

dice
Internet-Draft
Intended status: Standards Track
Expires: January 5, 2015

H. Tschofenig, Ed.
ARM Ltd.
July 4, 2014

A Datagram Transport Layer Security (DTLS) 1.2 Profile for the Internet
of Things
draft-ietf-dice-profile-03.txt

Abstract

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

A common design pattern in IoT deployments is the use of a constrained device (typically providing sensor data) that interacts with the web infrastructure. This document focuses on this particular pattern.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 5, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|---|----|
| 1. Introduction | 2 |
| 2. Terminology | 4 |
| 3. The Communication Model | 5 |
| 4. The Ciphersuite Concept | 6 |
| 5. Pre-Shared Secret Authentication with DTLS | 8 |
| 6. Raw Public Key Use with DTLS | 9 |
| 7. Certificate Use with DTLS | 11 |
| 8. Error Handling | 12 |
| 9. Session Resumption | 13 |
| 10. Compression | 14 |
| 11. Perfect Forward Secrecy | 14 |
| 12. Keep-Alive | 15 |
| 13. Random Number Generation | 16 |
| 14. Client Certificate URLs | 17 |
| 15. Trusted CA Indication | 17 |
| 16. Truncated MAC Extension | 17 |
| 17. Server Name Indication (SNI) | 18 |
| 18. Maximum Fragment Length Negotiation | 18 |
| 19. Negotiation and Downgrading Attacks | 18 |
| 20. Privacy Considerations | 19 |
| 21. Security Considerations | 19 |
| 22. IANA Considerations | 20 |
| 23. Acknowledgements | 20 |
| 24. References | 20 |
| 24.1. Normative References | 20 |
| 24.2. Informative References | 21 |
| Author's Address | 23 |

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:

- o Serves as a one-stop shop for implementers to know which pieces of the specification jungle contain relevant details.
- o Does not alter the DTLS 1.2 specification.
- o Does not introduce any new extensions.

- o Aligns with the DTLS security modes of the Constrained Application Protocol (CoAP) [RFC7252].

DTLS is used to secure a number of applications run over an unreliable datagram transport. CoAP [RFC7252] is one such protocol and has been designed specifically for use in IoT environments. CoAP can be secured a number of different ways, also called security modes. These security modes are as follows, see Section 5, Section 6, Section 7 for additional details:

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 mode 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 Cryptography: TLS supports client and server authentication using asymmetric cryptography. Two approaches for validating these public keys are available. First, [RFC7250] 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 consider the security threats and the security services that can be used to mitigate the threats. Enabling devices to upload data and retrieve configuration information, inevitably requires that Internet-connected devices be able to authenticate themselves to servers and vice versa as well as to ensure that the data and information exchanged is integrity and confidentiality protected. While these security services can be provided at different layers in the protocol stack the use of communication 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 communication 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 not familiar with DTLS the differences are:

- o 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.
- o Stream ciphers must not be used with DTLS. The only stream cipher defined for TLS 1.2 is RC4 and due to cryptographic weaknesses it is not recommended anymore even for use with TLS.
- o 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. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "MUST", "MUST NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Note that "Client" and "Server" in this document refer to TLS roles, where the Client initiates the TLS handshake. This does not restrict the interaction pattern of the protocols carried inside TLS as the record layer allows bi-directional communication. In the case of CoAP the "Client" can act as a CoAP Server or Client.

3. 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 communication architecture shown in Figure 1 assumes a uni-cast communication interaction with an IoT device utilizing a DTLS client and that client interacts with one or multiple DTLS servers.

Clients are preconfigured with the address or addresses of servers (e.g., as part of the firmware) they will communicate with as well as authentication information:

- o For PSK-based authentication (see Section 5), this includes the paired "PSK identity" and shared secret to be used with each server.
- o For raw public key-based authentication (see Section 6), this includes either the server's public key or the hash of the server's public key.
- o For certificate-based authentication (see Section 7), this may include a pre-populated trust anchor store that allows the client to perform path validation for the certificate obtained during the handshake with the server.

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

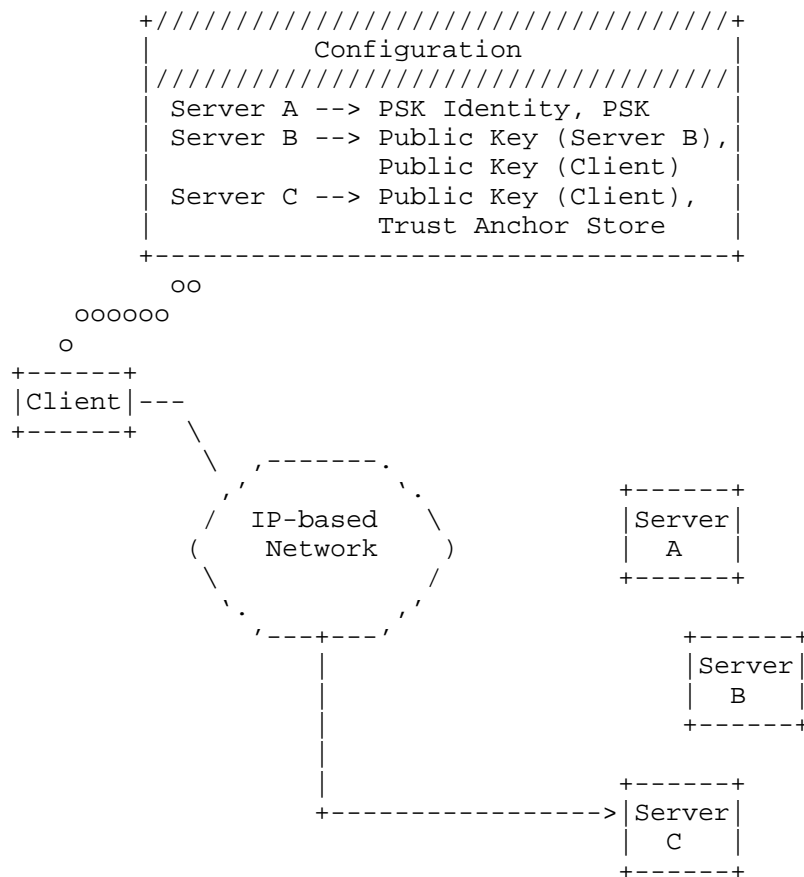


Figure 1: Constrained DTLS Client Profile.

4. The Ciphersuite Concept

TLS (and consequently DTLS) has the concept of ciphersuites and an IANA registry [IANA-TLS] was created to register the suites. A ciphersuite (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)
- o Information whether the ciphersuite is suitable for DTLS or only for TLS

The TLS ciphersuite TLS_PSK_WITH_AES_128_CCM_8, for example, uses a pre-shared authentication and key exchange algorithm. RFC 6655 [RFC6655] defines this ciphersuite. 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 128 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.

TLS 1.2 introduced Authenticated Encryption with Associated Data (AEAD) ciphersuites [RFC5116]. 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).

Some AEAD ciphersuites have shorter authentication tags and are therefore more suitable for networks with low bandwidth where small message size matters. The TLS_PSK_WITH_AES_128_CCM_8 ciphersuite that ends in "_8" has an 8-octet authentication tag, while the regular CCM ciphersuites have 16-octet authentication tags.

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.

This document references the CoAP recommended ciphersuite choices, which have been selected based on implementation and deployment experience from the IoT community. Over time the preference for certain algorithms will, however, change. Not all components of a ciphersuite change at the same speed. Changes are more likely to expect for ciphers, the mode of operation, and the hash algorithms.

Some deployment environments will also be impacted by local regulation, which might dictate a certain and less likely for public key algorithms (such as RSA vs. ECC).

5. 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 SHOULD 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 3 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. This approach is inline with RFC 4279 [RFC4279].

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) [RFC7252] 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.

6. Raw Public Key Use with DTLS

The use of raw public keys with DTLS, as defined in [RFC7250], 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 mechanism.

```

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 CoAP recommended ciphersuite for use with this credential type is `TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8` [RFC7251]. 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. The use of a Diffie-Hellman key exchange adds perfect forward secrecy (PFS). More details about PFS can be found in Section 11.

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.

7. Certificate Use with DTLS

The use of mutual certificate-based authentication is shown in Figure 4, which makes use of the cached info extension [I-D.ietf-tls-cached-info]. Support of the cached info extension is required. 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 [RFC7252]. This FQDN is stored in the SubjectAltName or in the CN, as explained in Section 9.1.3.3 of [RFC7252], 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 [RFC7252].

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.

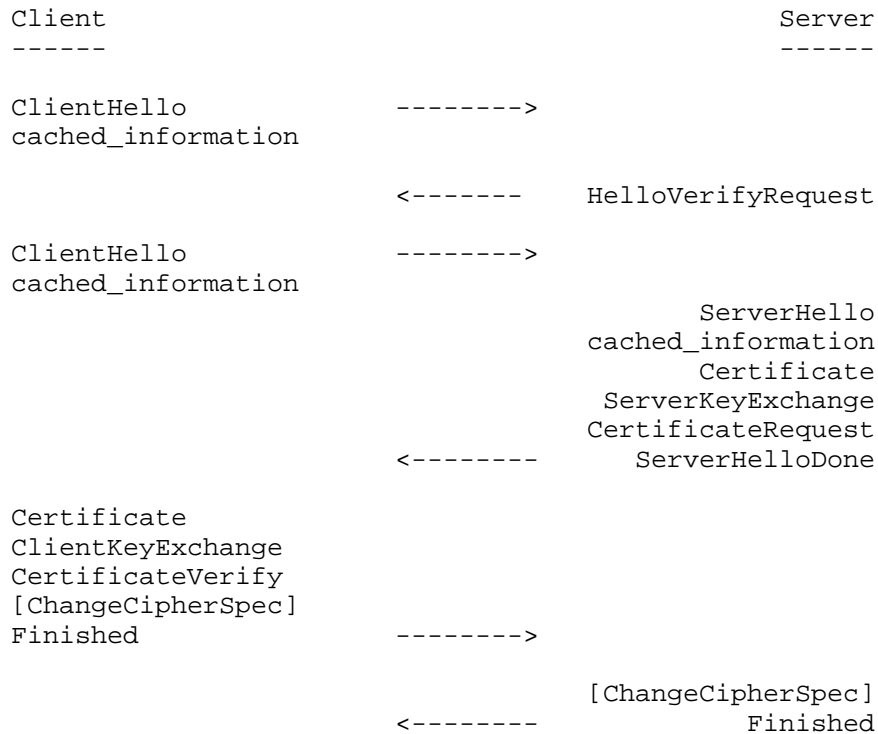


Figure 4: DTLS Mutual Certificate-based Authentication.

Regarding the ciphersuite choice the discussion in Section 6 applies. Further details about X.509 certificates can be found in Section 9.1.3.3 of [RFC7252].

8. 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:

- o bad_certificate,
- o unsupported_certificate,
- o certificate_revoked,
- o certificate_expired,
- o certificate_unknown,
- o unknown_ca, and
- o 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.

9. 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.

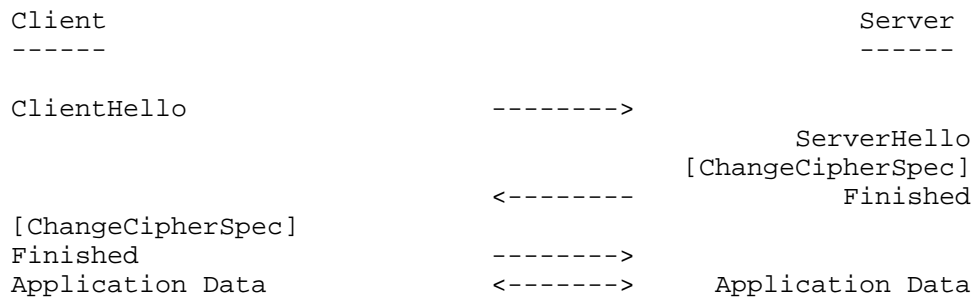


Figure 5: DTLS Session Resumption.

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 3 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.

10. Compression

[I-D.ietf-uta-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.

This DTLS client profile does not include DTLS layer compression.

11. Perfect Forward Secrecy

Perfect forward secrecy (PFS) 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 [RFC7252] does not offer this property since it does not utilize a Diffie-Hellman exchange. [I-D.ietf-uta-tls-bcp] on the other hand recommends using ciphersuites offering this security property and so do the public key-based ciphersuites recommended by the CoAP specification.

The use of PFS is certainly a trade-off decision since on one hand the compromise of long-term secrets of embedded devices is more likely than with many other Internet hosts but on the other hand a Diffie-Hellman exchange requires ephemeral key pairs to be generated, which can be demanding from a performance point of view. Finally, the impact of the disclosure of past conversations and the desire to increase the cost for pervasive monitoring (see [RFC7258]) has to be taken into account.

Our recommendation is to stick with the ciphersuite suggested in the CoAP specification. New ciphersuites support PFS for pre-shared secret-based authentication, such as [I-D.schmertmann-dice-ccm-psk-pfs], and might be available as a standardized ciphersuite in the future.

12. 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 Maximum Transmission Unit (MTU) Discovery.

A recommendation about the use of RFC 6520 depends on the type of message exchange an IoT device performs. There are three types of exchanges that need to be analysed:

Client-Initiated, One-Shot Messages

This is a common communication pattern where IoT devices upload data to a server on the Internet on an irregular basis. The communication may be triggered by specific events, such as opening a door.

Since the upload happens on an irregular and unpredictable basis and due to renumbering and Network Address Translation (NAT) a new DTLS session or DTLS session resumption can be used.

In this case there is no use for a keep-alive extension for this scenario.

Client-Initiated, Regular Data Uploads

This is a variation of the previous case whereby data gets uploaded on a regular basis, for example, based on frequent temperature readings. With such regular exchange it can be assumed that the DTLS context is still in kept at the IoT device. If neither NAT bindings nor IP address changes occurred then the DTLS record layer will not notice any changes. For the case where

IP and port changes happened it is necessary to re-create the DTLS record layer using session resumption.

In this scenario there is no use for a keep-alive extension. It is also very likely that the device will enter a sleep cycle in between data transmissions to keep power consumption low.

Server-Initiated Messages

In the two previous scenarios the client initiated the protocol interaction. In this case, we consider server-initiated messages. Since messages to the client may get blocked by intermediaries, such as NATs and stateful packet filtering firewalls, the initial connection setup is triggered by the client and then kept alive. Since state expires fairly quickly at middleboxes regular heartbeats are necessary whereby these keep-alive messages may be exchanged at the application layer or within DTLS itself.

For this message exchange pattern the use of DTLS heartbeat messages is quite useful. The MTU discovery mechanism, on the other hand, is less likely to be relevant since for many IoT deployments the most constrained link is the wireless interface at the IoT device itself rather than somewhere in the network. Only in more complex network topologies the situation might be different.

For server-initiated messages the heartbeat extension can be recommended.

13. Random Number Generation

The DTLS 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.

Recommendation: IoT devices using DTLS MUST offer ways to generate quality random numbers. Guidelines and requirements for random number generation can be found in RFC 4086 [RFC4086].

It is important to note that sources contributing to the randomness pool on laptops, or desktop PCs are not available on many IoT device, such as mouse movement, timing of keystrokes, air turbulence on the movement of hard drive heads, etc. Other sources have to be found or dedicated hardware has to be added.

14. Client Certificate URLs

This RFC 6066 [RFC6066] extension allows to avoid sending client-side certificates and URLs instead. This reduces the over-the-air transmission.

This is certainly a useful extension when a certificate-based mode for DTLS is used since the TLS cached info extension does not provide any help with caching information on the server side.

Recommendation: Add support for client certificate URLs for those environments where client-side certificates are used.

15. Trusted CA Indication

This RFC 6066 extension allows clients to indicate what trust anchor they support. With certificate-based authentication a DTLS server conveys its end entity certificate to the client during the DTLS exchange provides. Since the server does not necessarily know what trust anchors the client has stored it includes intermediate CA certs in the certificate payload as well to facilitate with certification path construction and path validation.

Today, in most IoT deployments there is a fairly static relationship between the IoT device (and the software running on them) and the server- side infrastructure and no such dynamic indication of trust anchors is needed.

Recommendation: For IoT deployments where clients talk to a fixed, pre-configured set of servers and where a software update mechanism is available this extension is not recommended. Environments where the client needs to interact with dynamically discovered DTLS servers this extension may be useful to reduce the communication overhead. Note, however, in that case the TLS cached info extension may help to reduce the communication overhead for everything but the first protocol interaction.

16. Truncated MAC Extension

This RFC 6066 extension was introduced to reduces the size of the MAC used at the Record Layer. This extension was developed for TLS

ciphersuites that used older modes of operation where the MAC and the encryption operation was performed independently.

For CoAP, however, the recommended ciphersuites use the newer Authenticated Encryption with Associated Data (AEAD) construct, namely the CBC-MAC mode (CCM) with eight-octet authentication tags.

Recommendation: Since this profile only supports AEAD ciphersuites this extension is not applicable.

17. Server Name Indication (SNI)

This RFC 6066 extension defines a mechanism for a client to tell a TLS server the name of the server it wants to contact. This is a useful extension for many hosting environments where multiple virtual servers are run on single IP address.

Recommendation: Unless it is known that a DTLS client does not interact with a server in a hosting environment that requires such an extension we advice to offer support for the SNI extension in this profile.

18. Maximum Fragment Length Negotiation

This RFC 6066 extension lowers the maximum fragment length support needed for the Record Layer from 2^{14} bytes to 2^9 bytes.

This is a very useful extension that allows the client to indicate to the server how much maximum memory buffers it uses for incoming messages. Ultimately, the main benefit of this extension is it to allows client implementations to lower their RAM requirements since the client does not need to accept packets of large size (such as 16k packets as required by plain TLS/DTLS).

Recommendation: Client implementations must support this extension.

19. 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.

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.

20. 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 mis-attribution 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.

21. Security Considerations

This entire document is about security.

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.

22. IANA Considerations

This document includes no request to IANA.

23. Acknowledgements

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

Big thanks also to Klaus Hartke, who wrote the initial version of this document.

24. References

24.1. 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-tls-cached-info] Santesson, S. and H. Tschofenig, "Transport Layer Security (TLS) Cached Information Extension", draft-ietf-tls-cached-info-16 (work in progress), February 2014.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [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] Seggellmann, R., Tuexen, M., and M. Williams, "Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension", RFC 6520, February 2012.
- [RFC7250] 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)", RFC 7250, June 2014.
- [RFC7251] McGrew, D., Bailey, D., Campagna, M., and R. Dugal, "AES-CCM Elliptic Curve Cryptography (ECC) Cipher Suites for TLS", RFC 7251, June 2014.

24.2. 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-02 (work in progress), May 2014.
- [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.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-01 (work in progress), March 2014.

- [I-D.ietf-uta-tls-bcp]
Sheffer, Y., Holz, R., and P. Saint-Andre,
"Recommendations for Secure Use of TLS and DTLS", draft-
ietf-uta-tls-bcp-01 (work in progress), June 2014.
- [I-D.schmertmann-dice-ccm-psk-pfs]
Schmertmann, L. and C. Bormann, "ECDHE-PSK AES-CCM Cipher
Suites with Forward Secrecy for Transport Layer Security
(TLS)", draft-schmertmann-dice-ccm-psk-pfs-00 (work in
progress), February 2014.
- [IANA-TLS]
IANA, "TLS Cipher Suite Registry",
[http://www.iana.org/assignments/tls-parameters/
tls-parameters.xhtml#tls-parameters-4](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.
- [RFC5116] McGrew, D., "An Interface and Algorithms for Authenticated
Encryption", RFC 5116, January 2008.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S.,
Housley, R., and W. Polk, "Internet X.509 Public Key
Infrastructure Certificate and Certificate Revocation List
(CRL) Profile", RFC 5280, May 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.

- [RFC6655] McGrew, D. and D. Bailey, "AES-CCM Cipher Suites for Transport Layer Security (TLS)", RFC 6655, July 2012.
- [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.
- [RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, June 2014.
- [RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, May 2014.

Author's Address

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

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

DICE Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 4, 2015

S. Keoh
University of Glasgow Singapore
S. Kumar, Ed.
O. Garcia-Morchon
E. Dijk
Philips Research
A. Rahman
InterDigital
July 03, 2014

DTLS-based Multicast Security in Constrained Environments
draft-keoh-dice-multicast-security-08

Abstract

The CoAP standard is fast emerging as a key protocol in the area of resource-constrained devices. Such IP-based systems are foreseen to be used for building and lighting automation systems where devices interconnect with each other, forming, for example, low-power and lossy networks (LLNs). Both multicast and its security are key needs in these networks. This draft presents a method for securing IPv6 multicast communication based on the DTLS which is already supported for unicast communication for CoAP devices. This draft deals with the adaptation of the DTLS record layer to protect multicast group communication, assuming that all group members already have the group security association parameters in their possession. The adapted DTLS record layer provides message confidentiality, integrity and replay protection to group messages using the group keying material before sending the message via IPv6 multicast to the group.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 4, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|---|----|
| 1. Introduction | 3 |
| 1.1. Terminology | 4 |
| 1.2. Outline | 5 |
| 2. Use Cases and Requirements | 5 |
| 2.1. Group Communication Use Cases | 5 |
| 2.2. Security Requirements | 6 |
| 3. Overview of DTLS-based Secure Multicast | 8 |
| 3.1. IP Multicast | 9 |
| 3.2. Securing Multicast in Constrained Networks | 9 |
| 4. Multicast Data Security | 10 |
| 4.1. SecurityParameter derivation | 11 |
| 4.2. Record layer adaptation | 11 |
| 4.3. Sending Secure Multicast Messages | 13 |
| 4.4. Receiving Secure Multicast Messages | 14 |
| 4.5. Unicast Responses to Multicast Messages | 14 |
| 4.6. Proxy Operation | 15 |
| 5. IANA Considerations | 16 |
| 6. Security Considerations | 16 |
| 6.1. Group level security | 16 |
| 6.2. Late joiners | 16 |
| 6.3. Uniqueness of SenderIDs | 17 |
| 6.4. Reduced sequence number space | 17 |
| 7. Acknowledgements | 17 |
| 8. References | 17 |
| 8.1. Normative References | 17 |
| 8.2. Informative References | 18 |
| Appendix A. Change Log | 20 |
| Authors' Addresses | 21 |

1. Introduction

There is an increased use of wireless networks in lighting and building management systems. This is mainly driven by the fact that the independence from physical wires allows for freedom of placement, portability and for reducing the cost of installation as less cable placement and drilling are required. Consequently, there is an ever growing number of electronic devices, sensors and actuators that have become Internet connected, thus creating a trend towards the Internet-of-Things (IoT). These connected devices are equipped with communication capability that enables them to interact with each other as well as with the wider Internet services. However, the devices in such wireless networks are characterized by power constraints (as these are usually battery-operated), have limited computational resources (low CPU clock, small RAM and flash storage) and often, the communication bandwidth is limited and unreliable (e.g., IEEE 802.15.4 radio). Hence, such wireless networks are also known as Low-power and Lossy Networks (LLNs).

In addition to the usual device-to-device unicast communication that allow devices to directly interact with each other, group communication is an important feature in constrained environments. It is more effective in constrained environments to convey messages to a group of devices without requiring the sender to perform multiple time and energy consuming unicast transmissions to reach each individual group member. For example, in a building and lighting automation system, the heating, ventilation, air-conditioning and lighting devices are often grouped according to the layout of the building, and commands are issued simultaneously to a group of devices. Group communication is based on the Constrained Application Protocol (CoAP) [I-D.ietf-core-coap] sent over IP-multicast [I-D.ietf-core-groupcomm].

Currently, CoAP messages are protected using Datagram Transport Layer Security (DTLS) [RFC6347]. However, DTLS is currently used to secure a connection between two endpoints and it cannot be used to protect multicast group communication. Group communication in constrained environments is equally important and should be secured as it is also vulnerable to the usual attacks over the air (eavesdropping, tampering, message forgery, replay, etc). There have been a lot of previous efforts in IETF to standardize mechanisms to secure multicast communication such as [RFC3830], [RFC4082], [RFC3740], [RFC4046], and [RFC4535]. However, these approaches are not necessarily suitable for constrained environments which have much more limited bandwidth and resources. For example, the MIKEY Architecture [RFC3830] is mainly designed to facilitate multimedia distribution, while TESLA [RFC4082] is proposed as a protocol for broadcast authentication of the source and not for protecting the

confidentiality of multicast messages. [RFC3740] and [RFC4046] provide reference architectures for multicast security. [RFC4535] describes Group Secure Association Key Management Protocol (GSAKMP), a security framework for creating and managing cryptographic groups on a network which can be reused for key management in our context with any needed adaptation for constrained networks.

This draft describes an approach to use DTLS as mandated in CoAP unicast to also support multicast security. We will assume that all devices in the group already have a group security association parameters based on a key management mechanism which is outside the scope of this draft. This draft focuses primarily on the adaptation of the DTLS record layer to protect multicast messages to be sent to the group, and thus providing confidentiality, integrity and replay protection to the CoAP group messages.

Lastly, even though this draft is written from the perspective of securing CoAP based group communication, it is important to note that DTLS is a powerful and flexible security protocol. Thus use of DTLS-based multicast for application layer protocols other than CoAP are possible as long as they follow the approach outlined in this draft.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

This specification uses the following terminology:

- o Group Controller: The entity that is responsible for creating a multicast group and establishing security associations among authorized group members. It is also responsible for renewing/ updating the multicast group keys.
- o Sender: The Sender is an entity that sends data to the multicast group. In a 1-to-N multicast group only a single sender transmits data to the group. In an M-to-N multicast group (where M and N are not necessarily the same value), M group members are senders.
- o Listener: The entity that receives multicast messages when listening to a specific multicast IP address.
- o Security Association (SA): A set of policy and cryptographic keys that provide security services to network traffic that matches that policy [RFC3740]. A Security Association usually contains the following attributes:

- * selectors, such as source and destination identifiers.
- * cryptographic policy, such as the algorithms, modes, key lifetimes, and key lengths used for authentication or confidentiality.
- * keying material for authentication, encryption and signing.
- o Group Security Association (GSA): A bundling of security associations (SAs) that together define how a group communicates securely. [RFC3740]
- o Keying material: Data that is specified as part of the SA which is needed to establish and maintain a cryptographic security association, such as keys, key pairs, and IVs [RFC4949].

1.2. Outline

This draft is structured as follows: Section 2 motivates the proposed solution with group communication use cases in LLNs and derives a set of requirements. Section 3 provides an overview of the proposed DTLS-based multicast security assuming that all devices in the group already have a group security association parameters in their possession. In Section 4, we describe the details of the adaptation of DTLS record layer for confidentiality and integrity protection of the multicast messages. Section 6 presents the security considerations.

2. Use Cases and Requirements

This section defines the use cases for group communication in LLNs and specifies a set of security requirements for these use cases.

2.1. Group Communication Use Cases

The "Group Communication for CoAP" draft [I-D.ietf-core-groupcomm] provides the necessary background for multicast based CoAP communication in constrained environments (e.g. LLNs). and the interested reader is encouraged to first read this document to understand the non-security related details. This document also lists a few multicast group communication uses cases with detailed descriptions and some are listed here briefly:

- a. Lighting automation: enabling synchronous operation of a group of 6LoWPAN [RFC4944] [RFC6282] connected lights in a room/floor/building. This ensures that the light preset like on/off/dim-level of a large group of luminaries are changed at the same

time, hence providing a visual synchronicity of light effects to the user.

- b. Parameter update: configuration settings of a group of similar devices are updated simultaneously and efficiently.
- c. Device and Service discovery: information about the devices in the local network and their capabilities can be queried and requested using multicast, e.g. by a commissioning device. The responses are sent back in unicast.

Elaborating on one of the main use cases that this document addresses, Lighting automation, consider a building equipped with 6LoWPAN IP-connected lighting devices, switches, and 6LoWPAN border routers; the devices are organized in groups according to their physical location in the building, e.g., lighting devices and switches in a room/floor can be configured as a single multicast group. The switches are then used to automate the lighting devices in the group by sending on/off/dimming commands to all lighting devices in the group. 6LoWPAN border routers that are connected to an IPv6 network backbone (which is also multicast enabled) are used to interconnect 6LoWPANs in the building. Consequently, this would also enable multicast groups to be formed across different physical subnets (which may be individually protected with L2 security). In such a multicast group, group messages can traverse from one physical subnet to another physical subnet through a IPv6 backbone which may not be protected. Additionally, other non-lighting devices (like window blind automation) may share the physical subnet for networking.

2.2. Security Requirements

The "Miscellaneous CoAP Group Communication Topics" draft [I-D.dijk-core-groupcomm-misc] already defines a set of security requirements for CoAP group communications. We re-iterate and further describe those security requirements in this section with respect to the use cases. The security requirements are classified into those that are assumed to be fulfilled and those that need to be fulfilled by the solution in this draft.

The security requirements which are out-of-scope of this draft and assumed to be already fulfilled:

- a. Establishment of a GSA: A secure mechanism must be used to distribute keying materials, multicast security policies and security parameters to members of a multicast group. A GSA must be established by the group controller (which manages the multicast group) among the group members. The 6LoWPAN border

router, a device in the 6LoWPAN, or a remote server outside the 6LoWPAN could play the role of the group controller. However, GSA establishment is outside the scope of this draft, and it is anticipated that an activity in IETF dedicated to the design of a generic key management scheme for the LLN will include this feature preferably based on [RFC3740], [RFC4046] and [RFC4535].

- b. Multicast data security ciphersuite: All group members must use the same ciphersuite to protect the authenticity, integrity and confidentiality of multicast messages. The ciphersuite is part of the GSA. Typically authenticity is more important than confidentiality in LLNs. Therefore the proposed multicast data security protocol must support at least ciphersuites with MAC only (NULL encryption) and AEAD [RFC5116] ciphersuites. Other ciphersuites that are defined for data record security in DTLS should also be preferably supported.
- c. Forward security: Devices that leave the group should not have access to any future GSAs. This ensures that a past member device cannot continue to decrypt confidential data that is sent in the group. It also ensures that this device cannot send encrypted and/or integrity protected data after it leaves the group. The GSA update mechanism has to be defined as part of the key management scheme.
- d. Backward confidentiality: A new device joining the group should not have access to any old GSAs. This ensures that a new member device cannot decrypt data sent before it joins the group. The key management scheme should ensure that the GSA is updated to ensure backward confidentiality.

The security requirements which need to be fulfilled by the solution described in this draft:

- a. Multicast communication topology: We consider both 1-to-N (one sender with multiple listeners) and M-to-N (multiple senders with multiple listeners) communication topologies. The 1-to-N communication topology is the simplest group communication scenario that would serve the needs of a typical LLN. For example, in the simple lighting automation use case, the switch is the only entity that is responsible for sending commands to a group of lighting devices. In more advanced lighting automation use cases, a N-to-M communication topology would be required, for example if multiple sensors (presence or day-light) are responsible to trigger events to a group of lighting devices.
- b. Multicast group size: The security solutions should support the typical group sizes that "Group Communication for CoAP" draft

[I-D.ietf-core-groupcomm] intends to support. Group size is the combination of the number of Senders and Listeners in a group with possible overlap (a Sender can also be a Listener but need not be always). In LLN use cases mentioned in the document, the number of Senders (normally the controlling devices) is much smaller than the number of Listeners (the controlled devices). A security solution that supports 1 to 50 Senders would cover the group sizes required for most use cases that are relevant for this document. The total number of group devices must be in the range of 2 to 100 devices. Groups larger than these should be divided into smaller independent multicast groups such as grouping lights of a building per floor.

- c. Multicast data confidentiality: Multicast message should be encrypted, as some control commands when sent in the clear could pose unforeseen privacy risks to the users of the system.
- d. Multicast data replay protection: It must not be possible to replay a multicast message as this would disrupt the operation of the group communication.
- e. Multicast data group authentication and integrity: It is essential to ensure that a multicast message originated from a member of the group and that messages have not been tampered with by attackers who are not members. The multicast group key which is known to all group members is used to provide authenticity to the multicast messages (e.g., using a Message Authentication Code, MAC). This assumes that all other group members are trusted not to tamper with the multicast message.

3. Overview of DTLS-based Secure Multicast

The goal of this draft is to secure CoAP Group communication by extending the use of the DTLS security protocol to allow for the use of DTLS record layer with minimal adaptation. The IETF CoRE WG has selected DTLS [RFC6347] as the default must-implement security protocol for securing CoAP, therefore it is desirable that DTLS be extended to facilitate CoAP-based group communication. Reusing DTLS for different purposes while guaranteeing the required security properties can avoid the need to implement multiple security protocols and this is especially beneficial when the target deployment consists of resource-constrained embedded devices. This section first describes group communication based on IP multicast, and subsequently sketches a solution for securing group communication using DTLS.

devices using various methods defined in [I-D.vanderstok-core-dna] such as DNS-SD [RFC6763] and Resource Directory [I-D.ietf-core-resource-directory]. The group controller communicates with individual devices to add them to the new group. Additionally it distributes the GSA consisting of keying material, security policies security parameters and ciphersuites using a standardized key management for constrained networks which is outside the scope of this draft. Additional ciphersuites may need to be defined to convey the bulk cipher algorithm, MAC algorithm and key lengths within the key management protocol.

Senders in the group can encrypt and authenticate the CoAP group messages from the application using the keying material into the DTLS record. The authenticated encrypted message is passed down to the lower layer of the IPv6 protocol stack for transmission to the multicast address as depicted in Figure 2. The listeners when receiving the message, use the multicast IPv6 destination address and port (i.e., Multicast identifier) to look up the GSA needed for that group connection. The received message is then decrypted and the authenticity is verified using the keying material for that connection.

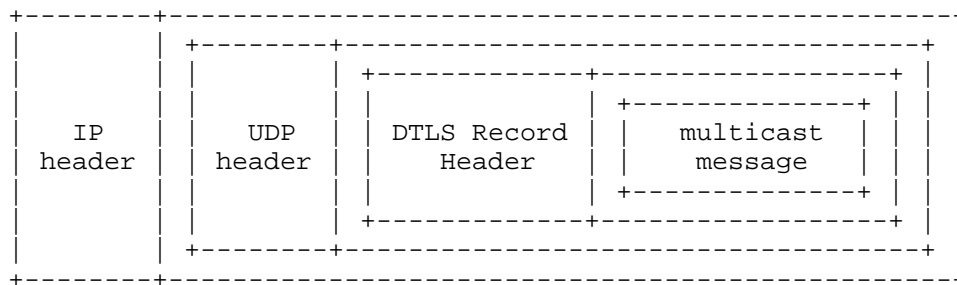


Figure 2: Sending a multicast message protected using DTLS Record Layer

4. Multicast Data Security

This section describes in detail the use of DTLS record layer to secure multicast messages. This assumes that group membership has been configured by the group controller, and all member devices in the group have the GSA.

4.1. SecurityParameter derivation

The GSA is used to derive the same "SecurityParameters" structure as defined in [RFC5246] for all devices.

The SecurityParameters.ConnectionEnd should be set to "server" for senders and "client" for listeners. The current read and write states can be derived from SecurityParameters by generating the six keying materials:

```
client write MAC key
server write MAC key
client write encryption key
server write encryption key
client write IV
server write IV
```

This requires that the client_random and server_random within the SecurityParameters are also set to the same value for all devices as part of the GSA to derive the same keying material for all devices in the group with the PRF function defined in Section 6.3 of [RFC5246]. Alternatively, the GSA could directly include the above six keying material when being configured in all group devices.

The current read and write states are instantiated for all group members based on the keying material and according to their roles: senders use "server write" parameters for the write state and listeners use "server write" parameters for the read state. Additionally each connection state contains the sequence number which is incremented for each record sent; the first record sent has the sequence number 0.

4.2. Record layer adaptation

In this section, we describe in detail the adaptation of the DTLS Record layer to enable multiple senders in the group to securely send information using a common group key, while preserving the confidentiality, integrity and freshness of the messages.

The following Figure 3 illustrates the structure of the DTLS record layer header, the epoch and seq_number are used to ensure message freshness and to detect message replays.

| | | | | | | |
|--------------|-----------------|--------|-------------|--------|------------------|-----------|
| 1 Byte | 2 Byte | 2 Byte | 6 Byte | 2 Byte | | |
| Content Type | Version Ma Mi | epoch | seq_ number | Length | Ciphertext (Enc) | MAC (Enc) |

Figure 3: The DTLS record layer header and optionally encrypted payload and MAC

The epoch is fixed by the DTLS handshake and the seq_number is initialized to 0. The seq_number is increased by one whenever a sender sends a new record message. This is the mechanism of DTLS to detect message replay. Finally, the message is protected (encrypted and authenticated with a MAC) using the session keys in the "server write" parameters.

One of the problems with supporting multiple senders is that, the seq_number used by senders need to be synchronized to avoid their reuse, otherwise packets sent by different senders may get discarded as replayed packets. Further, the bigger problem is using a single key in a multiple sender scenario leads to nonce reuse in AEAD cipher suites like AES-CCM [RFC6655] and AES-GCM [RFC5288] as defined in DTLS. Nonce reuse can completely break the security of these cipher suites.

According to the AES-CCM for TLS, Section 3 [RFC6655], the CCMNonce is a combination of a salt value and the sequence number.

```
struct {
    opaque salt[4];
    opaque nonce_explicit[8];
} CCMNonce;
```

The salt is the "client write IV" (when the client is sending) or the "server write IV" (when the server is sending) as defined in the "SecurityParameters". Further [RFC6655] requires that the value of the nonce_explicit MUST be distinct for each distinct invocation of the CCM encrypt function for any fixed key. When the nonce_explicit is equal to the sequence number of the TLS packets, the CCMNonce has the structure as below:

```

struct {
    uint32 client_write_IV; // low order 32-bits
    uint64 seq_num;         // TLS sequence number
} CCMClientNonce.

struct {
    uint32 server_write_IV; // low order 32-bits
    uint64 seq_num;         // TLS sequence number
} CCMServerNonce.

```

In DTLS, the 64-bit sequence number is the 16-bit epoch concatenated with the 48-bit `seq_number`. Therefore to prevent that the CCMNonce is reused, either all senders need to synchronize or separate non-overlapping sequence number spaces need to be created for each sender. Synchronization between senders is especially hard in constrained networks and therefore we go for the second approach of separating the sequence number spaces by embedding a unique sender identifier in the sequence number as suggested in [RFC5288].

Thus in addition to configuring each device in the group with the GSA, the controller needs to assign a unique SenderID to each device which has the sender role in the group. The size of the SenderID is 1-octet based on the requirement for the supported group size mentioned in Section 2.2. The list of SenderIDs are then distributed to all the group members by the controller.

The existing DTLS record layer header is adapted such that the 6-octet `seq_number` field is split into a 1-octet SenderID field and a 5-octet "truncated" `trunc_seq_number` field. Figure 4 illustrates the adapted DTLS record layer header.

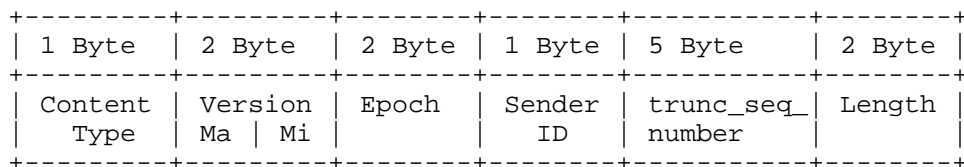


Figure 4: The adapted DTLS record layer header

4.3. Sending Secure Multicast Messages

Senders in the multicast group when sending a CoAP group message from the application, create the adapted DTLS record payload based on the "server write" parameters. Each sender in the group uses its own unique SenderID in the DTLS record layer header. It also manages its

own epoch and trunc_seq_number in the "server write" connection state; the first record sent has the trunc_seq_number 0. After creating the DTLS record, the trunc_seq_number is incremented in the "server write" connection state. The adapted DTLS record is then passed down to UDP and IPv6 layer for transmission on the multicast IPv6 destination address and port.

4.4. Receiving Secure Multicast Messages

When a listeners receives a protected multicast message from the sender, it looks up the corresponding "client read" connection state based on the multicast IP destination and port of the packet. This is fundamentally different from standard DTLS logic in that the current "client read" connection state is bound to the source IP address and port.

Listener devices in a multiple senders multicast group, need to store multiple "client read" connection states for the different senders linked to the SenderIDs. The keying material is same for all senders however the epoch and the trunc_seq_number of the last received packets needs to be kept different for different senders.

The listeners first perform a "server write" keys lookup by using the multicast IPv6 destination address and port of the packet. By knowing the keys, the listeners decrypt and check the MAC of the message. This guarantees that no one outside the group has spoofed the SenderID, as it is protected by the MAC. Subsequently, by authenticating the SenderID field, the listeners retrieve the "client read" connection state which contains the last stored epoch and trunc_seq_number of the sender, which is used to check the freshness of the message received. The listeners must ensure that the epoch is the same and trunc_seq_number in the message received is higher than the stored value, otherwise the message is discarded. Alternatively a windowing mechanism can be used to accept genuine out-of-order packets. Once the authenticity and freshness of the message have been checked, the listeners can pass the message to the higher layer protocols. The epoch and the trunc_seq_number in the corresponding "client read" connection state are updated as well.

4.5. Unicast Responses to Multicast Messages

In CoAP, responses to multicast messages are always sent back as unicast. That is, the group members that receive a multicast message may individually decide to send (or suppress) a unicast response as described in Section 2.5 of [I-D.ietf-core-groupcomm]. The unicast responses to a DTLS-based multicast message MUST be secured. Specifically, the unicast response may be sent back in a unicast DTLS message as described in Section 9.1 of [I-D.ietf-core-coap]. This

requires that a unicast DTLS session is already established between the multicast sender and the listener.

Either the multicast message sender or listener may initiate the unicast DTLS handshake to establish the DTLS session. If the DTLS handshake was initiated by the multicast message sender, it requires that the sender be aware of the membership of the multicast group. This can be accomplished, for example, as described in Section 2.6 of [I-D.ietf-core-groupcomm]. If the listener initiated the DTLS handshake, it may have done so, for example, after receiving a multicast message from a specific sender for the first time.

In the extreme scenario, a multicast sender may attempt to initiate the unicast DTLS handshake with all, or a subset of, known listeners just before or just after it sends out the DTLS-based multicast message. This may result in the multicast sender having to process unicast DTLS handshake messages from multiple multicast listeners in a short period.

For matching a CoAP response to its corresponding CoAP multicast request, the matching rules for multicast CoAP in Section 8.2 of [I-D.ietf-core-coap] are used: only the Token value MUST match. Note in particular that the matching rules for unicast DTLS of Section 9.1.2 of [I-D.ietf-core-coap] do not apply in the multicast request case.

Note: There is an obvious timing and processing load issue for the multicast sender in all scenarios where multiple DTLS sessions are established just before or just after the sender sends out the DTLS-based multicast message. In the case that the DTLS handshake initiation is left to the listeners, the processing load in the multicast sender (i.e. unicast DTLS client) is reduced somewhat by the fact that CoAP requires a randomized back-off delay before responding to a multicast request. The delay is determined by the Leisure mechanism as described in Section 8.2 of [I-D.ietf-core-coap].

4.6. Proxy Operation

CoAP allows a client to designate a (forward) proxy to process its CoAP request for both unicast and multicast scenarios as described in Section 2.10 of [I-D.ietf-core-groupcomm]. In this case, the proxy (and not the client) appears as the originating point to the destination server for the CoAP request.

As mentioned in Section 11.2 of [I-D.ietf-core-coap], proxies are by their nature men-in-the-middle and break DTLS protection of CoAP message exchanges. Therefore, in a DTLS-based multicast scenario

involving a proxy, a two-step approach is required. First, the client will send a unicast DTLS request to the proxy. The proxy will then receive and decrypt the unicast message. The proxy will then take the contents of the received message and create a new multicast message and secure it using DTLS-based multicast before sending it out to the group. For this approach to work properly, the client needs to be able to designate the proxy as an authorized sender. The mechanism for this authorization is outside the scope of this draft.

5. IANA Considerations

This memo includes no request to IANA.

6. Security Considerations

Some of the security issues that should be taken into consideration are discussed below.

6.1. Group level security

This proposal uses a single group key to protect communication within the group. This requires that all group members are trusted, for e.g. they do not forge messages as a different sender in the group. In many use case, the devices in a group belong to a common authority and are configured by a commissioner. In a professional lighting scenario, the roles of the senders and listeners are configured by the lighting commissioner and devices follow those roles.

The use of the protocol should take into consideration the risk of compromise of a group device in a deployment scenario. Therefore the group size should be limited to 100 devices unless additional source authenticity mechanisms are implemented at the application layer. Further, the damage due to a compromised key can be limited by increasing the frequency of re-keying based on the unique unicast key-pair shared by each device with the controller. Additionally the risk of compromise is reduced when deployments are in physically secured locations, like lighting inside office buildings.

6.2. Late joiners

Listeners who are late joiners to a multicast group, do not know the current epoch and `trunc_seq_number` being used by different senders. When they receive a packet from a sender with a random `trunc_seq_number` in it, it is impossible for the listener to verify if the packet is fresh and has not been replayed by an attacker. To overcome this late joiner security issue, the group controller which can act as a listener in the multicast group can maintain the epoch and `trunc_seq_number` of each sender. When late joiners send a

request to the group controller to join the multicast group, the group controller can send the list of epoch and trunc_seq_numbers as part of the GSA.

6.3. Uniqueness of SenderIDs

It is important that SenderIDs are unique to maintain the security properties of the DTLS record layer messages. However in the event that two or more senders are configured with the same SenderID, a mechanism needs to be present to avoid a security weakness and recover from the situation. One such mechanism is that all senders of the multicast group are also listeners. This allows a sender which receives a packet from a different device with its own SenderID in the DTLS header to become aware of a clash. Once aware, the sender can inform the controller on a secure channel about the clash along with the source IP address. The controller can then provide a different SenderID to either device or both.

6.4. Reduced sequence number space

The DTLS record layer seq_number is truncated from 6 octets to 5 octets. This reduction of the seq_number space should be taken into account to ensure that epoch is incremented before the trunc_seq_number wraps over. The sender or the controller can increase the epoch number by sending a ChangeCipherSpec message whenever the trunc_seq_number has been exhausted. This should be done as part of the key management mechanism which is not defined in this draft.

7. Acknowledgements

The authors greatly acknowledge discussion, comments and feedback from Dee Denteneer, Peter van der Stok, Zach Shelby, Michael StJohns and Marco Tiloca. Additionally thanks to David McGrew for suggesting options for recovering from a SenderID clash, and John Foley for the extensive review and pointing us to the AERO draft. We also appreciate prototyping and implementation efforts by Pedro Moreno Sanchez who worked as an intern at Philips Research.

8. References

8.1. Normative References

- [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-core-groupcomm]
Rahman, A. and E. Dijk, "Group Communication for CoAP",
draft-ietf-core-groupcomm-19 (work in progress), June
2014.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC5116] McGrew, D., "An Interface and Algorithms for Authenticated
Encryption", RFC 5116, January 2008.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security
(TLS) Protocol Version 1.2", RFC 5246, August 2008.
- [RFC5288] Salowey, J., Choudhury, A., and D. McGrew, "AES Galois
Counter Mode (GCM) Cipher Suites for TLS", RFC 5288,
August 2008.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer
Security Version 1.2", RFC 6347, January 2012.
- [RFC6655] McGrew, D. and D. Bailey, "AES-CCM Cipher Suites for
Transport Layer Security (TLS)", RFC 6655, July 2012.

8.2. Informative References

- [FIPS.180-2.2002]
National Institute of Standards and Technology, "Secure
Hash Standard", FIPS PUB 180-2, August 2002,
<[http://csrc.nist.gov/publications/fips/fips180-2/
fips180-2.pdf](http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf)>.
- [FIPS.197.2001]
National Institute of Standards and Technology, "Advanced
Encryption Standard (AES)", FIPS PUB 197, November 2001,
<[http://csrc.nist.gov/publications/fips/fips197/
fips-197.pdf](http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf)>.
- [I-D.dijk-core-groupcomm-misc]
Dijk, E. and A. Rahman, "Miscellaneous CoAP Group
Communication Topics", draft-dijk-core-groupcomm-misc-06
(work in progress), June 2014.
- [I-D.ietf-core-resource-directory]
Shelby, Z., Bormann, C., and S. Krco, "CoRE Resource
Directory", draft-ietf-core-resource-directory-01 (work in
progress), December 2013.

- [I-D.mcgrew-aero] McGrew, D. and J. Foley, "Authenticated Encryption with Replay protection (AERO)", draft-mcgrew-aero-01 (work in progress), February 2014.
- [I-D.vanderstok-core-dna] Stok, P., Lynn, K., and A. Brandt, "CoRE Discovery, Naming, and Addressing", draft-vanderstok-core-dna-02 (work in progress), July 2012.
- [RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.
- [RFC3740] Hardjono, T. and B. Weis, "The Multicast Group Security Architecture", RFC 3740, March 2004.
- [RFC3830] Arkko, J., Carrara, E., Lindholm, F., Naslund, M., and K. Norrman, "MIKEY: Multimedia Internet KEYing", RFC 3830, August 2004.
- [RFC4046] Baugher, M., Canetti, R., Dondeti, L., and F. Lindholm, "Multicast Security (MSEC) Group Key Management Architecture", RFC 4046, April 2005.
- [RFC4082] Perrig, A., Song, D., Canetti, R., Tygar, J., and B. Briscoe, "Timed Efficient Stream Loss-Tolerant Authentication (TESLA): Multicast Source Authentication Transform Introduction", RFC 4082, June 2005.
- [RFC4535] Harney, H., Meth, U., Colegrove, A., and G. Gross, "GSAKMP: Group Secure Association Key Management Protocol", RFC 4535, June 2006.
- [RFC4785] Blumenthal, U. and P. Goel, "Pre-Shared Key (PSK) Ciphersuites with NULL Encryption for Transport Layer Security (TLS)", RFC 4785, January 2007.
- [RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, September 2007.
- [RFC4949] Shirey, R., "Internet Security Glossary, Version 2", RFC 4949, August 2007.
- [RFC5374] Weis, B., Gross, G., and D. Ignjatic, "Multicast Extensions to the Security Architecture for the Internet Protocol", RFC 5374, November 2008.

- [RFC6282] Hui, J. and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", RFC 6282, September 2011.
- [RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, February 2013.

Appendix A. Change Log

(To be removed by RFC editor before publication.)

Changes from keoh-03 to keoh-04:

- o Added description of Proxy operation in a DTLS-based multicast scenario in Section 4.5 (Proxy Operation).
- o Corrected text in Section 2.2 (Security Requirements), item "h", to indicate that multicast source authentication is not specified in this version of the draft.
- o Clarified that draft is written primarily for securing of CoAP based group communication, but that other protocols may also be supported if they have similar characteristics. See Section 1 (Introduction).
- o Ran IETF spell checker and ID-Nits tools and corrected various issues throughout the document.
- o Various editorial updates.

Changes from keoh-04 to keoh-05:

- o In section 2.1, removed the firmware upgrade usecase and clarified the commissioning use case. The lighting use-case expanded with shared and multiple subnets issues.
- o In Section 2.2, (b) reduced the group size to 100; (h) clarified data source authenticity
- o Added new Section 6.1 (Group level security) in security considerations to make clear the risks of the single group key.

Changes from keoh-05 to keoh-06:

- o Added description of protection of unicast responses to multicast request in new Section 4.5 (Unicast Responses to Multicast Messages).

- o Clarified that CoAP may be run over either LLNs or regular networks. This also included changing the title of the I-D.
- o Various editorial updates.

Changes from keoh-06 to keoh-07:

- o Clarified that either the sender or receiver may initiate the unicast DTLS handshake (for the protected unicast response) in Section 4.5.
- o Various editorial updates.

Changes from keoh-07 to keoh-08:

- o Changed focus of usage of the DTLS-multicast solution from "control applications" to a "Lighting automation" theme.
- o Added request/response matching rules for DTLS-multicast.
- o Various editorial updates for better clarity.

Authors' Addresses

Sye Loong Keoh
University of Glasgow Singapore
Republic PolyTechnic, 9 Woodlands Ave 9
Singapore 838964
SG

Email: SyeLoong.Keoh@glasgow.ac.uk

Sandeep S. Kumar (editor)
Philips Research
High Tech Campus 34
Eindhoven 5656 AE
NL

Email: sandeep.kumar@philips.com

Oscar Garcia-Morchon
Philips Research
High Tech Campus 34
Eindhoven 5656 AE
NL

Email: oscar.garcia@philips.com

Esko Dijk
Philips Research
High Tech Campus 34
Eindhoven 5656 AE
NL

Email: esko.dijk@philips.com

Akbar Rahman
InterDigital
1000 Sherbrooke Street West
Montreal H3A 3G4
CA

Email: akbar.rahman@interdigital.com