P., and Schlyter, J., *The DNS-Based Authentication of Named Entities (DANE)*
1801 *Transport Layer Security (TLS) Protocol: TLSA*, Internet Engineering Task Force (IETF)
1802 Request for Comments (RFC) 6698, August 2012, <https://doi.org/10.17487/RFC6698>
- 1803 [39] Hollenbeck, S., *Transport Layer Security Protocol Compression Methods*, Internet
1804 Engineering Task Force (IETF) Request for Comments (RFC) 3749, May 2004,
1805 <https://doi.org/10.17487/RFC3749>
- 1806 [40] Housley, R., and Polk, T., *Planning for PKI, Best Practices Guide for Deploying Public*
1807 *Key Infrastructure* (John Wiley & Sons. ISBN 0-471-39702-4, 2001)
- 1808 [41] Joint Task Force Transformation Initiative, *Security and Privacy Controls for Federal*
1809 *Information Systems and Organizations*, NIST Special Publication (SP) 800-53 Revision 4,
1810 National Institute of Standards and Technology, Gaithersburg, Maryland, April 2013,
1811 <https://doi.org/10.6028/NIST.SP.800-53r4>
- 1812 [42] K. Bhargavan, E., Delignat-Lavaud, A., Pironi, A., Langley, A., and Ray, M., *Transport*
1813 *Layer Security (TLS) Session Hash and Extended Master Secret Extension*, Internet Engineering
1814 Task Force (IETF) Request for Comments (RFC) 7627, September 2015,
1815 <https://doi.org/10.17487/RFC7627>
- 1816 [43] Krawczyk, H., and Eronen, P., *HMAC-based Extract-and-Expand Key Derivation*
1817 *Function (HKDF)*, Internet Engineering Task Force (IETF) Request for Comments (RFC) 5869,
1818 May 2010, <https://doi.org/10.17487/RFC5869>
- 1819 [44] Kuhn, D.R., Hu, V.C., Polk, W.T., and Chang, S.-J., *Introduction to Public Key*
1820 *Technology and the Federal PKI Infrastructure*, NIST Special Publication (SP) 800-32, National
1821 Institute of Standards and Technology, Gaithersburg, Maryland, February 2001,
1822 <https://doi.org/10.6028/NIST.SP.800-32>
- 1823 [45] Langley, A., *The POODLE bites again*,
1824 <https://www.imperialviolet.org/2014/12/08/poodleagain.html>
- 1825 [46] Laurie, B., Langley, A., and Kasper, E., *Certificate Transparency*, Internet Engineering
1826 Task Force (IETF) Request for Comments (RFC) 6962, June 2013,
1827 <https://doi.org/10.17487/RFC6962>

- 1828 [47] McGrew, D., and Bailey, D., *AES-CCM Cipher Suites for Transport Layer Security*
1829 (*TLS*), Internet Engineering Task Force (IETF) Request for Comments (RFC) 6655, July 2012,
1830 <https://doi.org/10.17487/RFC6655>
- 1831 [48] Moeller, B., and Langley, A., *TLS Fallback Signaling Cipher Suite Value (SCSV) for*
1832 *Preventing Protocol Downgrade Attacks*, Internet Engineering Task Force (IETF) Request for
1833 Comments (RFC) 7507, April 2015, <https://doi.org/10.17487/RFC7507>
- 1834 [49] Möller, B., Duong, T., and Kotowicz, K., *This POODLE Bites: Exploiting The SSL 3.0*
1835 *Fallback*, September 2014, <https://www.openssl.org/~bodo/ssl-poodle.pdf>
- 1836 [50] Paterson, K.G., Ristenpart, T., and Shrimpton, T., *Tag size does matter: attacks and*
1837 *proofs for the TLS record protocol*. Proc. 17th international conference on The Theory and
1838 Application of Cryptology and Information Security, Seoul, South Korea, 2011, Proceedings of
1839 the 17th international conference on The Theory and Application of Cryptology and Information
1840 Security, <https://doi.org/10.1007/978-3-642-25385-0>
- 1841 [51] Pettersen, Y., *The Transport Layer Security (TLS) Multiple Certificate Status Request*
1842 *Extension*, Internet Engineering Task Force (IETF) Request for Comments (RFC) 6961, 2013,
1843 <https://doi.org/10.17487/RFC6961>
- 1844 [52] Polk, T., McKay, K., and Chokhani, S., *Guidelines for the Selection, Configuration, and*
1845 *Use of Transport Layer Security (TLS) Implementations*, NIST Special Publication (SP) 800-52
1846 Revision 1, National Institute of Standards and Technology, Gaithersburg, Maryland, April 2014,
1847 <https://doi.org/10.6028/NIST.SP.800-52r1>
- 1848 [53] Polk, W.T., Hastings, N.E., and Malpani, A., *Public key infrastructures that satisfy*
1849 *security goals*, IEEE Internet Computing, 2003, 7, (4), pp. 60-67
- 1850 [54] Rescorla, E., *SSL and TLS: Designing and Building Secure Systems* (Addison-Wesley.
1851 ISBN 0201615983, 2001)
- 1852 [55] Rescorla, E., *TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois*
1853 *Counter Mode (GCM)*, Internet Engineering Task Force (IETF) Request for Comments (RFC)
1854 5289, August 2008, <https://doi.org/10.17487/RFC5289>
- 1855 [56] Rescorla, E., *The Transport Layer Security (TLS) Protocol Version 1.3*, July 2017,
1856 <https://datatracker.ietf.org/doc/draft-ietf-tls-tls13/>
- 1857 [57] Rescorla, E., Ray, M., Dispensa, S., and Oskov, N., *Transport Layer Security (TLS)*
1858 *Renegotiation Indication Extension*, Internet Engineering Task Force (IETF) Request for
1859 Comments (RFC) 5746, February 2010, <https://doi.org/10.17487/RFC5746>
- 1860 [58] Rizzo, J., and Duong, T., *The CRIME Attack*, EKOparty Security Conference, 2012
- 1861 [59] Salowey, J., Choudhury, A., and McGrew, D., *AES Galois Counter Mode (GCM) Cipher*
1862 *Suites for TLS*, Internet Engineering Task Force (IETF) Request for Comments (RFC) 5288,
1863 August 2008, <https://doi.org/10.17487/RFC5288>

- 1864 [60] Salter, M., and Housley, R., *Suite B Profile for Transport Layer Security (TLS)*, Internet
1865 Engineering Task Force (IETF) Request for Comments (RFC) 6460, January 2012,
1866 <https://doi.org/10.17487/RFC6460>
- 1867 [61] Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., and Adams, C., *X.509*
1868 *Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*, Internet
1869 Engineering Task Force (IETF) Request for Comments (RFC) 6960, 2013,
1870 <https://doi.org/10.17487/RFC6960>
- 1871 [62] Seggelmann, R., Tuexen, M., and Williams, M., *Transport Layer Security (TLS) and*
1872 *Datagram Transport Layer Security (DTLS) Heartbeat Extension*, Internet Engineering Task
1873 Force (IETF) Request for Comments (RFC) 6520, February 2012,
1874 <https://doi.org/10.17487/RFC6520>
- 1875 [63] Sheffer, Y., Holz, R., and Saint-Andre, P., *Summarizing Known Attacks on Transport*
1876 *Layer Security (TLS) and Datagram TLS (DTLS)*, Internet Engineering Task Force (IETF)
1877 Request for Comments (RFC) 7457, February 2015, <https://doi.org/10.17487/RFC7457>
- 1878 [64] The Federal Bridge Certification Authority, *X.509 Certificate Policy For The Federal*
1879 *Bridge Certification Authority (FBCA)*, Version 2.31, June 2017,
1880 [https://www.idmanagement.gov/wp-content/uploads/sites/1171/uploads/FBCA-Certificate-](https://www.idmanagement.gov/wp-content/uploads/sites/1171/uploads/FBCA-Certificate-Policy-v2.31-06-29-17.pdf)
1881 [Policy-v2.31-06-29-17.pdf](https://www.idmanagement.gov/wp-content/uploads/sites/1171/uploads/FBCA-Certificate-Policy-v2.31-06-29-17.pdf)
- 1882 [65] U.S. Department of Commerce, *Advanced Encryption Standard*, Federal Information
1883 Processing Standards (FIPS) Publication 197, November 2001,
1884 <https://doi.org/10.6028/NIST.FIPS.197>
- 1885 [66] U.S. Department of Commerce, *Digital Signature Standard (DSS)*, Federal Information
1886 Processing Standards (FIPS) Publication 186-4, July 2013,
1887 <https://doi.org/10.6028/NIST.FIPS.186-4>
- 1888 [67] U.S. Department of Commerce, *The Keyed-Hash Message Authentication Code (HMAC)*,
1889 Federal Information Processing Standards (FIPS) Publication 198-1, July 2008,
1890 <https://doi.org/10.6028/NIST.FIPS.198-1>
- 1891 [68] U.S. Department of Commerce, *Personal Identity Verification (PIV) of Federal*
1892 *Employees and Contractors*, Federal Information Processing Standards (FIPS) Publication 201-
1893 2, August 2013, <https://doi.org/10.6028/NIST.FIPS.201-2>
- 1894 [69] U.S. Department of Commerce, *Secure Hash Standard (SHS)*, Federal Information
1895 Processing Standards (FIPS) Publication 180-4, August 2015,
1896 <https://doi.org/10.6028/NIST.FIPS.180-4>
- 1897 [70] U.S. Department of Commerce, *Security Requirements for Cryptographic Modules*,
1898 Federal Information Processing Standards (FIPS) Publication 140-2, May 2001,
1899 <https://doi.org/10.6028/NIST.FIPS.140-2>
- 1900 [71] U.S. General Services Administration, *DAP: Digital Analytics Program*,

- 1901 <https://www.digitalgov.gov/services/dap/>, [accessed December 5, 2016]
- 1902 [72] US-CERT/NIST, *CVE-2014-0160*, National Vulnerability Database, 2014,
1903 <https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-0160>
- 1904 [73] Yee, P., *Updates to the Internet X.509 Public Key Infrastructure Certificate and*
1905 *Certificate Revocation List (CRL) Profile*, Internet Engineering Task Force (IETF) Request for
1906 Comments (RFC) 6818, January 2013, <https://doi.org/10.17487/RFC6818>
- 1907

1908 **Appendix G—Revision History**

1909 **G.1 Original**

1910 The original version of SP 800-52 was published in June 2005 [18]. At the time, only TLS 1.0
 1911 was final (TLS 1.1 was still under development). TLS 1.1 became a standard in April 2006, and
 1912 TLS 1.2 became a standard in August 2008. SP 800-52 became outdated, and guidance on keys
 1913 and cipher suites was incorporated into SP 800-57 Part 3 [23]. In March 2013, SP 800-52 was
 1914 withdrawn.

1915 **G.2 Revision 1**

1916 The first revision of SP 800-52 was published in April 2014 [52]. The revision was a new
 1917 document that bore little resemblance to the original. At the time, TLS 1.2 was still not prevalent
 1918 and the Federal PKI consisted mainly of RSA certificates. Recommendations were made with
 1919 this in mind so that federal agencies could follow the guidelines with either existing technology
 1920 or technology that was under development. Agencies were advised to develop a plan to migrate
 1921 to TLS 1.2.

1922 After revision 1 was posted, the guidance on keys and cipher suites was removed from SP 800-
 1923 57 Part 3.

1924 **G.3 Revision 2**

1925 Since revision 1, support for TLS 1.2 and cipher suites using ephemeral key exchanges has
 1926 increased, and new attacks have come to light. Revision 2 (this document) requires that TLS 1.2
 1927 be supported, and contains several changes to certificate and cipher suite recommendations.

1928 Revision 2 includes recommendations for TLS 1.3. TLS 1.3 is not yet widely supported, but
 1929 many vendors are working to quickly add support for it to their products. TLS 1.3 offers many
 1930 improvements over previous versions of TLS, so revision 2 advises agencies to develop a plan to
 1931 migrate to TLS 1.3.

1932 Revision 2 also has increased discussion on TLS attacks and guidance on mitigation.

1933 Certificate requirements have also changed in this revision. In particular, status information for
 1934 TLS server certificates is required to be made available via the Online Certificate Status
 1935 Protocol. This revision of the TLS guidelines relaxes requirements on which signature
 1936 algorithms can sign which key types in certificates.