Appendix D. Implementation Notes

The TLS protocol cannot prevent many common security mistakes. This section provides several recommendations to assist implementors.

D.1. Random Number Generation and Seeding

TLS requires a cryptographically secure pseudorandom number generator (PRNG). Care must be taken in designing and seeding PRNGs. PRNGs based on secure hash operations, most notably SHA-1, are acceptable, but cannot provide more security than the size of the random number generator state.

To estimate the amount of seed material being produced, add the number of bits of unpredictable information in each seed byte. For example, keystroke timing values taken from a PC compatible's 18.2 Hz timer provide 1 or 2 secure bits each, even though the total size of the counter value is 16 bits or more. Seeding a 128-bit PRNG would thus require approximately 100 such timer values.

[RANDOM] provides guidance on the generation of random values.

D.2. Certificates and Authentication

Implementations are responsible for verifying the integrity of certificates and should generally support certificate revocation messages. Certificates should always be verified to ensure proper signing by a trusted Certificate Authority (CA). The selection and addition of trusted CAs should be done very carefully. Users should be able to view information about the certificate and root CA.

D.3. Cipher Suites

TLS supports a range of key sizes and security levels, including some that provide no or minimal security. A proper implementation will probably not support many cipher suites. For instance, anonymous Diffie-Hellman is strongly discouraged because it cannot prevent man- in-the-middle attacks. Applications should also enforce minimum and maximum key sizes. For example, certificate chains containing 512- bit RSA keys or signatures are not appropriate for high-security applications.

D.4. Implementation Pitfalls

Implementation experience has shown that certain parts of earlier TLS specifications are not easy to understand, and have been a source of interoperability and security problems. Many of these areas have

been clarified in this document, but this appendix contains a short list of the most important things that require special attention from implementors.

TLS protocol issues:

- Do you correctly handle handshake messages that are fragmented to multiple TLS records (see Section 6.2.1)? Including corner cases like a ClientHello that is split to several small fragments? Do you fragment handshake messages that exceed the maximum fragment size? In particular, the certificate and certificate request handshake messages can be large enough to require fragmentation.
- Do you ignore the TLS record layer version number in all TLS records before ServerHello (see Appendix E.1)?
- Do you handle TLS extensions in ClientHello correctly, including omitting the extensions field completely?
- Do you support renegotiation, both client and server initiated? While renegotiation is an optional feature, supporting it is highly recommended.
- When the server has requested a client certificate, but no suitable certificate is available, do you correctly send an empty Certificate message, instead of omitting the whole message (see Section 7.4.6)?

Cryptographic details:

- In the RSA-encrypted Premaster Secret, do you correctly send and verify the version number? When an error is encountered, do you continue the handshake to avoid the Bleichenbacher attack (see Section 7.4.7.1)?
- What countermeasures do you use to prevent timing attacks against RSA decryption and signing operations (see Section 7.4.7.1)?
- When verifying RSA signatures, do you accept both NULL and missing parameters (see Section 4.7)? Do you verify that the RSA padding doesn't have additional data after the hash value? [FI06]
- When using Diffie-Hellman key exchange, do you correctly strip leading zero bytes from the negotiated key (see Section 8.1.2)?
- Does your TLS client check that the Diffie-Hellman parameters sent by the server are acceptable (see Section F.1.1.3)?
- How do you generate unpredictable IVs for CBC mode ciphers (see Section 6.2.3.2)?
- Do you accept long CBC mode padding (up to 255 bytes; see Section 6.2.3.2)?
- How do you address CBC mode timing attacks (Section 6.2.3.2)?
- Do you use a strong and, most importantly, properly seeded random number generator (see Appendix D.1) for generating the premaster secret (for RSA key exchange), Diffie-Hellman private values, the DSA "k" parameter, and other security-critical values?