DICE Working Group
Internet-Draft
Intended Status: Standard Track
Expires: September 11, 2014

S. Raza
SICS, Stockholm
H. Shafagh
ETH Zurich
O. Dupont
Cisco Systems, Paris

March 10, 2014

Compression of Record and Handshake Headers for Constrained Environments

draft-raza-dice-compressed-dtls-00

Abstract

This document describes header compression mechanisms for the Datagram Transport Layer Security (DTLS) [RFC6347] based on the encoding scheme standardized in [RFC6282]. The DTLS Record Header (RH), Handshake Header (HH), and optionally handshake message headers are compressed using Next Header Compression (NHC) defined in [RFC6282]. This document neither invalidates any encoding schemes proposed in 6LoWPAN [RFC6282] nor compromises the end-to-end security properties provided by DTLS. This document aims to increase the applicability of DTLS and, thus, CoAPs [draft-ietf-core-coap-18] in constrained environments.

Status of this Memo

Copyright and License Notice

Table of Contents

1 Introduction

To protect CoAP transmissions, Datagram TLS (DTLS) has been proposed as the primary security protocol. Analogous to TLS-protected HTTP (HTTPs), the DTLS-secured CoAP protocol is termed CoAPs. DTLS is a chatty protocol and requires numerous message exchanges to establish a secure session. While DTLS supports a wide range of cryptographic primitives for peer authentication and payload protection, it was originally designed for network scenarios where message length was not a critical design criterion. Therefore, it is inefficient to use the DTLS protocol, as it is, for constrained devices. To cope with constrained resources and the size limitations of IEEE 802.15.4-based networks, 6LoWPAN header compression mechanisms are defined. [RFC6282] defines how IPv6 datagrams can be routed over IEEE 802.15.4 [IEEE802.15.4]-based networks. [RFC6282] defines header compression schemes that can significantly reduce the size of IP, IP extensions, and UDP headers. It is particularly beneficial to apply the 6LoWPAN header compression mechanisms to compress other protocols having well-defined header fields, such as DTLS. This document provides header compression for the DTLS Record, Handshake, and handshake messages headers with 6LoWPAN header compression mechanisms. This enables the routing of heavy-weight IP traffic to resource-constrained [IEEE802.15.4]-based wireless network.

The DTLS header compression defined in this documents does not compromise the DTLS ability to provide end-to-end security between constrained nodes and hosts on the Internet. The security in [IEEE802.15.4]-based IP networks or what is more commonly known 6LoWPAN networks is particularly important as we connect the insecure Internet with the vulnerable wireless network. The purpose of DTLS header compression is twofold. First, achieving energy efficiency by reducing the message size, since communication requires more energy than computation. Second, avoiding 6LoWPAN fragmentation that is applied when the size of a datagram is larger than the link layer MTU. Avoiding fragmentation, whenever possible, is also important from the security point of view as the 6LoWPAN protocol is vulnerable to fragmentation attacks [WiSec13].

Generic Header Compression (GHC) [draft-bormann-6lowpan-ghc-06], analogous to NHC, is also defined to allow upper layer (UDP payload and above) header compression. 6LoWPAN-GHC is a generic compression scheme for all headers and header-like structures, and is not targeted for the DTLS protocol; also, it is generally a slightly less efficient approach. It is an alternative to the approach presented in this document and it is worth evaluating the two approaches for the DTLS Record and Handshake headers.

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].

2. Linking DTLS Header Compression with 6LoWPAN

[RFC6282] defines the general format of NHC that can be used to encode DTLD headers. In order to apply 6LoWPAN header compression mechanisms to compress headers in the UDP payload, we either require a modification in the current NHC encodings for UDP in the 6LoWPAN standard, or need to define a new NHC for UDP with different ID bits. The first solution requires modification in the current standard and hence is not a favorable solution. The second solution, that is used in this document, is an extension to the 6LoWPAN standard; a similar approach is adapted to distinguish NHC from GHC [draft-bormann-6lowpan-ghc-06]. The ID bits 11110 in the NHC for UDP, as defined in the 6LoWPAN standard, indicate that the UDP payload is not compressed. We define the ID bits 11011 in the NHC for UDP to indicate that the UDP payload is compressed with 6LoWPAN_NHC. The ID bits 11011 are currently unassigned in the 6LoWPAN standard. Figure 1 shows our proposed NHC for UDP that allows compression of UDP payload; in the case of DTLS, the UDP payload contains the NHC compressed DTLS headers.

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
| 1 | 1 | 0 | 1 | 1 | C |   P   |
+---+---+---+---+---+---+---+---+
```

Figure 1: 6LOWPAN_NHC for UDP which allows compression of UDP payload

3. LOWPAN_NHC for the Record Header

The Record protocol adds header fields of 13 bytes length to each packet that is sent throughout the lifetime of a device that uses DTLS. The header compression proposed in this section reduces the Record header length to 4 bytes (plus one byte for the NHC). In contrary to the handshake header and messages, the Record header remains un-encrypted in all cases. Thus it can always be compressed using the mechanism explained in this section.

In order to provide header compression for the Record and Handshake headers, this document discusses two cases. In the first case, the Record header fragment field contains a handshake message; the next section defines header compression regarding this case. In the second case, the fragment field in the Record header is not a handshake message, it is mostly application data, or could be a DTLS alert message or ChangeCipherSpec. Figure 2 shows 6LoWPAN_NHC encoding for the Record header (LOWPAN_NHC_R).

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
| 1 | 0 | 0 | 1 | V | EC|   SN  |
+---+---+---+---+---+---+---+---+
```

Figure 2: Proposed LOWPAN NHC encoding for the DTLS Record header

The encoded bits have the following functions:

- o The first four bits in the NHC represent the NHC ID we define for the Record header. These are set to 1001.

- o Version (V): If 0, the version is the DTLS latest version which is 1.2, and the field is omitted. If 1, the version field is carried inline.

- o Epoch (EC): If 0, an 8 bit epoch is used and the left most 8 bits are omitted. If 1, all 16 bits of the epoch are carried inline. In most cases the actual epoch is either 0 or 1. Therefore, an 8 bit epoch is used most of the time, allowing for a higher space.

- o Sequence Number (SN): The sequence number consists of 48 bits, of which some are leading zeros. If SN is set to 00, a 16 bit sequence number is used and the left most 32 bits are omitted. If 01, a 24 bit sequence number is used and the left most 24 bits are omitted. If 10, a 32 bit sequence number is used and the left most 16 bits are omitted. If 11, all 48 bits of the sequence number are carried inline. The SN field in the Record header contains a value 1 for the first packet sent, and it is incremented sequentially for the subsequent packets. Note that by using 16-bit sequence number we do not limit the size of sequence number to 2^(16-1), but propose to use 16 bits for the sequence number prior to the transmission of the 2^16th packet on a DTLS connection. From the 2^16 to 2^(24-1) we propose to use 24-bit sequence numbers. Follow the same procedure for the 32-bit sequence numbers as well. However, the sender and the receiver sequence-number-counters must be reset prior to sending the 2^48th packet.

In the Record header, content_type field is always carried inline. The length field in the Record header is omitted as we expect only one DTLS record per UDP packet in constrained environments. While a source device inside a 6LoWPAN sends one DTLS record per UDP packet, a typical destination device on the conventional Internet side may send multiple DTLS records in a single UDP packet. However, as the 6BR performs the compression/decompression of incoming packets, there is the possibility to enforce one DTLS record per UDP packet before routing these packets in 6LoWPAN networks. The length field can be deduced from the lower layers: either from the 6LoWPAN header or the IEEE 802.15.4 header. Figure 3 shows a sample NHC compressed IP/UDP packet secured with the Record protocol.

```
|   octet 1      |   octet 2      |    octet 3     |     octet 4    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         LOWPAN_IPHC             |  Hop Limit     | Source Address|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Source Address|      Destination Address         | LOWPAN_NHC_UDP|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|S Port |D Port |            Checksum              | LOWPAN_NHC_R   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Content Type  |      Epoch     |        Sequence Number            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                                   |
+       Initialization Vector (IV) [16 bytes for AES]               +
|                                                                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Application Data Fragment (Variable Size)                  |
+                                +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                |                                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                                  +
|                                                                   |
+         MAC (Variable Size)    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                |    padding     |Padding Length  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: A sample NHC compressed IP/UDP packet containing an application data such as a CoAP message.

4. LOWPAN_NHC for the Record Plus Handshake Headers

In the case where the Record header fragment field contains a handshake message, we compress both the Record header and the Handshake header using a single encoding byte and define 6LoWPAN_NHC for Record+Handshake (6LoWPAN_NHC_RH). The Handshake protocol requires 12 bytes of the handshake header. Using the proposed 6LoWPAN_NHC_RH the handshake header length is reduced to 3 bytes. Figure 4 shows 6LoWPAN NHC encoding for the Record+Handshake headers.

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
| 1 | 0 | 0 | 1 | V |EC |SN | F |
+---+---+---+---+---+---+---+---+
```

Figure 4: LOWPAN_NHC encoding for the DTLS Record plus Handshake headers

The encoded bits have the following functions:

- o The first four bits represent the ID field that is used to distinguish 6LoWPAN_NHC_RH from other encodings, and to comply with 6LoWPAN_NHC encoding scheme. In case of 6LoWPAN_NHC_RHS we set the ID bits to 1000.

- o The Version (V) and Epoch (EC) are encoded using the same scheme presented in Section 3.

- o If SN is set to 0, a 16 bit sequence number is used and the left most 32 bits are omitted. If 1, all 48 bits of the sequence number are carried inline.

- o Fragment (F): If 0, the handshake message is not fragmented and the fields fragment_offset and fragment_length are omitted. This is the common case, which occurs when a handshake message is not larger than the maximum record size. If 1, the fields fragment_offset and fragment_length are carried inline.

In contrary to the scheme defined in Section 3, the content_type field is always omitted as it is obvious based on the ID bits that the content type is the Handshake protocol. The message_type and message_sequence fields of the Handshake header are always carried inline. The length field in the Handshake headers is always omitted as it can be deduced from the lower layers: either from the 6LoWPAN header or the IEEE 802.15.4 header. We have to un-compress layer-wise from lower to higher layers until the UDP header is uncompressed. Then the length of the UDP payload is known and the DTLS payload length can be calculated.

With this combined encoding scheme the 25 bytes of Record plus Handshake headers are bring down to 6 bytes (plus one additional byte for the 6LoWPAN_NHC_RH). Considering that a handshake process consists of 10 messages, sending 18 less bytes for each message is a very significant saving. This contributes to the feasibility of using the chatty handshake protocol for constrained nodes.

5. LOWPAN_NHC for the Handshake Messages

The Handshake protocol consists of 10 messages, all having well-defined headers. We can compress some of the handshake messages. Two of the handshake messages with most number of header fields are ClientHello and ServerHello. Using the 6LoWPAN_NHC for the ClientHello message (6LoWPAN_NHC_CH) defined in this document, we can omit all ClientHello fields except the random field. The minimum possible size of a ClientHello message without the random field is 10 bytes: version (2), session_id length(1), cookie length (1), cipher_suites length (2), cipher_suites (2), compression_methods length (1), compression_methods (1). By appling 6LoWPAN_NHC_CH the minimum possible size of a ClientHello message without a random field is 1 byte that is used to encode 6LoWPAN_NHC_CH. This is the common case when DTLS is used to secure CoAP messages. Figure 5 depicts the NHC encoding for the ClientHello message.

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
| 1 | 0 | 1 | 0 | SI| C |CS |CM |
+---+---+---+---+---+---+---+---+
```

Figure 5: LOWPAN_NHC encoding for the DTLS ClientHello Message

The function of each compressed header field is described below:

o The first four bits in the 6LoWPAN_NHC_CH represent the ID field which are set to 1010.

o Session ID (SI) and Cookie (C): If 0, the session_id and/or cookie fields are not available and these fields and 8 bits of the prefixed length fields are omitted. In the (D)TLS protocol, session_id is empty if no session is available, or if the client wishes to generate new security parameters. The ClientHello message uses session_id only if the DTLS client wants to resume the old session. If SI or C is set to 1, the session_id and/or cookie fields are carried inline.

o Cipher Suites (CS): If 0, the default (mandatory) cipher suite for CoAP that supports automatic key management is used and this field and the prefixed 16 bits length field are omitted. In the current CoAP draft, TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 is a mandatory cipher suite. If CS is set to 1, the cipher_suites field is carried inline.

o Compression Methods (CM): If 0, the default compression method, i.e., COMPRESSION_NULL is used and this field and the prefixed 8 bits length field are omitted. If CM is set to 1, the compression_methods field is carried inline.

The random field in the ClientHello is always carried inline whereas the version field is always omitted. The version contains the same value as in the DTLS Record header. In case of TLS/SSL the version field was defined to let a TLS client specify an older version to be compatible with an SSL client, which is rarely used in practice. All current versions of web browsers use the same TLS version in Record and ClientHello. DTLS 1.2 (adapted from TLS 1.2) mentions that the client sends its latest supported version in the ClientHello message. All DTLS versions (1.0 and 1.2) have compatible ClientHello messages. If the server does not support this version, then the ServerHello message contains its supported version. If the client is not capable of handling server's version, it terminates the connection with a protocol version alert.

Figure 6 shows a sample compressed IP/UDP datagram that contains a ClientHello.

```
|   octet 1       |   octet 2       |    octet 3      |    octet 4     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         LOWPAN_IPHC               | Hop Limit       | Source Address|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Source Address|      Destination Address            | LOWPAN_NHC_UDP|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|S Port |D Port |             Checksum                | LOWPAN_NHC_RHS|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Epoch     |          Sequence Number            | Message Type  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Message Sequence            | LOWPAN_NHC_C    |               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+               +
|                                                                      |
+            Client Random (32 bytes)                                  +
|                                                                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 6: A sample NHC compressed IP/UDP packet containing the ClientHello message.

This document also proposes 6LoWPAN_NHC for the ServerHello message (LOWPAN_NHC_SH). ServerHello is very similar to ClientHello except that the length of the cipher_suites and compression_methods fields are fixed to 16 and 8 bits, respectively. Figure 7 shows the 6LoWPAN-NHC encoding for the ServerHello message.

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
| 1 | 0 | 1 | 1 | V |SI |CS |CM |
+---+---+---+---+---+---+---+---+
```

Figure 7: LOWPAN_NHC encoding for the DTLS ServerHello Message

The function of each compressed header field is described below:

o The first four bits in the 6LoWPAN_NHC_SH represent the ID field set to 1011.

o Version (V): In order to avoid version negotiation in the initial handshake, the DTLS 1.2 standard suggests that the server implementation should use DTLS version 1.0. If V is set to 0, the version is DTLS 1.0 and the version field is omitted. However the DTLS 1.2 clients must not assume that the server does not support higher versions or it will eventually negotiate DTLS 1.0 rather than DTLS 1.2. If V is set to 1, the version field is carried inline.

o Session ID (SI), Cipher Suite (CS), and Compression Method (CM) are encoded in a similar fashion as discussed above for the ClientHello message. In order to not compromise security the random field in the ServerHello, like in the ClientHello message, is always carried inline.

6. Summary of DTLS header sizes with and without Compression

DTLS Header	Without Compression [bytes]	With Compression [bytes]
Record	13	4* or 5
Handshake	12	3
ClientHello	10**	1
ServerHello	6**	1

* For Record plus handshake case (Section 4) the size is 4.
** Without the random field

Table 1: With the header compression defined in this document we can clearly reduce significant communication overhead in resource-constrained networks.

7. Implementation Considerations

We provide an open source implementation of the proposed compression scheme in the Contiki operating system. The implementation is released under BSD license and can be obtained at the following URI: http://www.shahidraza.info/resources/CoAP-DTLS.zip. We also evaluate the compressed DTLS and the details are published in Lithe [Lithe13].

8. Security Considerations

The compression scheme proposed in this document does not compromise any of the security provided by the DTLS Record header and the Handshake header. In particular, the SN field is compressed in an on-demand fashion, as described in Section 3. In order to overcome replay attacks, it is recommended that the communication end-points re-establish a connection using handshake before the sequence number overflows. However, in constrained environments, different implementations can decide the overflow size; 2^16, 2^24, 2^32, or 2^48. This leads to a trade-off between the overhead incurred by establishing a new secure connection (i.e. a re-handshake) and by sending more bits of sequence number. The random number field, Initialization Vector (IV), and Message Authentication Code (MAC) are also not compressed to take full advantage of DTLS security.

9. IANA Considerations

[RFC6282] creates a new IANA registry for the LOWPAN_NHC header type. This document requests the assignment of following contents:

11011XXX: The 6LOWPAN_NHC encoding for the UDP header where the UDP is compressed with LOWPAN_NHC.

1000XXXX: The 6LOWPAN_NHC encoding for the Record plus Handshake headers (LOWPAN_NHC_RH).

1001XXXX: The 6LOWPAN_NHC encoding for the Record header (LOWPAN_NHC_R).

1010XXXX: The 6LOWPAN_NHC encoding for the DTLS ClientHello message (LOWPAN_NHC_CH)

1011XXXX: The 6LOWPAN_NHC encoding for the DTLS ServerHello message (LOWPAN_NHC_SH)

The Capital letter X in bit positions represent class-specific bit assignments as defined in Section 3, 4, and 5.

10. Acknowledgements

The work is funded by CALIPSO, Connect All IP-based Smart Objects, funded by the European Commission under FP7 with contract number FP7-ICT-2011.1.3-288879.

11. References

11.1. Normative References

[KEYWORDS] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC6282] Hui, J., Ed., and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", RFC 6282, September 2011.

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

[RFC4303] J. Hui, P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", RFC 6282, September 2011

11.2. Informative References

[WiSec13] R. Hummen, J. Hiller, H. Wirtz, M. Henze, H. Shafagh, and K. Wehrle, "6LoWPAN fragmentation attacks and mitigation mechanisms," in Proceedings of the 6th ACM Conference on Security and Privacy in Wireless and Mobile Networks, Apr. 2013, Budapest, Hungry.

[Lithe13] S. Raza, H. Shafagh, K. Hewage, R. Hummen, Thiemo Voigt, "Lithe: Lightweight Secure CoAP for the Internet of Things". IEEE Sensors Journal, 13(10), 3711-3720, October 2013.

Authors' Addresses

Shahid Raza
SICS Swedish ICT AB (SICS)
Isafjordsgatan 22, 16440 Kista
SWEDEN

Phone: +46-(0)768831797
EMail: shahid@sics.se

Hossein Shafagh
ETH Zurich
Universitatstrasse 6, CH-8092 Zurich
SWITZERLAND

Phone: +41 44 63 26136
EMail: shafagh@ethz.ch

Olivier Dupont
Cisco
Cisco Systems, Paris
FRANCE

Phone: +33 158 043 480
Email: odupont@cisco.com