Public Comments on Draft NIST Special Publication 800-52 Revision 2, *Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations* (November 15, 2017)

2/14/2018

NIST received the following public comments on draft Special Publication 800-52 Revision 2, *Guidelines for the Selection, Configuration, and use of Transport Layers Security (TLS) Implementations* (November 2017). These public comments were received by the February 1, 2018 deadline.

From Steve Lewis:

Dear NIST,

In general a good document and some good guidance provided.

The standard is promoting the use of early TLS versions. This conflict other standards, such as PCI DSS, that as withdrawing support for the early versions.
The document tries to mitigate this issue with Appendix E, but lines 1669 and 1670 are incorrect.

The document states 'The decision to support TLS 1.0 must be assessed on a case-by-case basis by the system administrator.'

The document should state that something like, ' Use of TLS 1.0 must be technically assessed and approved by executive management on a case by case basis, to ensure associated risks and potential business implications are known and accepted'

From Elaine Barker:

General:
Suggest using full justification on the paragraphs.

Should there be a compilation of the differences in TLS 1.2 and 1.3?

Specific:

Line 122: Provide a reference and/or a summary of the attacks.

Line 130-131: Is this saying that TLS 1.3 is only supporting RSA, rather than either DH or ECDH?

Line 170: Remove the 2nd occurrence of "to".

Line 292: Either insert "that" after "requires" or "to" after "services".

Line 429: Please change "and the resulting key is used as" to "which are used to compute" (avoiding the use of "key").

Line 481: Add something like "and presented in the certificate during the handshake."

Line 513, Footnote 7: Refer to 800-90 for the RBGs.

Line 538: Insert "the" before "use."

Lines 547-548: Is there a way to find out which servers have incorrectly implemented server negotiation (other than by testing)?

Line 560, footnote 12: The curves are being moved to SP 800-186 during the DSS revision. Should this be mentioned in any way?

Lines 647-648: Is there a list of trusted CA? Also, replace the semicolon with a comma.

Line 652: What does TLS 1.3 do with regard to key exchange algorithms? Shouldn't there be some reference or mention about that?

Line 674: Should there be an explanation about what HKDF is used for?

Line 724: Add "web site" at the end.

Lines 730-741: The parenthetical "terms" need to be explained?

Line 839: Insert "as" after "as well."

Line 860: Remove "encryption" since only "algorithms" is used earlier in the sentence.

Lines 889-890: This seems to be a requirement on the CAVP. This needs to be reworded to indicate that only if there is a successful validation can the implementation be used.

Line 983: Insert "as a" before "means"

Line 986: Change to "This extension in TLS 1.3" for clarity (assuming this is what is meant).

Lines 990-991: This sentence is confusing. Is this saying that even if TLS 1.3 is not supported by the server, the server shall support the extension?

Line 1000: Define group as a set of domain parameters for the algorithm?

Line 1042: Does this mean that the tag can't be truncated in TLS 1.3, or just that the extension is not define/allowed in 1.3?

Line 1078: Insert "information" after "state"?

Line 1081: Make "extension" plural.

Line 1099: Change to "as well as"

Lines 1106-1107: How about something like "...request a client to provide the client certificate in order to cryptographically authenticate that client"

Line 1150: This occurrence of "Federal" may need to be lowercase.

Lines 1432-1434: Insert "the" before "local" and before "resources" (twice)39.

Line 1439: Insert "the" before "Online".

Lines 1442 and 1444" "Federal" should be lowercase.

Line 1439: Insert "the" before "addition".

Line 1489: Whose policies? The organization's?

Lines 1525-1542: This is hard to parse: Consider the following:

"As shown in Section 3.3.1, these cipher suites have the following form:

*TLS_KeyExchangeAlg_*WITH*_EncryptionAlg_MessageAuthenticationAlg*

KeyExchangeAlg consists of one or two mnemonics.

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

Next is the word WITH, followed by *EncryptionAlg*, which is used to indicate the symmetric encryption algorithm and associated mode of operation."

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

Lines 1545-1546: Don't understand this sentence. Do you mean the following? "If the signature algorithms extension is provided in the certificate,..."

Line 1559: Consider inserting the following before the first sentence:

"As shown in Section 3.3.1, TLS 1.3 cipher suites have the form:

TLS*_AEAD_HASH"*

Line 1542: Change to "belong to the same organization"?

Lines 1602 and 1603: I think that "PSK" and "DHE" are in the wrong order.

Lines 1628, 1630 and 1633: "Federal" should be lowercase.

Line 1636: "leverages" looks strange following "Entities". Consider spelling out DANE in the title so that this sentence could just begin with "DANE"?

Line 1678: Do you mean that it would be unnecessary to support TLS 1.1 as a separate/stand-alone thing? Should the preceding sentence be revised to put TLS 1.2 before TLS 1.1?

From Gary Gapinski:

1. This version is commendably less prolix than the prior version, and is improved by stating each normative requirement as few times as possible (ideally, once).
2. Requirements stated using the imperatives "shall" and "shall not" are at odds with https://www.plainlanguage.gov/guidelines/conversational/use-must-to-indicate-requirements/.
3. There are multiple instances (lines 103–104, 142–143, 170, 537–538, and 1231) of statements resembling "… agencies **shall** develop migration plans to support TLS 1.3 by January 1, 2020…". This is ambiguous as worded. It is not obvious as stated if the plans must be completed no later than 2020-01-01 and that TLS 1.3 support may be deferred to a planned time of future convenience, or that TLS 1.3 must be supported no later than 2020-01-01 with an associated migration plan presumably having been developed prior to that. The latter interpretation appears to be best defined in line 538. If that is the correct interpretation, it would be preferable that a companion date be separately declared for migration plan initiation or completion.
4. If TLS 1.3 is to be accommodated no later than 2020-01-01, sources of TLS 1.3 implementations must be available well prior to implementation. At this time, that is by no means assured, nor can a procurement levy be introduced until TLS 1.3 implementations are available, reliable, supported, and validated.
5. Will the introduction of the TLS 1.3 *protocol* affect the status of previously validated cryptographic modules? The question arises at least in part because the related cipher suites are novel even though the underlying algorithms are approved. If CMVP re-validation is triggered, 2020 appears an even more unlikely target.
6. "publicly accessible" (lines 146, 292, and page 8 footnote 8) could be misinterpreted (with respect to citizen- or business-facing in line 539) as simply accessible via the public Internet, even though some systems accessible in that fashion may not be for public use (i.e., are "government-only" services).
7. page iv footnote 1 last sentence typo — plurality mismatch. ("it is" should be "they are" or "either is"; "it is" follows later in sentence and likewise needs number agreement).
8. line 170 typo — "develop migration plans to support **to** TLS 1.3 by 2020".
9. §2.1 line 422 is missing a comma after the parenthetical phrase.
10. §2.7 line 517 has a plurality mismatch likely best solved by changing "which allows" to "allowing".
11. §3.1 does not indicate that server protocol preference order should be 1.3 ≻ 1.2 ≻ 1.1 ≻ 1.0. This is implicit, but perhaps worthy of explication.
12. §3.2 lines 553-556: only the RSA and ECDSA types are likely to be used. Including the others (DSA, DH, ECDH) will cause confusion.
13. §3.2 lines 561–562 erroneously define relying parties in the context of networks.
14. §3.2 lines 563–564 appears to exclude CAs operated by organizations for internal use (and which internal CAs do not provide an OCSP service). Is that what was intended? Lines 1933–1936 state that OCSP is required.

15. §3.2 lines 572–575 mentions IP address in the SAN, which is not allowed by the CA/BF BRs.
16. §3.2 line 573: there must be at least one FQDN in SAN.
17. §3.2.1 line 606:
    - table 3-1 generally has some conflicts with CA/BF BRs.
    - table 3-1 Subject Alternative Name value clashes with CA/BF BRs (IP addresses are not allowed).
    - table 3-1 Subject Distinguished Name value is not relevant for commercial CA issuers. CN is deprecated per CA/BF BRs.
18. §3.3.1 We have had inquiries regarding the availability of ChaCha20 (TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256, TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256, TLS_CHACHA20_POLY1305_SHA256) as an approved algorithm. There is good contemporary support for these (TLS 1.2) cipher suites, and TLS_CHACHA20_POLY1305_SHA256 is a "SHOULD" in TLS 1.3. Is it reasonable to expect that it will become an approved algorithm between now and January 1, 2020? Or should systems with constrained resources expect to use the CCM cipher suites (which lack good contemporary support at least in browsers)?
19. §3.3.1 contains cipher suites which appear in the RFC 7540 blacklist.
    - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
    - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
    - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
    - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
    - TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
    - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
    - TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
    - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
    - TLS_DHE_RSA_WITH_AES_128_CBC_SHA
    - TLS_DHE_RSA_WITH_AES_256_CBC_SHA
    - TLS_DHE_DSS_WITH_AES_128_CBC_SHA256
    - TLS_DHE_DSS_WITH_AES_256_CBC_SHA256
    - TLS_DHE_DSS_WITH_AES_128_CBC_SHA
    - TLS_DHE_DSS_WITH_AES_256_CBC_SHA
    - all the DH certificate cipher suites
    - all the ECDH certificate cipher suites
20. §3.3.1.1 lines 713–719 are quite useful but perhaps some (non-normative) mention of the relative efficiency of cipher suites with or without hardware assist could aid implementor's choices.
21. §3.3.1.1.1 and §3.3.1.1.2: contemporary browsers do not offer CCM cipher suites (nor any of the §3.3.1.1.[345] ones). Is the inclusion of CCM to future-proof server cipher suite offerings, or just there for those who need to use them?

22. §3.3.1.1.5 line 826 typo — in suite identifier (**013x**, 0x01).
23. line 1081 typo (pluralize extension).

From Andreas Bartelt:

Hello,

I've written a Bosch-internal guideline on the secure use of TLS 1.2 in the past, and I'm currently working on a guideline for TLS interoperability use cases.

After reading through your draft document of SP 800-52 rev. 2, which is already in pretty good shape, I'd like to provide the following suggestions for further improvement:

- lines 122ff: it should be noted that this kind of vulnerability (e.g., ROBOT) can be effectively mitigated via careful implementation at the server side (i.e., this is not strictly a protocol-level vulnerability). Consequently, a corresponding implementation requirement could be formulated in order to counter the problem. Reason for this suggestion: RSA key transport might still be the "least worst option" for some interoperability use cases (e.g., in case TLS clients can only handle DHE parameters smaller than 2048 bits). I'm also aware of some popular load balancer appliances from F5 which can only efficiently handle 1024-bit DHE parameters via HW acceleration (i.e., RSA key transport might also be the better fallback option in this case). AFAIK, some large companies such as Google deploy the same strategy for TLS interoperability use cases (i.e., they have eliminated DHE cipher suites as a fallback in favor of RSA key transport since this variant typically works in a more robust fashion). The drawback that RSA key transport doesn't provide forward secrecy might be acceptable for some use cases.
- lines 327/328 "The list of extensions that can be used with TLS 1.3 has been reduced considerably." Comment: I think this statement is not precise. TLS 1.3 actually uses a lot of extensions. However, many of them are disjunct from the TLS extensions from previous versions -- even if they implement the same or a very similar mechanism. Statement should probably be: "The list of extensions from previous TLS versions that can be used with TLS 1.3 has been reduced considerably."

Minimum TLS Server Requirements

- lines 606ff / table 3-1 and lines 1249 / table 4-1: Serial number shall be unique. Comment: after collision attacks against MD5 had been practically demonstrated against a public CA in the past, a common practical mitigation was to randomize the serial number of certificates. This common practice is still kept alive in CAB forum's Baseline Requirements and public CAs follow this practice. Consequently, wouldn't it be better to require the serial number to be unique as well as random with 128 bits of entropy?
- lines 606ff / table 3-1: I'd suggest to add the "Must Staple" X.509v3 extension (RFC 7633) [possibly even as a SHALL requirement] in order to be able to signal to clients that

the server side is capable of and expected to provide a stapled OCSP response. AFAIK, only the combination of the OCSP stapling TLS extension with the Must Staple X.509v3 certificate extension renders OCSP checks reliable (i.e., this approach provides a reasonably good reason for a "hard fail" policy decision in case a stapled OCSP response is not provided by the server then).

- With regard to the out-of-band usage of OCSP, the (optional) use of OCSP nonces should be discussed in the guideline. In the context of OCSP stapling, I guess the use of OCSP nonces wouldn't make much sense which could also be mentioned.
- In the case that the OCSP Status Request extension from RFC 6066 is used ("status_request"), it should be made more clear in the guideline that checking the revocation status of intermediate CAs SHALL (additionally) be done by some other means (i.e., implementation / deployment requirement which is possibly not in direct scope of the TLS implementation).
- lines 677-682: Many TLS libraries as well as server-side TLS-enabled applications (e.g., apache, nginx) provide an option to enforce the server-side cipher suites preference. I'd suggest to mention this, make this an implementation requirement and hint that server-side preference should [or shall?] always be enforced in interoperability use cases.

cipher suites

- lines 715-719, sections 3.3.1.1.1 & .2, 3.3.1.2, Appendix C: CCM_8 cipher suites are listed and even preferred over non-truncated 128/256-bit CBC/HMAC-based cipher suites; in my opinion, CCM_8 cipher suites almost never provide convincing trade-offs since they only save 8 bytes from each TLS record during bulk data transfer; however, at the same time, the security strength of the authenticator is reduced to 64 bits; even if only "online" attacks by brute force guessing are feasible without further knowledge about the session key, I wouldn't consider this to be a good trade-off for almost all use cases since the additional record layer overhead of a full 128 bit authenticator (i.e., 8 bytes more than for CCM_8) almost never matters in practice – even for deeply embedded use cases which might involve small link layer MTUs. [Slightly related to this, I'd consider RFC 7925 to be quite problematic for the general scope of "IoT" since it trades security for (minimal) bandwidth savings without providing a convincing rationale why / when this kind of trade-off would strictly be required; this is the only RFC / (D)TLS profile I'm aware of which mandates the use of CCM_8 cipher suites] Suggestion: eliminate CCM_8 cipher suites from guideline, or, alternatively, only discuss them as a possible trade-off in the context of deeply embedded use cases which have to cope with a very small link layer MTU.
- chacha20-poly1305 based cipher suites are not listed in the guideline; since they typically provide good trade-offs for use cases where no dedicated crypto HW acceleration is available, they should be included [I'm aware that this is not yet among NIST-approved primitives / schemes; I think this should be changed]
- For TLS 1.2 and below, I think it would make sense to generally prefer ECDHE based cipher suites over DHE based cipher suites (cipher suite order), or even to eliminate DHE from the recommendation for TLS below 1.3. Reason: DHE based key exchange doesn't enable negotiation of DHE modulus size at protocol level during the handshake in the original RFC standard descriptions for TLS 1.2 and below. An alternative solution would be an implementation requirement that the "supported groups" TLS extension from RFC 7919 SHALL be supported by all TLS implementations for all TLS versions [see line

1008]. However, this wouldn't help for interoperability use cases with 3rd party clients which typically don't support this extension. [see also above the rationale for keeping RSA key transport around as a better fallback option for some interoperability use cases]

- general remark: the guideline doesn't provide a detailed discussion on DHE parameter strength [there's just a generic requirement with regard to security level in lines 885/886]. In particular, it should be discussed how a security level at >=112 bits can be ensured via DHE in practice (see above point for more discussion); alternative solution: remove all DHE based cipher suites from recommendation for TLS <1.3.
- Nonce construction for AEAD cipher suites. For AEAD cipher suites in general, deterministic nonce construction is only mandatory in TLS 1.3. With TLS 1.2, deterministic nonce construction is not mandated for all AEAD cipher suites (e.g., for AES-GCM cipher suites, random nonce construction would also be allowed in TLS 1.2 but not in TLS 1.3). The use of random nonce construction provides increased risk of collisions due to the birthday paradox (i.e., this would render the requirements for regularly refreshing session key material much stronger). Implementation requirement: TLS 1.2 implementations SHALL always make use of the sequence number for (deterministic) AEAD nonce construction as it's defined for TLS 1.3.

TLS extensions:

- The shorter overview lists of TLS extensions in sections 3.4.2 and 4.4.2 are not fully consistent with their more detailed discussions which then follow and should be kept in sync [e.g., with regard to their order]. In the following, I'll discuss only the contents of the corresponding subsections which provide the more detailed descriptions.
- sections 3.4.1 and 4.4.1 Renegotiation Indication. Comment: it should be mentioned that this extension is only applicable to TLS versions below 1.3. The guideline also doesn't provide a discussion on support of the actual renegotiation mechanisms yet [required for refreshing session key material]. For TLS <1.3, renegotiation is handled via renegotiation message and for TLS 1.3 via KeyUpdate message (for refreshing session keys) and PSK-based resumption (for resumption of previous TLS connections). From a security perspective, it wouldn't be sufficient to trigger renegotiation/KeyUpdate only when the sequence counter wraps [SHALL requirement] since the maximum (session) key lifetime is typically reached much earlier. The TLS 1.3 draft only provides a weak (SHOULD requirement in section 5.5 Limits on Key Usage) requirement in this context and TLS <1.3 RFCs don't provide any discussion of this at all. I guess the guideline should make this a SHALL requirement. Maybe (automatically triggered) renewal of session key material should even be an implementation requirement (i.e., TLS libraries should respect a suitable maximum key lifetime of any negotiated cipher suite automatically). Moreover, there should probably be a discussion if the use of HTTP/2 with TLS 1.2 is acceptable (see RFC 7540 – this TLS profile forbids TLS renegotiation in case HTTP/2 is used with TLS 1.2; consequently, there is no way for refreshing session key material besides terminating / resuming the session). Also related to renegotiation: client-initiated renegotiation might result in some potential DoS scenarios which should be discussed in the guideline (allowing it or not should probably be a policy decision depending on requirements of the use case).
- sections 3.4.1.2 and 4.4.1.2. Is this a SHALL requirement because it's mandated in TLS 1.3 draft or because of RFC 7525? Maybe the guideline should mention that SNI is mainly a functional requirement and not strictly a security requirement. The guideline

currently doesn’t provide a discussion of an SNI related policy in the context of session resumption [resumption is effectively possible via one or more mechanisms in all TLS versions – i.e., session resumption / session ticket for TLS <=1.2 and PSK based resumption for TLS 1.3]. From a security standpoint, I think it would make sense to require for all TLS versions (SHALL requirement) that resumption is only allowed for the same SNI [the TLS 1.3 draft doesn’t seem to strictly mandate that resumption is not conducted across SNIs]. Moreover, should there be a SHALL requirement that resumption is not allowed after fatal alerts [TLS 1.3 draft seems to allow this since “it matches current practice”]?

- sections 3.4.1.3 and 4.4.1.3 Session Hash and Extended Master Secret. Comment: I think shortening this to “Extended Master Secret” would be sufficient [IANA and the TLS 1.3 draft also refer to this extension by the shorter name]. The guideline should mention that this TLS extension only applies to TLS versions below 1.3.
- lines 956, 991, 1340, 1359 Fallback SCSV: the word “extension” should probably be removed here since this mechanism is technically not implemented as a TLS extension but as a special cipher suite value.
- sections 3.4.2.3 and 4.4.2.2: “Negotiated Groups” à would “Supported Groups” still be better? [also see used name from IANA] I’m slightly confused that the TLS 1.3 draft uses the term “negotiated groups” (which is also less precise with regard to its actual function). The guideline should mention that semantics of this extension are different for TLS 1.3 [TLS 1.3 scope: (EC)DHE] and versions below [scope: (EC)DHE and allowed curves for ECDSA signatures]. In case DHE based cipher suites are kept in the guideline’s recommendations, I’d propose to change the MAY requirement with regard to ffdhe into a SHOULD or even a SHALL requirement [see discussion from above].
- sections 3.4.2.5 and [discussion missing in section on client extensions]: the guideline should mention that the EC point format extension is only applicable to TLS versions below 1.3.
- sections 3.4.2.6 and 4.4.2.4. “Multiple Certificate Status” à “Multiple Certificate Status Request”. Discussion should mention that this TLS extension is only applicable to TLS versions below 1.3. TLS 1.3 also enables checking the status of intermediate CAs via stapled OCSP by allowing to include the stapled OCSP information with each certificate. Maybe there should be a corresponding implementation requirement in order to make sure that OCSP information for intermediate CAs is actually provided in case TLS 1.3 is used.
- sections 3.4.2.8 and 4.4.2.8 “Truncated HMAC”. See above discussion on CCM_8. If at all, should only be discussed as a trade-off in the specific context of seriously constrained endpoints. With regard to the general use case, I’d move this to “Discouraged TLS Extensions”.
- sections 3.4.2.9 and 4.3.2.11 Pre-shared Key [check consistency in 4.3.2.11 heading with regard to capital “-S” used throughout the rest of the document]. Since this extension is part of the resumption mechanism from TLS 1.3 [formerly called “session resumption”], the guideline should discuss possible security ramifications of using this mechanism. In case a resumption PSK gets established during the TLS 1.3 handshake, it will then be used for purposes of endpoint authentication (i.e., instead of asymmetric endpoint authentication via certificates) during future resumptions. In other words, TLS 1.3 uses dynamically established symmetric keys for endpoint authentication during session

resumption which might render the endpoint authentication during resumption weaker under specific circumstances. For example, such a resumption PSK might not be as well protected as the ("long-term") private key which corresponds to the server's certificate (e.g., the private key might be securely stored inside an HSM while a resumption PSK might potentially be stored somewhere in the TLS library's memory without any specific protection). A compromised resumption PSK enables a man-in-the-middle attack during the resumption handshake. Implementation / deployment requirement: all resumption PSKs SHALL be stored as securely as the private key which was used for initial endpoint authentication during the TLS handshake. The guideline should probably also add more discussion on how the ticket mechanisms from TLS 1.3 should be used (e.g., making use of single-use tickets vs. reusing tickets which might also result in specific implementation requirements).

- sections 3.4.2.10 and 4.3.2.12 Pre-Shared Key Exchange Modes. The guideline should discuss the respective trade-offs of different PSK exchange modes (i.e., psk_ke vs. psk_dhe_ke). It should be noted that TLS 1.3 makes use of a slightly different definition of "forward secrecy" than is typically used. The term forward secrecy usually refers to independence of session keys from long-term keys [which are typically the private keys from certificates]. With this definition, the forward secrecy property is already achieved after the initial TLS 1.3 handshake (i.e., even psk_ke based resumption would still be forward secret under this definition since the resumption PSK as well as new session keys are already independent from the long-term keys). However, in TLS 1.3, forward secrecy actually refers to independence of session keys with regard to each TLS connection (i.e., psk_dhe_ke makes use of (EC)DHE and refers to forward secrecy relative to the resumption PSK). As a result, I personally don't have a strong preference for psk_dhe_ke but the trade-offs may also depend on the way how the ticket mechanism is used. From a security standpoint, I guess the most conservative approach still would be not to allow resumption at all. Further remark: from a security standpoint, I'd prefer to let the server be in full control which PSK exchange mode will be used during a resumption handshake. However, according to my understanding of the current TLS 1.3 draft (which might be wrong), the PSK exchange modes which are allowed for resumption are eventually signaled by the client side (i.e., the server can then only disallow PSK resumption but has no further control to enforce a specific PSK resumption variant).
- sections 3.4.2.7 and 4.4.2.6 Trusted CA Indication. Discussion should mention that this TLS extension is only applicable to TLS versions below 1.3. TLS 1.3 uses its own Certificate Authorities (certificate_authorities) extension for indicating acceptable certificate authorities.
- The guideline already discusses some TLS extensions which are mainly targeted at constrained endpoints (e.g., Trusted CA Indication, Truncated HMAC, CCM_8 etc). However, the list of TLS extensions in this specific context is not complete yet. Suggestion: remove constrained endpoints and the corresponding TLS extensions from the scope of the guideline. Alternatively, also discuss the remaining TLS extensions [e.g., Maximum Fragment Length Negotiation (max_fragment_length) from RFC 6066, Cached Information Extension (cached_info) from RFC 7924, Raw Public Key from RFC 7250 etc.] in the "Conditional TLS Extensions" sections, or (probably better) in a separate section which is dedicated to constrained endpoints.

- In addition to the “signature_algorithms” extension, TLS 1.3 also has an “signature_algorithms_cert” extension which is not discussed yet in the guideline (supporting it is a SHOULD requirement in the TLS 1.3 draft). The “signature_algorithms_cert” extension should probably be discussed in the “Conditional TLS Extensions” section.
- The guideline doesn’t yet discuss the use of session tickets (TLS Session Resumption without Server-Side State extension from RFC 5077 / IANA SessionTicket TLS) in TLS 1.2 and below. This TLS extension is widely used on the Internet (e.g., most browsers and also many TLS libraries enable it by default). In case the server-side “ticket encryption key” is not rotated frequently [most TLS libraries / TLS-enabled servers don’t rotate multiple ticket encryption keys by default], the confidentiality of the ticket encryption key becomes a single point of failure with regard to the confidentiality of all TLS sessions (i.e., a passive observer on the Internet could decrypt all TLS sessions in case the corresponding ticket encryption key gets leaked by some means (e.g., via heartbleed, ticketbleed, spectre, meltdown or other kinds of information leakage; forward secrecy from (EC)DHE doesn’t help in this case). The guideline should discuss this TLS extension and provide guidance (e.g., SHALL not be used at all, or only be used in case the server side implements an appropriate ticket encryption key management strategy in order to reduce the associated risks [implementation requirement]).
- Similarly, with regard to (stateful) session resumption mechanism from TLS 1.2 and below, there should be an implementation requirement that old (i.e., expired) sessions can be flushed from the cache via API call (or that this happens even automatically), and that the session cache size is configurable.
- The ALPN extension from RFC 7301 is not yet discussed in the guideline. Although this extension probably doesn’t directly affect security in many cases, there are some variants which might affect security (e.g., negotiation of HTTP/2 via ALPN which then forbids renegotiation in the case of TLS 1.2 – see discussion from above).
- The Heartbeat extension from RFC 7520 is only very shortly mentioned in the guideline. Since there are valid use cases for this extension in combination with DTLS (but typically not with TLS), it might be a good idea to further discuss this extension (also see comment below with regard to including a discussion of DTLS into the guideline).
- There is no discussion of the TLS False Start mechanism for TLS 1.2 and below (see RFC 7918) in the guideline. I guess a conservative recommendation would be to disallow it [implementation requirement].
- With regard to client-authentication in TLS 1.3, the guideline does not yet provide a discussion on the use of the OID Filters “oid_filters” extension.
- With regard to client-authentication in TLS 1.3, the guideline does not yet provide a discussion on the use of the Post-Handshake Client Authentication “post_handshake_auth” extension.

Other observations and remarks:

- The discussion of DTLS is currently not in scope of the guideline. Comment: since securing DTLS is very similar to TLS with only very few distinctive mechanisms (e.g., path MTU discovery via heartbeat extension), it might be a good idea to include DTLS security into the scope of the guideline (e.g., in a dedicated section). In case DTLS would be discussed in a separate guideline, this would necessarily require a lot of content duplication between these guidelines.

- I'd propose that the use of x25519 (and possibly x448) should also be allowed for ECDHE for all TLS versions (via supported_groups extension) [I'm aware that these are not yet among NIST-approved primitives / schemes; I think this should be changed].
- I'd propose that the use of ed25519 (and possibly ed448) should also be allowed for signatures in TLS 1.3 (via signature_algorithms and signature_algorithms_cert extensions) [I'm aware that this is not yet among NIST-approved primitives / schemes; I think this should be changed].
- Moreover, in the context RSA-based signatures for TLS 1.3, shouldn't there be a discussion / recommendation on RSA-PSS based signatures as a better replacement for RSA-PKCS1-v1_5 signatures in certificates [signature_algorithms_cert] as well as during the handshake [signature_algorithms]?
- section 2 in the current TLS 1.3 draft states: "Note that while the server may send application data prior to receiving the client's Authentication messages, any data sent at that point is, of course, being sent to an unauthenticated peer." Remark: might this be a problem with regard to security? Should there be a corresponding implementation requirement for TLS 1.3 server implementations not to do this?
- I suppose there should be an implementation requirement that the maximum verification depth of certificate chains is configurable in the TLS implementation. The guideline should probably also define a policy limit with regard to the maximally allowed certification path length in case this has not already been defined somewhere else.

Further Thoughts:
After reading through the current TLS 1.3 draft, I've decided not to include this protocol version into my upcoming TLS guidelines at Bosch. From a security perspective, TLS 1.3 provides only minor improvements over TLS 1.2 which could typically already be achieved via suitable TLS 1.2 configurations / implementations / deployments. TLS 1.3 looks to me like a whole new layer of unwanted protocol complexities and extensions which largely adds to the attack surface of existing TLS implementations (i.e., a lot of new code is required for adding TLS 1.3 support including its extensions). I also have the impression that TLS 1.3 is only barely interoperable with previous TLS versions. A lot of features from previous versions are now solved by different mechanisms in TLS 1.3, resulting in even more bloated TLS implementations in the case of interoperability use cases. The TLS 1.3 standard also has to account for buggy implementations of TLS 1.2 by introducing ugly hacks to the protocol which are bound to stay there forever (e.g., the "supported_versions" extension and legacy fields which require magic values etc). As a result, for TLS interoperability use cases, adding TLS 1.3 on top of the existing protocols doesn't look like a good trade-off to me. TLS 1.3 also makes a lot of configuration choices even more orthogonal than before (configurability of symmetric crypto + hash, signature algorithms for cert / protocol, (EC)DHE parameters, PSK handshakes etc) which introduces even more degrees of freedom which are also directly exposed to administrators who have to understand and configure all of this stuff. In my opinion, TLS 1.3 mainly introduces further complexities into a protocol which already was far too complex.

I think the current state of crypto and for end-to-end securing TCP/IP based communication between applications across the Internet looks about like this:

- cryptographic primitives: we have reasonably good assurance thanks to NIST standards
- cryptographic schemes: we have reasonably good assurance thanks to NIST standards

- “real-world” TCP/IP enabled cryptographic protocols: we have to use the overly complex TLS standard from IETF which has no strong formal basis and multiple security vulnerabilities at protocol level have been identified the past; there is also a severe lack of standardized alternatives which are suitable for “real-world” scenarios
- implementations of TLS: are prone to vulnerabilities which is promoted by its huge complexity
- wouldn’t it be a good time for a NIST competition regarding cryptographic protocols which would be well-suited for the TCP/IP-related use cases from above? My hope is that a resulting protocol standard could also pave the way to simpler, more secure implementations in the future. What do you think? Do you also see the need for having a TLS replacement in the future?

Thank you very much for making these kinds of guidelines available to the public!

From NSA:

1. page iii, line 122-123: Please verify the statement about PKCS#1 v1.5. It is no longer 'disallowed' specifically. (this is a request for accuracy)
2. page iii, lines 127-136: I believe the use of static DH or ECDH is possible for TLS1.3. It is not documented in the standard, but there is nothing in the protocol that would block it. Second, the use of ECDH is actually quite prevalent these days, at least in the non-Government space. And lastly, it is very unlikely that the extension being proposed will be adopted by the IETF. At best it will be an independent submission. Network monitoring via this mechanism should be a last resort.
3. page iii, lines 142-143, page iv, lines 170-171: There should be absolutely no plan to deprecate TLS1.2. In fact in the case where static RSA is required, those use cases should stay on TLS1.2. I do agree that TLS 1.3 should be supported, but don't give the impression that TLS1.2 is being deprecated by saying 'migration plan'. Or add a clarifying sentence/phrase stating that TLS1.2 will be acceptable in the long term.
4. Section 1.1, lines 278-284, and lines 292-294: Combine these paragraphs.
5. Section 1.1, lines 289-291: We would prefer if the use of TLS VPNs wasn't highlighted. There are no standards that guide the use of TLS as a VPN, and current implementations are all over the map wrt security.
6. Section 1.2, lines 323-328: Probably also worth mentioning that the cipher suites are now a selection menu like IPSec. As for the list of extensions... this will change as the protocol ages, we suggest removing that sentence.
7. Section 1.3, lines 354-356: Change to, 'For example, Datagram TLS (DTLS), which runs over UDP, is outside...'. Perhaps add the link to the reference in Appendix F?
8. Section 2.1, line 387: If all three non-record protocols are discussed in this section, why is it titled 'Handshake Protocol'? Because the record is sort of mentioned, I would just change it to 'the TLS Protocols' with a consideration given to having subsections - handshake, alert, change cipher spec, and record.

9. Section 2.1, line 395-396: We would recommend either moving this sentence up to where the handshake is addressed, or deleting it. It appears to be tacked on in the current draft.
10. Section 2.1, line 421: In TLS1.3, is it possible to compress alerts? I believe the compression options have been removed from at least part of the protocol.
11. Section 2.1, change cipher spec: I would think that it would be wise to at least introduce this protocol. Pre-TLS1.3, client authentication is commonly accomplished using change cipher spec vice establishing a new connection (a server auth splash screen is usually first with an acknowledgement by the user).
12. Section 2.2, line 432: We need to be careful. In TLS1.3 the master secret is derived from the handshake master secret which is derived using exchanged credentials, just as in the pre-TLS1.3 protocols. We need to be careful to explain that 0RTT is not allowed w/in Government systems. While that isn't going to be done in this section, care needs to be taken to ensure a misconception doesn't occur here. It might also be nice to point out that part of the handshake is now encrypted in TLS1.3.
13. Section 2.4, lines 451-460: This functionality is more complicated, perhaps reference the section that outlines the extension which allows for the 'encrypt then MAC' option.
14. Section 2.5, lines 474-481: If a server has an RSA key/cert that is used for both key establishment and digital signature, then this description is fine. If a server has an ECDSA key/cert then it doesn't work. Please remove the last sentence.
15. Section 2.6, lines 490-491: the parenthetical comment is distracting. In addition, the goal will be to discourage the use of 0RTT, so the footnote should also be deleted.
16. Section 3.1, line 537-538: TLS1.3 is not expected to supersede TLS1.2. There are situations where TLS1.2 will be used for a long time. Any situation that requires static RSA, will require the use of TLS1.2. Please be clear in how this is stated in the first paragraph.
17. Section 3.1, lines 541, Section 4.1: I understanding allowing TLS1.1, but what is the rationale for allowing TLS 1.0? The last sentence should be SHALL NOT allow the use of SSL2.0, SSL3.0 or TLS1.0.
18. Section 3.2, lines 553-556, Section 4.2.1, lines 1243-1246, and various other sections: I see the need for RSA and ECDSA certs. We should be deprecating DSS, frankly. Neither DH or ECDH need certificates, those exchanges should be ephemeral... If you really think someone has any of these, Todd Johnson (Treasury) should be able to confirm/deny that suspicion for you.
19. Section 3.2, line 559: Please replace 'Suite B' with 'CNSA' (and add the reference for CNSSP 15 - the document that specifies the CNSA suite). Or if you want to wait, we are making all the Suite B RFCs historic, eventually there will be a CNSA for TLS RFC, but it might be a while.
20. Section 3.2, line 561: 'accessible to systems residing on a different network' is an odd turn of phrase. Do we really want to encourage self-signed certificate on TLS servers? I would merely say, 'TLS servers shall be configured with certificates issued by a CA'.

21. Section 3.2, lines 563-565: Wouldn't it be simpler to just say that a server shall use a CA that publishes both CRLs and OCSP responses?
22. Section 3.2, lines 570-572: What document are those section references from?
23. Section 3.2, lines 578-579: Either expand on 'verifying the entity's details', or remove the phrase entirely (ending the sentence with 'further vetting.').
24. Table 3-1, Subject DN, and Section 4.5.1 lines 1447-1449: Remove 'host IP address' as an option.
25. Table 3-1 and Table 4-1, Subject Public Key: For rsaEncryption, make it clear that 2048-bit RSA key modulus or larger is acceptable. Not that I want you to add it, but if DH public keys are here, shouldn't ECDH public keys also be here?
26. Section 3.2.2, lines 628-630: Recommend removing the sentence about EV.
27. Section 3.2.2, lines 637-644: This paragraph is very circular. Recommend deleting the entire paragraph or at least delete all but the first sentence.
28. Section 3.3, lines 651-652: TLS1.3 also specifies cipher suites for key exchange. What is the reason for the parenthetical statement?
29. Section 3.3.1, line 670-671: These are symmetric cipher suites for TLS1.3, supported groups (key exchange) and signature suites are specified separately. [it is a selection menu style cipher suite selection now]
30. Section 3.3.1.1.2, line 756: Typo? should be SHA384?
31. Section 3.3.1.1.3, 3.3.1.1.4, and 3.3.1.1.5: We believe these sections should be deleted. If not, there is another typo on line 784, should be SHA384.
32. Section 3.3.2.1, lines 855-857: I believe the second sentence in this paragraph can be deleted with no loss of clarity. Add the word 'NIST' before 'recommended cipher suites' in the third sentence.
33. Section 3.3.2.2, lines 866-868: It is unclear why this is important. Recommend deleting the section.
34. Section 3.4.3, 3.4.3.2, 3.6 lines 1196-1198, and 4.4.3 #2: This should be a 'shall not'. Is that possible? We really don't want that to be used.
35. Section 3.7 and 4.7: We really would like this to be a 'shall not' as well.
36. Table 4-1, Extended Key Usage: The anyExtendedKeyUsage extension 'shall not' be present. This is to prevent the certificate from being misused for applications such as code signing.
37. Section 4.4: What is the point here? It seems to be stating a fact vice supplying guidance. Is the client expected to reject a connection where the extensions required are not handled properly?
38. Section 4.4.2, #2.: Negotiated Groups is called 'supported groups' in TLS1.3.
39. Appendix B.1, lines 1544-1548: Please use an example where the cipher suite is preferred (we would like say that a DSS certificate is disallowed).

40. Appendix C: The concept of using a preshared key w/ an asymmetric key pair for either authentication or key agreement should be addressed here to provide interim protection for a quantum computer threat. In the TLS1.3 protocol, it is currently possible to use the combination for key agreement. There is an extension being worked to allow the combination to be used for authentication as well (draft not yet available). Of course in this case, distribution cannot be accomplished via an asymmetric key agreement exchange.
41. Appendix D.2: Including DANE is a fantastic idea, but there are no guidance statements. For example we would not recommend the self-signed options as a 'good' choice. Either of the two options where the certificate is signed by a trusted entity (CA or Root CA), would be our preference. It is also fine for servers, but the use of DANE as a repository for people certificates would have to be examined more closely before recommending it.
42. Appendix E: We need to seriously discourage TLS1.0. What possible use case is there? Any modern server/browser/client is capable of at least TLS1.1.
43. Appendix F: There are some very old references [1], [19], [37], [40], [42], [53], [54]. These should be checked to see if they are the most recent information available.
44. Appendix F: lines 1730-1735: update these references. I would actually reference the base pages, so the reference doesn't go out of date so quickly.