dice K. Hartke
Internet-Draft Universitaet Bremen TZI
Intended status: Informational H. Tschofenig
Expires: August 18, 2014 ARM Ltd.
February 14, 2014

A DTLS 1.2 Profile for the Internet of Things
draft-hartke-dice-profile-03

Abstract

This document defines a DTLS profile that is suitable for Internet of Things applications and is reasonably implementable on many constrained devices.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document defines a DTLS 1.2 [RFC6347] profile that offers communication security for Internet of Things (IoT) applications and is reasonably implementable on many constrained devices. It aims to meet the following goals:

- One-stop shop for implementers through the specification jungle.

- This document does not alter the DTLS 1.2 specification.

- This document does not introduce new extensions.

- This profile aligns with the DTLS security modes of the Constrained Application Protocol (CoAP) [I-D.ietf-core-coap].

DTLS is used to secure a number of applications run over an unreliable datagram transport. CoAP [I-D.ietf-core-coap] is one such protocol and has been designed specifically for use in IoT environments. CoAP can be secured using a number of different ways, also called security modes. These security modes are:

No Security Protection at the Transport Layer: No DTLS is used but instead application layer security functionality is assumed.

Shared Secret-based DTLS Authentication: DTLS supports the use of shared secrets [RFC4279]. This credential is useful if the number of communication relationships between the IoT device and servers is small and for very constrained devices. Shared secret-based authentication mechanisms offer good performance and require a minimum of data to be exchanged.

DTLS Authentication using Asymmetric Credentials: TLS supports client and server authentication using asymmetric credentials. Two approaches for validating these public key are available. First, [I-D.ietf-tls-oob-pubkey] allows raw public keys to be used in TLS without the overhead of certificates. This approach requires out-of-band validation of the public key. Second, the use of X.509 certificates [RFC5280] with TLS is common on the Web today (at least for server-side authentication) and certain IoT environments may also re-use those capabilities. Certificates bind an identifier to the public key signed by a certification authority (CA). A trust anchor store has to be provisioned on the device to indicate what CAs are trusted. Furthermore, the certificate may contain a wealth of other information used to make authorization decisions.

As described in [I-D.ietf-lwig-tls-minimal] an application designer developing an IoT device needs to think about the security threats that need to be mitigated. For many Internet connected devices it is, however, likely that authentication of the device and the server infrastructure will be required. Along with the ability to upload sensor data and to retrieve configuration information the need for integrity and confidentiality protection will arise. While these security services can be provided at different layers in the protocol stack the use of channel security, as offered by DTLS, has been very popular on the Internet and it is likely to be useful for IoT scenarios as well. In case the channel security features offered by DTLS meet the security requirements of your application the remainder of the document might offer useful guidance.

Not every IoT deployment will use CoAP but the discussion regarding choice of credentials and cryptographic algorithms will be very similar. As such, the discussions in this document are applicable beyond the use of the CoAP protocol.

The design of DTLS is intentionally very similar to TLS. Since DTLS operates on top of an unreliable datagram transport a few enhancements to the TLS structure are, however necessary. RFC 6347 explains these differences in great detail. As a short summary, for those familiar with TLS the differences are:

- An explicit sequence number and an epoch field is included in the TLS Record Layer. Section 4.1 of RFC 6347 explains the processing rules for these two new fields. The value used to compute the MAC is the 64-bit value formed by concatenating the epoch and the sequence number.

- Stream ciphers must not be used with DTLS. The only stream cipher defined for TLS 1.2 is RC4.

- The TLS Handshake Protocol has been enhanced to include a stateless cookie exchange for Denial of Service (DoS) resistance. Furthermore, the header has been extended to deal with message loss, reordering, and fragmentation. Retransmission timers have been included to deal with message loss. For DoS protection a new handshake message, the HelloVerifyRequest, was added to DTLS. This handshake message is sent by the server and includes a stateless cookie, which is returned in a ClientHello message back to the server. This type of DoS protection mechanism has also been incorporated into the design of IKEv2. Although the exchange is optional for the server to execute, a client implementation has to be prepared to respond to it.

2. The Communication Model

This document describes a profile of DTLS 1.2 and to be useful it has to make assumptions about the envisioned communication architecture. The architecture shown in Figure 1 assumes a uni-cast communication interaction with an IoT device acting as a client and the client interacts with one or multiple servers. Which server to contact is based on pre-configuration onto the client (e.g., as part of the firmware). This configuration information also includes information about the PSK identity and the corresponding secret to be used with that specific server (in case of symmetric credentials). For asymmetric cryptography mutual authentication is assumed in this profile. For raw public keys the public key or the hash of the public key is assumed to be available to both parties. For certificate-based authentication the client may have a trust anchor store pre-populated, which allows the client to perform path validation for the certificate obtained during the handshake with the server. The client also needs to know which certificate or raw public key it has to use with a specific server.

This document only focuses on the description of the DTLS client-side functionality.

```
              +////////////////////////////////////+
              |          Configuration             |
              |////////////////////////////////////|
              | Server A --> PSK Identity, PSK     |
              | Server B --> Public Key (Server B),|
              |              Public Key (Client)   |
              | Server C --> Public Key (Client),  |
              |              Trust Anchor Store    |
              +------------------------------------+
                oo
          oooooo
         o
      +------+
      |Client|---
      +------+   \
                  \  ,-------.
                   ,'         `.            +------+
                  /  IP-based   \           |Server|
                 (    Network    )          |  A   |
                  \             /           +------+
                   `.         ,'
                     '---+---'                  +------+
                         |                      |Server|
                         |                      |  B   |
                         |                      +------+
                         |
                         |                  +------+
                         +----------------->|Server|
                                            |  C   |
                                            +------+
```

Figure 1: DTLS Profile: Assumed Communication Model.

A future version of this document may provide profiles for other communication architectures.

3. The Ciphersuite Concept

TLS (and consequently DTLS) introduced the concept of ciphersuites and an IANA registry [IANA-TLS] was created to keep track of the specified suites. A ciphersuites (and the specification that defines it) contains the following information:

o Authentication and Key Exchange Algorithm (e.g., PSK)

o Cipher and Key Length(e.g., AES with 128 bit keys)

- o Mode of operation (e.g., CBC)

- o Hash Algorithm for Integrity Protection (e.g., SHA in combination with HMAC)

- o Hash Algorithm for use with the Pseudorandom Function (e.g. HMAC with the SHA-256)

- o Misc information (e.g., length of authentication tags)

The TLS ciphersuite TLS_PSK_WITH_AES_256_CBC_SHA, for example, uses a pre-shared authentication and key exchange algorithm. RFC 4279, which defined this ciphersuite predates publication of TLS 1.2. It uses the Advanced Encryption Standard (AES) encryption algorithm, which is a block cipher. Since the AES algorithm supports different key lengths (such as 128, 192 and 256 bits) this information has to be specified as well and the selected ciphersuite supports 256 bit keys. A block cipher encrypts plaintext in fixed-size blocks and AES operates on fixed block size of 128 bits. For messages exceeding 128 bits, the message is partitioned into 128-bit blocks and the AES cipher is applied to these input blocks with appropriate chaining, which is called mode of operation. In our example, the mode of operation is cipher block chaining (CBC). Since encryption itself does not provide integrity protection a hash function is specified as well, which will be used in concert with the HMAC function. In this case, the Secure Hash Algorithm (SHA).

TLS 1.2 introduced Authenticated Encryption with Associated Data (AEAD) ciphersuites. AEAD is a class of block cipher modes which encrypt (parts of) the message and authenticate the message simultaneously. Examples of such modes include the Counter with CBC-MAC (CCM) mode, and the Galois/Counter Mode (GCM).

TLS 1.2 also replaced the combination of MD5/SHA-1 hash functions in the TLS pseudo random function (PRF) with cipher-suite-specified PRFs. For this reason authors of more recent TLS 1.2 ciphersuite specifications explicitly indicate the MAC algorithm and the hash functions used with the TLS PRF.

4. Pre-Shared Secret Authentication with DTLS

The use of pre-shared secret credentials is one of the most basic techniques for DTLS since it is both computational efficient and bandwidth conserving. Pre-shared secret based authentication was introduced to TLS with RFC 4279 [RFC4279]. The exchange shown in Figure 2 illustrates the DTLS exchange including the cookie exchange. While the server is not required to initiate a cookie exchange with every handshake, the client is required to implement and to react on it when challenged.

```
      Client                                               Server
      ------                                               ------
      ClientHello                 -------->

                                  <--------    HelloVerifyRequest
                                                (contains cookie)

      ClientHello                  -------->
      (with cookie)
                                                      ServerHello
                                               *ServerKeyExchange
                                   <--------      ServerHelloDone
      ClientKeyExchange
      ChangeCipherSpec
      Finished                     -------->
                                                 ChangeCipherSpec
                                   <--------             Finished

      Application Data             <------->     Application Data
```

Legend:

* indicates an optional message payload

Figure 2: DTLS PSK Authentication including the Cookie Exchange.

[RFC4279] does not mandate the use of any particular type of identity. Hence, the TLS client and server clearly have to agree on the identities and keys to be used. The mandated encoding of identities in Section 5.1 of RFC 4279 aims to improve interoperability for those cases where the identity is configured by a person using some management interface. Many IoT devices do, however, not have a user interface and most of their credentials are bound to the device rather than the user. Furthermore, credentials are provisioned into trusted hardware modules or in the firmware by the developers. As such, the encoding considerations are not applicable to this usage environment. For use with this profile the PSK identities MUST NOT assume a structured format (as domain names, Distinguished Names, or IP addresses have) and a bit-by-bit comparison operation can then be used by the server-side infrastructure.

As described in Section 2 clients may have pre-shared keys with several different servers. The client indicates which key it uses by including a "PSK identity" in the ClientKeyExchange message. To help the client in selecting which PSK identity / PSK pair to use, the server can provide a "PSK identity hint" in the ServerKeyExchange message. For Iot environments a simplifying assumption is made that the hint for PSK key selection is based on the domain name of the server. Hence, servers SHOULD NOT send the "PSK identity hint" in the ServerKeyExchange message and client MUST ignore the message.

RFC 4279 requires TLS implementations supporting PSK ciphersuites to support arbitrary PSK identities up to 128 octets in length, and arbitrary PSKs up to 64 octets in length. This is a useful assumption for TLS stacks used in the desktop and mobile environment where management interfaces are used to provision identities and keys. For the IoT environment, however, many devices are not equipped with displays and input devices (e.g., keyboards). Hence, keys are distributed as part of hardware modules or are embedded into the firmware. As such, these restrictions are not applicable to this profile.

Constrained Application Protocol (CoAP) [I-D.ietf-core-coap] currently specifies TLS_PSK_WITH_AES_128_CCM_8 as the mandatory to implement ciphersuite for use with shared secrets. This ciphersuite uses the AES algorithm with 128 bit keys and CCM as the mode of operation. The label "_8" indicates that an 8-octet authentication tag is used. This ciphersuite makes use of the default TLS 1.2 Pseudorandom Function (PRF), which uses HMAC with the SHA-256 hash function.

5. Raw Public Key Use with DTLS

The use of raw public keys with DTLS, as defined in [I-D.ietf-tls-oob-pubkey], is the first entry point into public key cryptography without having to pay the price of certificates and a PKI. The specification re-uses the existing Certificate message to convey the raw public key encoded in the SubjectPublicKeyInfo structure. To indicate support two new TLS extensions had been defined as shown in Figure 3, namely the server_certificate_type and the client_certificate_type. To operate this mechanism securely it is necessary to authenticate and authorize the public keys out-of-band. This document therefore assumes that a client implementation comes with one or multiple raw public keys of servers, it has to communicate with, pre-provisioned. Additionally, a device will have its own raw public key. To replace, delete, or add raw public key to this list requires a software update, for example using a firmware update.

```
    Client                                           Server
    ------                                           ------

    ClientHello             -------->
    client_certificate_type
    server_certificate_type

                            <-------    HelloVerifyRequest

    ClientHello             -------->
    client_certificate_type
    server_certificate_type

                                               ServerHello
                                   client_certificate_type
                                   server_certificate_type
                                               Certificate
                                         ServerKeyExchange
                                        CertificateRequest
                            <--------      ServerHelloDone

    Certificate
    ClientKeyExchange
    CertificateVerify
    [ChangeCipherSpec]
    Finished                -------->

                                        [ChangeCipherSpec]
                            <--------             Finished
```

Figure 3: DTLS Raw Public Key Exchange including the Cookie Exchange.

The ciphersuite for use with this credential type is TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 [I-D.mcgrew-tls-aes-ccm-ecc]. This elliptic curve cryptography (ECC) based AES-CCM TLS ciphersuite uses the Elliptic Curve Diffie Hellman (ECDHE) as the key establishment mechanism and an Elliptic Curve Digital Signature Algorithm (ECDSA) for authentication. This ciphersuite make use of the AEAD capability in DTLS 1.2 and utilizes an eight-octet authentication tag. Based on the Diffie-Hellman it provides perfect forward secrecy (PFS). More details about the PFS can be found in Section 10.

RFC 6090 [RFC6090] provides valuable information for implementing Elliptic Curve Cryptography algorithms.

Since many IoT devices will either have limited ways to log error or no ability at all, any error will lead to implementations attempting to re-try the exchange.

QUESTION: [I-D.sheffer-tls-bcp] recommends a different ciphersuite, namely TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 [RFC5289] or alternatively TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 (with a 2048-bit or 1024 DH parameters as second and third priority, respectively). Is TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 a good choice?

6. Certificate Use with DTLS

The use of mutual certificate-based authentication is shown in Figure 4. Note that the figure also makes use of the cached info extension, which is indicated by the TLS extension (cached_information) and the changed content in the exchanged certificates. Caching certificate chains allows the client to reduce the communication overhead significantly since otherwise the server would provide the end entity certificate, and the certificate chain. Because certificate validation requires that root keys be distributed independently, the self-signed certificate that specifies the root certificate authority is omitted from the chain. Client implementations MUST be provisioned with a trust anchor store that contains the root certificates. The use of the Trust Anchor Management Protocol (TAMP) [RFC5934] is, however, not envisioned. Instead IoT devices using this profile MUST rely a software update mechanism to provision these trust anchors.

When DTLS is used to secure CoAP messages then the server provided certificates MUST contain the fully qualified DNS domain name or "FQDN". The coaps URI scheme is described in Section 6.2 of [I-D.ietf-core-coap]. This FQDN is stored in the SubjectAltName or in the CN, as explained in Section 9.1.3.3 of [I-D.ietf-core-coap], and used by the client to match it against the FQDN used during the look-up process, as described in RFC 6125 [RFC6125]. For the profile in this specification does not assume dynamic discovery of local servers.

For client certificates the identifier used in the SubjectAltName or in the CN MUST be an EUI-64 [EUI64], as mandated in Section 9.1.3.3 of [I-D.ietf-core-coap].

For certificate revocation neither the Online Certificate Status Protocol (OCSP) nor Certificate Revocation Lists (CRLs) are used. Instead, this profile relies on a software update mechanism. While multiple OCSP stapling [RFC6961] has recently been introduced as a mechanism to piggyback OCSP request/responses inside the DTLS/TLS handshake to avoid the cost of a separate protocol handshake further investigations are needed to determine its suitability for the IoT environment.

```
 Client                                               Server
 ------                                               ------

 ClientHello             -------->
 cached_information

                         <-------    HelloVerifyRequest

 ClientHello             -------->
 cached_information
                                            ServerHello
                                     cached_information
                                            Certificate
                                      ServerKeyExchange
                                     CertificateRequest
                         <--------      ServerHelloDone

 Certificate
 ClientKeyExchange
 CertificateVerify
 [ChangeCipherSpec]
 Finished                -------->

                                     [ChangeCipherSpec]
                         <--------             Finished
```

Figure 4: DTLS Mutual Certificate-based Authentication.

Regarding the ciphersuite choice the discussion in Section 5 applies. Further details about X.509 certificates can be found in Section 9.1.3.3 of [I-D.ietf-core-coap].

QUESTION: What restrictions regarding the depth of the certificate chain should be made? Is one level enough?

7. Error Handling

DTLS uses the Alert protocol to convey error messages and specifies a longer list of errors. However, not all error messages defined in the TLS specification are applicable to this profile. All error messages marked as RESERVED are only supported for backwards compatibility with SSL and are therefore not applicable to this profile. Those include decryption_failed_RESERVED, no_certificate_RESERVE, and export_restriction_RESERVED. A number of the error messages are applicable only for certificate-based authentication ciphersuites. Hence, for PSK and raw public key use the following error messages are not applicable: bad_certificate, unsupported_certificate, certificate_revoked, certificate_expired, certificate_unknown, unknown_ca, and access_denied.

Since this profile does not make use of compression at the TLS layer the decompression_failure error message is not applicable either.

RFC 4279 introduced a new alert message unknown_psk_identity for PSK ciphersuites. As stated in Section 2 of RFC 4279 the decryption_error error message may also be used instead. For this profile the TLS server MUST return the decryption_error error message instead of the unknown_psk_identity.

Furthermore, the following errors should not occur based on the description in this specification:

protocol_version: This document only focuses on one version of the DTLS protocol.

insufficient_security: This error message indicates that the server requires ciphers to be more secure. This document does, however, specify the only acceptable ciphersuites and client implementations must support them.

user_canceled: The IoT devices in focus of this specification are assumed to be unattended.

8. Session Resumption

Session resumption is a feature of DTLS that allows a client to continue with an earlier established session state. The resulting exchange is shown in Figure 5. In addition, the server may choose not to do a cookie exchange when a session is resumed. Still, clients have to be prepared to do a cookie exchange with every handshake.

```
         Client                                               Server
         ------                                               ------

         ClientHello                  -------->
                                                         ServerHello
                                                  [ChangeCipherSpec]
                                      <--------             Finished
         [ChangeCipherSpec]
         Finished                     -------->
         Application Data             <------->     Application Data
```

Figure 5: DTLS Session Resumption.

Clients MUST implement session resumption to improve the performance of the handshake (in terms of reduced number of message exchanges, lower computational overhead, and less bandwidth conserved).

Since the communication model described in Section 2 does not assume that the server is constrained. RFC 5077 [RFC5077] describing TLS session resumption without server-side state is not utilized by this profile.

9. TLS Compression

[I-D.sheffer-tls-bcp] recommends to always disable DTLS-level compression due to attacks. For IoT applications compression at the DTLS is not needed since application layer protocols are highly optimized and the compression algorithms at the DTLS layer increase code size and complexity. Hence, for use with this profile compression at the DTLS layer MUST NOT be implemented by the DTLS client.

10. Perfect Forward Secrecy

Perfect forward secrecy is designed to prevent the compromise of a long-term secret key from affecting the confidentiality of past conversations. The PSK ciphersuite recommended in the CoAP specification [I-D.ietf-core-coap] does not offer this property. [I-D.sheffer-tls-bcp] on the other hand recommends using ciphersuites offering this security property.

QUESTION: Should the PSK ciphersuite offer PFS?

11. Keep-Alive

RFC 6520 [RFC6520] defines a heartbeat mechanism to test whether the other peer is still alive. The same mechanism can also be used to perform path MTU discovery.

QUESTION: Do IoT deployments make use of this extension?

12. Negotiation and Downgrading Attacks

CoAP demands version 1.2 of DTLS to be used and the earlier version of DTLS is not supported. As such, there is no risk of downgrading to an older version of DTLS. The work described in [I-D.bmoeller-tls-downgrade-scsv] is therefore also not applicable to this environment since there is no legacy server infrastructure to worry about.

QUESTION: Should we say something for non-CoAP use of DTLS?

To prevent the TLS renegotiation attack [RFC5746] clients MUST respond to server-initiated renegotiation attempts with an Alert message (no_renegotiation) and clients MUST NOT initiate them. TLS and DTLS allows a client and a server who already have a TLS connection to negotiate new parameters, generate new keys, etc by initiating a TLS handshake using a ClientHello message. Renegotiation happens in the existing TLS connection, with the new handshake packets being encrypted along with application data.

13. Privacy Considerations

The DTLS handshake exchange conveys various identifiers, which can be observed by an on-path eavesdropper. For example, the DTLS PSK exchange reveals the PSK identity, the supported extensions, the session id, algorithm parameters, etc. When session resumption is used then individual TLS sessions can be correlated by an on-path adversary. With many IoT deployments it is likely that keying material and their identifiers are persistent over a longer period of time due to the cost of updating software on these devices.

User participation with many IoT deployments poses a challenge since many of the IoT devices operate unattended, even though they will initially be enabled by a human. The ability to control data sharing and to configure preference will have to be provided at a system level rather than at the level of a DTLS profile, which is the scope of this document. Quite naturally, the use of DTLS with mutual authentication will allow a TLS server to collect authentication information about the IoT device (potentially over a long period of time). While this strong form of authentication will prevent misattribution it also allows strong identification. This device-related data collection (e.g., sensor recordings) will be associated with other data to be truly useful and this extra data might include personal data about the owner of the device or data about the environment it senses. Consequently, the data stored on the server-side will be vulnerable to stored data compromise. For the communication between the client and the server this specification prevents eavesdroppers to gain access to the communication content. While the PSK-based ciphersuite does not provide PFS the asymmetric version does. No explicit techniques, such as extra padding, have been provided to make traffic analysis more difficult.

14. Security Considerations

This entire document is about security.

The TLS protocol requires random numbers to be available during the protocol run. For example, during the ClientHello and the ServerHello exchange the client and the server exchange random numbers. Also, the use of the Diffie Hellman exchange requires random numbers during the key pair generation. Special care has to be paid when generating random numbers in embedded systems as many entropy sources available on desktop operating systems or mobile devices might be missing, as described in [Heninger]. Consequently, if not enough time is given during system start time to fill the entropy pool then the output might be predictable and repeatable, for example leading to the same keys generated again and again. Guidelines and requirements for random number generation can be found in RFC 4086 [RFC4086].

We would also like to point out that designing a software update mechanism into an IoT system is crucial to ensure that both functionality can be enhanced and that potential vulnerabilities can be fixed. This software update mechanism is also useful for changing configuration information, for example, trust anchors and other keying related information.

15. IANA Considerations

This document includes no request to IANA.

16. Acknowledgements

Thanks to Rene Hummen, Sye Loong Keoh, Sandeep Kumar, Eric Rescorla, Zach Shelby, and Sean Turner for helpful comments and discussions that have shaped the document.

17. References

17.1. Normative References

[EUI64] "GUIDELINES FOR 64-BIT GLOBAL IDENTIFIER (EUI-64) REGISTRATION AUTHORITY", April 2010, <http://standards.ieee.org/regauth/oui/tutorials/EUI64.html>.

[I-D.ietf-core-coap] Shelby, Z., Hartke, K., and C. Bormann, "Constrained Application Protocol (CoAP)", draft-ietf-core-coap-18 (work in progress), June 2013.

[I-D.ietf-tls-cached-info] Santesson, S. and H. Tschofenig, "Transport Layer Security (TLS) Cached Information Extension", draft-ietf-tls-cached-info-15 (work in progress), October 2013.

[I-D.ietf-tls-oob-pubkey] Wouters, P., Tschofenig, H., Gilmore, J., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", draft-ietf-tls-oob-pubkey-11 (work in progress), January 2014.

[I-D.mcgrew-tls-aes-ccm-ecc] McGrew, D., Bailey, D., Campagna, M., and R. Dugal, "AES-CCM ECC Cipher Suites for TLS", draft-mcgrew-tls-aes-ccm-ecc-08 (work in progress), February 2014.

[RFC4279] Eronen, P. and H. Tschofenig, "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", RFC 4279, December 2005.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5746] Rescorla, E., Ray, M., Dispensa, S., and N. Oskov, "Transport Layer Security (TLS) Renegotiation Indication Extension", RFC 5746, February 2010.

[RFC6066] Eastlake, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, January 2011.

[RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, March 2011.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

[RFC6520] Seggelmann, R., Tuexen, M., and M. Williams, "Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension", RFC 6520, February 2012.

17.2. Informative References

[Heninger] Heninger, N., Durumeric, Z., Wustrow, E., and A. Halderman, "Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices", 21st USENIX Security Symposium, https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/heninger, 2012.

[I-D.bmoeller-tls-downgrade-scsv] Moeller, B. and A. Langley, "TLS Fallback Signaling Cipher Suite Value (SCSV) for Preventing Protocol Downgrade Attacks", draft-bmoeller-tls-downgrade-scsv-01 (work in progress), November 2013.

[I-D.campagna-suitee] Campagna, M., "A Cryptographic Suite for Embedded Systems (SuiteE)", draft-campagna-suitee-04 (work in progress), October 2012.

[I-D.cooper-ietf-privacy-requirements] Cooper, A., Farrell, S., and S. Turner, "Privacy Requirements for IETF Protocols", draft-cooper-ietf-privacy-requirements-01 (work in progress), October 2013.

[I-D.greevenbosch-tls-ocsp-lite] Greevenbosch, B., "OCSP-lite - Revocation of raw public keys", draft-greevenbosch-tls-ocsp-lite-01 (work in progress), June 2013.

[I-D.gutmann-tls-encrypt-then-mac] Gutmann, P., "Encrypt-then-MAC for TLS and DTLS", draft-gutmann-tls-encrypt-then-mac-05 (work in progress), December 2013.

[I-D.hummen-dtls-extended-session-resumption]
Hummen, R., Gilger, J., and H. Shafagh, "Extended DTLS Session Resumption for Constrained Network Environments", draft-hummen-dtls-extended-session-resumption-01 (work in progress), October 2013.

[I-D.ietf-lwig-guidance]
Bormann, C., "Guidance for Light-Weight Implementations of the Internet Protocol Suite", draft-ietf-lwig-guidance-03 (work in progress), February 2013.

[I-D.ietf-lwig-terminology]
Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained Node Networks", draft-ietf-lwig-terminology-06 (work in progress), December 2013.

[I-D.ietf-lwig-tls-minimal]
Kumar, S., Keoh, S., and H. Tschofenig, "A Hitchhiker's Guide to the (Datagram) Transport Layer Security Protocol for Smart Objects and Constrained Node Networks", draft-ietf-lwig-tls-minimal-00 (work in progress), September 2013.

[I-D.ietf-tls-applayerprotoneg]
Friedl, S., Popov, A., Langley, A., and S. Emile, "Transport Layer Security (TLS) Application Layer Protocol Negotiation Extension", draft-ietf-tls-applayerprotoneg-04 (work in progress), January 2014.

[I-D.pettersen-tls-version-rollback-removal]
Pettersen, Y., "Managing and removing automatic version rollback in TLS Clients", draft-pettersen-tls-version-rollback-removal-02 (work in progress), August 2013.

[I-D.sheffer-tls-bcp]
Sheffer, Y. and R. Holz, "Recommendations for Secure Use of TLS and DTLS", draft-sheffer-tls-bcp-01 (work in progress), September 2013.

[IANA-TLS]
IANA, "TLS Cipher Suite Registry", http://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml#tls-parameters-4, 2014.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, July 2003.

[RFC4086] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

[RFC4492] Blake-Wilson, S., Bolyard, N., Gupta, V., Hawk, C., and B. Moeller, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)", RFC 4492, May 2006.

[RFC5077] Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State", RFC 5077, January 2008.

[RFC5289] Rescorla, E., "TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM)", RFC 5289, August 2008.

[RFC5934] Housley, R., Ashmore, S., and C. Wallace, "Trust Anchor Management Protocol (TAMP)", RFC 5934, August 2010.

[RFC6090] McGrew, D., Igoe, K., and M. Salter, "Fundamental Elliptic Curve Cryptography Algorithms", RFC 6090, February 2011.

[RFC6961] Pettersen, Y., "The Transport Layer Security (TLS) Multiple Certificate Status Request Extension", RFC 6961, June 2013.

[RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, July 2013.

Authors’ Addresses

Klaus Hartke
Universitaet Bremen TZI
Postfach 330440
Bremen D-28359
Germany

Phone: +49-421-218-63905
Email: hartke@tzi.org

Hannes Tschofenig
ARM Ltd.
110 Fulbourn Rd
Cambridge CB1 9NJ
Great Britain

Email: Hannes.tschofenig@gmx.net
URI: http://www.tschofenig.priv.at