ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 12, December 2014

Secure CoAP Using Enhanced DTLS for Internet of Things

Ajit A.Chavan, Mininath K. Nighot

Department of Computer Engineering, KJ College of Engineering and Management Research, Pune, India.

Assistant Professor, Department of Computer Engineering, KJ College of Engineering and Management Research, Pune, India.

ABSTRACT: The Internet of Things (IoT) is next generation technology that is intended to improve and optimize daily life by operating intelligent sensors and smart objects together. At application layer, communication of resource-constrained devices is expected to use constrained application protocol (CoAP).Communication security is an important aspect of IoT environment. However closed source security solutions do not help in formulating security in IoT so that devices can communicate securely with each other. To protect the transmission of confidential information secure CoAP uses datagram transport layer security (DTLS) as the security protocol for communication and authentication of communicating devices. DTLS was initially designed for powerful devices that are connected through reliable and high bandwidth link. This paper proposes a collaboration of DTLS and CoAP for IoT. Additionally proposed DTLS header compression scheme that helps to reduce packet size, energy consumption and avoids fragmentation by complying the 6LoWPAN standards. Also proposed DTLS header compression scheme does not compromises the point-to-point security provided by DTLS. Since DTLS has chosen as security protocol underneath the CoAP, enhancement to the existing DTLS also provided by introducing the use of raw public key in DTLS.

KEYWORDS: CoAP, DTLS,Generic Header Compression,Next Header Compression.

I. INTRODUCTION

IP-connected smart devices are becoming part of the Internet hence forming the Internet of Things (IoT). TCP performance is inefficient in wireless networks, due to its congestion control algorithm and it is not working well with the low-power radios and lossy links found in sensor networks. Therefore, the connectionless UDP is mostly used in the IoT. Further, HTTP, which is primarily designed to run over TCP, is inefficient in lossy and constrained environments. CoAP is designed to meet specific requirements such as simplicity, low overhead, and multicast support in resource-constrained environments. Security is particularly important for the Things as they are connected to the untrusted Internet. Compressed IPsec can be used to secure the communication between nodes in 6LoWPAN networks and hosts in the Internet [4]. Next Header Compression (NHC) encodings has been defined to compress the Authentication Header (AH) and Encapsulating Security Payload (ESP) extension headers.

Jorge et al. [5] extend our solution and include IPsec in tunnel mode. They implement and evaluate their proposal in TinyOS. IPsec security services are shared among all applications running on a particular machine. Even though our 6LoWPAN compressed IPsec can be used to provide lightweight E2E security at the network layer, it is not primarily designed for web protocols such as HTTP or CoAP. For web protocols TLS or DTLS are common security solutions. TLS works over TCP, whereas in 6LoWPAN networks UDP is preferred. Brachmann et al. [6] propose TLS-DTLS mapping to secure the IoT. However, this requires the presence of a trusted 6BR and E2E security breaks at the 6BR. Kothmayr et al. [7] investigate the use of DTLS in 6LoWPANs with a Trusted Platform Module (TPM) to get hardware support for the RSA algorithm. However, they have used DTLS as it is without using any compression method which would shorten the lifetime of the entire network due to the redundant bits in DTLS messages. Granjal et al. [8] evaluate the use of DTLS as it is with CoAP for secure communication. They note that payload space scarcity would be problematic with applications that require larger payloads. As an alternative, they suggest to employ security at other layers such as compressed form of IPsec. In a recent work, Keoh et al. [9] have discussed the implications of securing the IP-connected IoT with DTLS and propose an architecture for secure network access and management of unicast and multicast keys with extended DTLS.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 12, December 2014

CoAP proposes to use Datagram Transport Layer Security (DTLS) as the security protocol for automatic key management and for data encryption and integrity protection, as well as for authentication[1]. CoAP with DTLS support is termed as secure CoAP (CoAPs). DTLS is a chatty protocol and requires numerous message exchanges to establish a secure session. DTLS 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 IoT devices. To match with constrained resources and the size limitations of IEEE 802.15.4-based networks, 6LoWPAN header compression mechanisms are defined. The 6LoWPAN standard already defines the header compression format for the IP header, IP extension headers, and the UDP header. It is particularly beneficial to apply the 6LoWPAN header compression mechanisms to compress DTLS header. In this paper we provide a secure CoAP by compressing the DTLS protocol with 6LoWPAN header compression mechanisms. This mechanism serves two purposes first it achieves 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 [4].

II. BACKGROUND

In this section, we highlight the protocols and technology involved in the development of the secure CoAPs, the HTTPs variant for the IoT.

A. *CoAP*

CoAP is a web protocol that runs over the unreliable UDP protocol and is designed primarily for the IoT. CoAP is a variant of the most used synchronous web protocol, HTTP ,and is tailored for constrained devices and machine-to-machine communication. However, while CoAP provides a REST interface similar to HTTP, it focuses on being more lightweight and cost-effective than its variant for today's Internet. To protect CoAP transmissions, Datagram TLS (DTLS) has been proposed as the primary security protocol [2]. Analogous to TLS protected HTTP (HTTPs), the DTLS-secured CoAP protocol is termed CoAPs. In CoAP, as with HTTP, the Universal Resource Identifier (URI) is used to access the resources on a given host. CoAP is a relatively simple request and response protocol providing both reliable and unreliable forms of communication. For the CoAP protocol, the "coap" URI scheme will be used. A web resource on an IoT device can then be accessed securely via CoAPs protocol as:

coaps://myIPv6Address:port/MyResource

A CoAP-enabled device may be acting in a client role, a server role, or both, or sending non conformable messages without response. The reasons that a new protocol is defined for constrained IP networks, instead of simply reusing HTTP, is to greatly reduce overhead in implementation complexity (code size) and to reduce the bandwidth requirements. Such data reduction also helps to increase reliability (by reducing link layer fragmentation) and reduce latency in typical low-power lossy wireless networks, such as IEEE 802.15.4.

B. DTLS

DTLS is arguably the most suited single security protocol for providing channel security, mainly because it is a rather complete security protocol that can perform authentication, key exchange, and protecting application data with the negotiated keying material and algorithms. Using DTLS as the sole security suite for IoT, the following security protection can be achieved.

- *Network Access:*-DTLS as an authentication protocol can be used to authenticate new devices joining the network either using the PSK mode, raw public key, or public key certificate. The result of a successful DTLS handshake creates a secure channel between the new device and the authorizing entity (e.g., the 6LBR) . This secure channel enables the authorizing entity to distribute the L2 key securely to the joining device based on rules which have been configured by the network owner. If the new device and the authorizing entity are one-hop at the MAC layer, then the DTLS handshake messages are not dropped by the MAC layer. However, if new device and the authorizing entity are multihop at the MAC layer, the DTLS messages are dropped at the MAC layer since they are not yet protected with the L2 key.
- *Secure Communication channel:*-a DTLS end-to-end session can be established between two communicating devices, one inside the 6LoWPAN and the other outside, to securely transport application data (CoAP messages).

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 12, December 2014

The application data is protected by the DTLS record layer, i.e., authenticated and encrypted with a fresh and unique session key. DTLS consists of two layers: the lower layer contains the Record protocol and the upper layer contains either of the three protocols namely Handshake, Alert, and ChangeCipherSpec, or application data. The Record protocol is a carrier for the upper layer protocols. The Record header contains among others content type and fragment fields. Based on the value in the content type, the fragment field contains either the Handshake protocol, Alert protocol, ChangeCipherSpec protocol, or application data. The Record header is primarily responsible to cryptographically protect the upper layer protocols or application data once the handshake process is completed. The Record protocols protection includes confidentiality, integrity protection and authenticity. Handshake protocol is a complex chatty process and contains numerous message exchanges in an asynchronous fashion. The handshake messages, usually organized in flights, are used to negotiate security keys, cipher suites and compression methods. The full handshake process is shown in figure 1.

- *Key Management:*-As DTLS has the capability of renewing session keys, this mechanism can be utilized to support key management in a 6LoWPAN network. During the Network Access phase, the 6LBR distributes an L2 key as part of the network access authentication procedure. It is thus possible to reuse the same channel to facilitate key management, thus enabling the L2 key to be updated by the 6LBR when necessary.

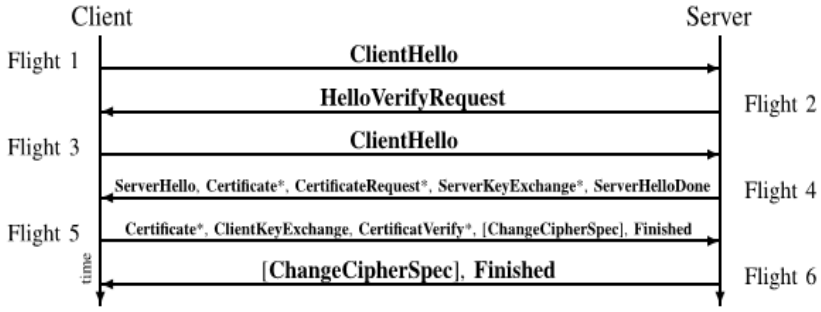

Fig.1. Full DTLS Handshake Process

C.6LoWPAN

The 6LoWPAN standard defines header compression and fragmentation mechanisms of IPv6 datagrams within IPv6-connected WSNs, also called 6LoWPAN networks. The compression mechanism consists of IP Header Compression (IPHC) and Next Header Compression (NHC). The IPHC encodings can compress the IPv6 header length to 2 bytes for a single hop network and 7 bytes in a multi-hop case (1-byte IPHC, 1-byte dispatch, 1-byte Hop Limit, 2-byte Source Address, and 2-byte Destination Address). Among other encoding bits in the IPHC is the NH bit that, when set, indicates the next header is compressed using NHC. The NHC is used to encode the IPv6 extension headers and UDP header. The size of NHC encodings is a multiple of octets (mostly one octet) which contain variable length ID bits and the encoding bits for a specific header. It is therefore worth extending the 6LoWPAN header compression mechanisms to compress these protocol headers. The 6LoWPAN standard-defined NHC encoding can be used to compress headers up to UDP, but not the upper layers. A new NHC is needed because there is no NH bit in the NHC for UDP which indicates that the UDP payload is also compressed. In Section IV, we provide 6LoWPANDTLS integration and 6LoWPAN NHCs to compress DTLS.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 12, December 2014

III. SYSTEM OVERVIEW

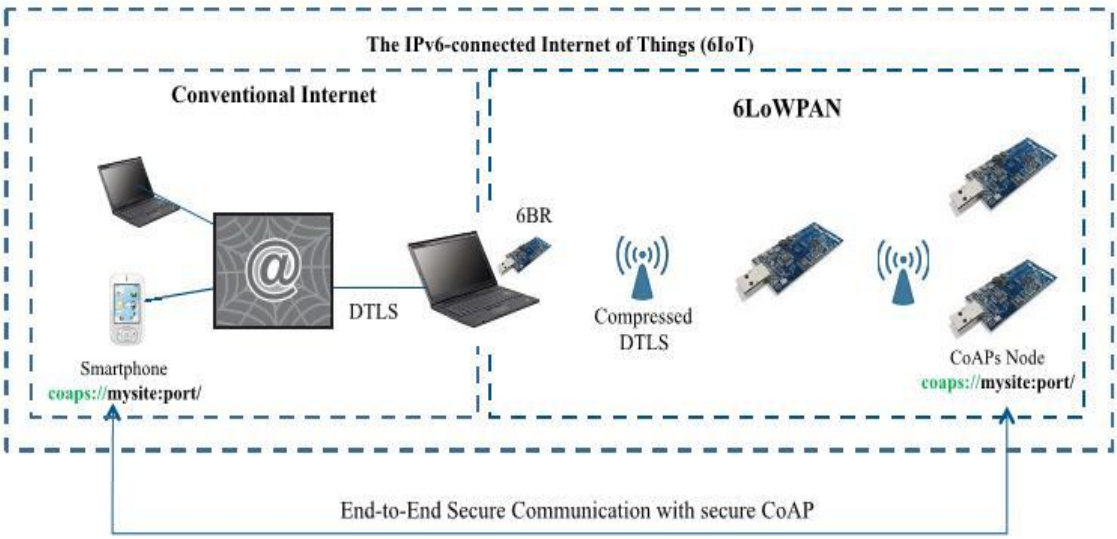

Fig.2. An IoT that uses CoAPs to secure communication between sensor nodes in 6LoWPANs and hosts in the Internet.

Our compressed DTLS maintains true End-to-End (E2E) security between Lithe enabled hosts in 6LoWPAN networks and typical Internet hosts that use uncompressed CoAPs. Figure 2 shows a typical IoT setup, where a 6LoWPAN network consisting of CoAPs enabled nodes is connected through a 6LoWPAN Border Router (6BR) with the Internet. As depicted in Figure 2, the header compression is applied within the 6LoWPAN network only, i.e., between constrained nodes and the 6LoWPAN border Router (6BR). A 6BR is used between 6LoWPAN networks and the Internet to compress/decompress or/and fragment/reassemble messages before forwarding between the two realms. In this IoT setup, the CoAPs enabled devices can securely communicate with internet hosts, such as standard computers, smartphones, etc., which support the CoAPs protocol. In order to adapt chatty security protocols, such as DTLS, for the resource-constrained IoT devices, it is beneficial to apply 6LoWPAN header compression mechanisms to these protocols as well.

IV. DTLS COMPRESSION

In this section, in addition to describing 6LoWPAN header compression for DTLS, we detail how our compressed DTLS can be linked to 6LoWPAN in a compliant way.

A. *DTLS-6LoWPAN Integration*

The proposed scheme in this paper is an extension to the 6LoWPAN standard; a similar approach is adapted to distinguish NHC from GHC. 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 ID bits 11011 to indicate that the UDP payload is compressed with 6LoWPAN-NHC. The ID bits 11011 are currently unassigned in the 6LoWPAN standard. Figure 3 shows our proposed NHC for UDP that allows compression of UDP payload; in our case, the UDP payload contains the 6LoWPAN-NHC compressed DTLS headers.

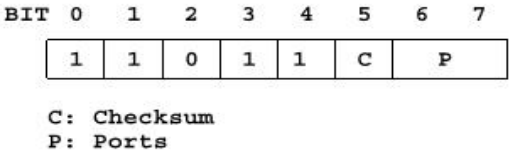

Fig.3. Our proposed 6LoWPAN-NHC for UDP, where ID bits 11011 indicate that the UDP payload is compressed.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 12, December 2014

B. *6LoWPAN-NHC for the Record and Handshake Headers*

The Record protocol adds 13 bytes long header fields to each packet that is sent throughout the lifetime of a device that uses DTLS. The handshake protocol, on the other hand, adds 12 bytes of header to handshake messages. We propose 6LoWPANNHC for compressing the Record and Handshake headers, and reduce the header length to 5 and 3 bytes, respectively. In the first 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 we define 6LoWPAN-NHC for Record+Handshake (6LoWPAN-NHC-RHS). In the second case, we define 6LoWPAN-NHC for the Record header (6LoWPAN-NHC-R) where the fragment field in the Record header is application data and not a Handshake message.Figure 4 shows 6LoWPANNHC encodings for the Record+Handshake header and for the Record header. The encoded bits have the following functions: The first four bits represent the ID field that is used to distinguish 6LoWPAN-NHC-RHS from other encodings, and to comply with 6LoWPAN-NHC encoding scheme. In case of 6LoWPAN-NHC-RHS we set the ID bits to 1000, and in case of 6LoWPAN-NHC-R we set the ID bits to 1001.

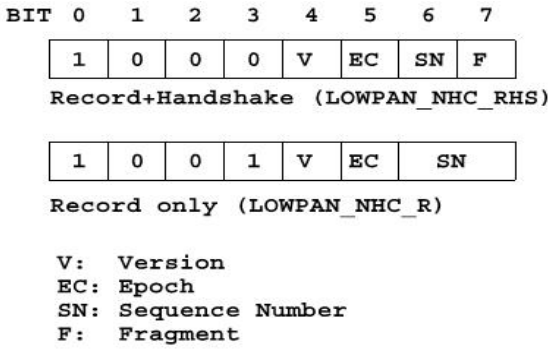

Fig. 4. Record and the Handshake header.

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.

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 a higher space.

Sequence Number (SN): The sequence number consists of 48 bits of which some are leading zeros. 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. In case of 6LoWPAN-NHC-R, as shown in Figure 4, we use two bits for SN and can more efficiently compress the sequence number field. Here if SN is set to 00, a 16 bit sequence number is used and the left most 32 bits are omitted. If 01, a 32 bit sequence number is used and the left most 16 bits are omitted. If 10, a 24 bit sequence number is used and the left most 24 bits are omitted. If 11, all 48 bits of the sequence number are carried inline.

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 the handshake message is not larger than the maximum record size. If 1, the fields fragment offset and fragment length are carried inline.

C. *6LoWPAN-NHC for ClientHello*

During the handshake process the ClientHello message is sent twice, the first time without cookie and the second time with the servers cookie.Figure 5 shows 6LoWPAN-NHC encoding for the ClientHello message.

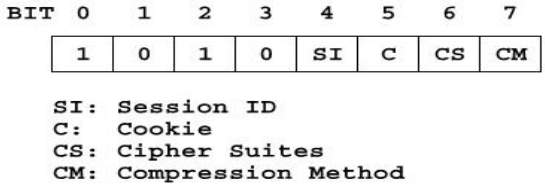

Fig. 5. Client Hello message.

10.15680/ijircce.2014.0212069

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 12, December 2014

The first four bits in the 6LoWPAN-NHC-CH represent the ID field which are set to 1010.
Session ID (SI): If 0, the session id is not available and this field and 8 bits of the prefixed length field 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. The actual session id field in the ClientHello contains 0 to 32 bytes. However, it is always prefixed with an 8 bit field that contains the size of the session id. If SI is set 1, the session id field is carried inline.

Cookie (C): If 0, the cookie field is not available and this field and its prefixed 8 bits length field are omitted. The actual cookie field in the ClientHello contains 0 to 255 bytes.However, it always has an 8 bits length field that contains the size of the cookie. If C is set 1, the cookie field is carried in line.

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. The actual cipher suites field contains 2 to 2^{16-1} bytes and is always prefixed with a 16 bits field that contains the size of the cipher suites. If CS is set 1, the cipher suites field is carried inline.

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. The actual compression methods field contains 1 to 2^{8-1} bytes. It is always prefixed with an 8 bits field that contains the size of the compression methods. If CM is set 1, the compression methods field is carried inline.

Figure 6 shows an uncompressed IP/UDP datagram that contains a ClientHello. A 6LoWPAN compressed IP/UDP datagram, with our proposed compressed DTLS, containing the ClientHello message is depicted in Figure 7. After applying IPHC and 6LoWPAN-NHC header compression, the datagram size is significantly reduced.

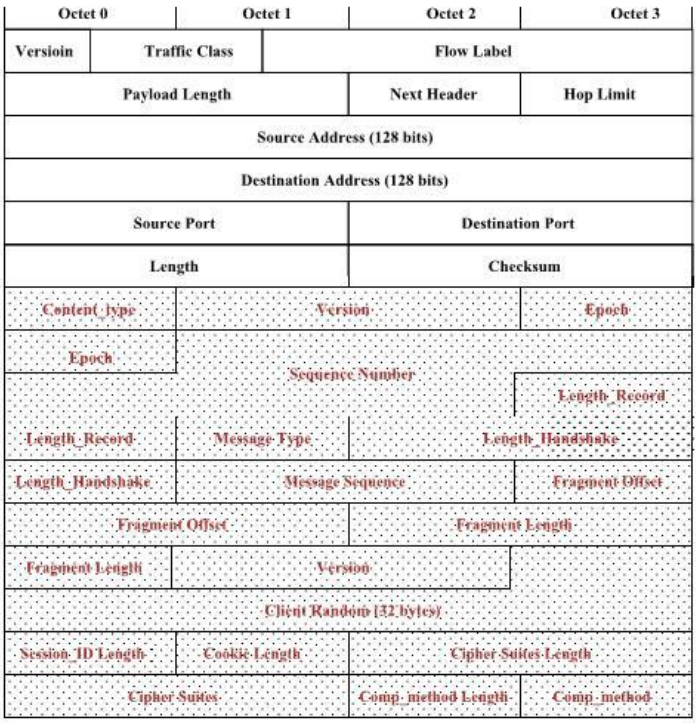

Fig. 6. An uncompressed full IP/UDP datagram containing a DTLS ClientHello Message.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 12, December 2014

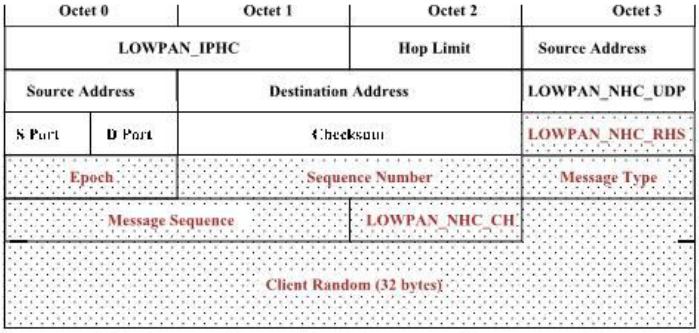

Fig. 7. A 6LoWPANcompressedfull IP/UDP datagram containing a DTLS ClientHello Message.

D. *6LoWPAN-NHC for ServerHello*

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 5c shows the 6LoWPAN-NHC encoding for the ServerHello message. The function of each compressed header field is described below: The first four bits in the 6LoWPANNHCSH represent the ID field set to 1011.

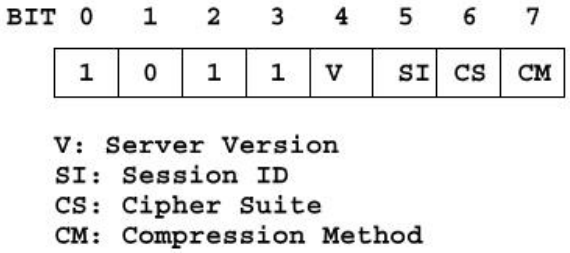

Fig. 8. ServerHello message.

*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.

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

E. *6LoWPAN-NHC for other Handshake Messages*

The remaining mandatory handshake messages ServerHelloDone, ClientKeyExchange, and Finish have no fields that could be compressed, hence all fields are carried inline. The optional handshake messages Certificate that contains the chain of certificates and Certificate Verify that contains the digital signature of the handshake message are as well carried inline.

V. ENHANCEMENT

In addition to the proposed system, we are adding the new feature to improve the end to end authentication by using raw public key. The PSK mode of operation in DTLS only supports partial interoperability between IoT devices because device manufacturers would need to preshare some keying materials with each other in order to allow for their devices to securely communicate with each other. Establishing such a trust in a multivendor environment can be difficult. The other end of the spectrum is the X.509-based Public Key Infrastructure (PKIX) to enable IoT devices to authenticate each other. The use of raw public keys with DTLS has been standardized to alleviate the burden of IoT devices from storing and transmitting X.509 certificates when performing the DTLS handshake protocol. This allows the devices to be preconfigured (during the manufacturing time) with public keys of dedicated servers that the device needs to communicate with, as well as a public key for the device itself.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 12, December 2014

When devices execute the DTLS handshake protocol using raw public keys, the newly defined TLS extensions of client certificate type and server certificate type must be used.The raw public keys are encapsulated in a SubjectPublicKeyInfo structure which only contains the raw key and an algorithm identifier. The IoT device acting as the DTLS Client initiates the DTLS handshake protocol, indicating that it is capable of processing raw public keys when sending the ClientHello message with the RawPublicKey extension. The server fulfills the clients request, indicates this via the RawPublicKey value in the server certificate type payload, and provides a raw public key into the Certificate payload back to the client. In case that client authentication is required, the Server can send a CertificateRequest message to the client, demanding the client to send its raw public key for authentication. The client, who has a raw public key preprovisioned, returns it in the Certificate payload to the Server.

VI. CONCLUSION AND FUTURE WORK

Secure CoAP is the basic need of resource constrained devices in real IoT environment. DTLS is the standard protocol to enable secure CoAP (CoAPs).This system, proposed the possibility of reducing the overhead of DTLS by means of 6LoWPAN header compression, and present the first DTLS header compression mechanism for 6LoWPAN. Compressed DTLS for CoAPs is efficient in energy consumption of nodes, memory requirement and network response. As a future work this system can be deployed in real world IoT environment containing smart sensors, constrained devices and smartphones etc with real time application. Such deployment helps to deeply study and evaluate significance of this system with confidential application.

REFERENCES

1. Z. Shelby, K. Hartke, C. Bormann, and B. Frank. (2013, May). Constrained Application Protocol (CoAP). Internet-Draft draft-ietf-corecoap-16[Online]. Available: http://datatracker.ietf.org/drafts/current/ .
2. Datagram Transport Layer Security Version 1.2, RFC Standard 6347,Jan. 2012.
3. Dunkels, B. Grönvall, and T. Voigt, "Contiki—A lightweight and flexible operating system for tiny networked sensors," inProc. 29th Annu. IEEE Int. Conf. Local Comput. Netw., Nov. 2004, pp. 455–462.
4. S. Raza, S. Duquennoy, A. Chung, D. Yazar, T. Voigt, and U. Roedig, "Securing communication in 6LoWPAN with compressed IPsec," in Proc. 7th Int. Conf. DCOSS, Barcelona, Spain, Jun. 2011, pp. 1–8.
5. J. Granjal, E. Monteiro, and J. S. Silva, "Network-layer security for the internet of things using TinyOS and BLIP," Int. J. Commun. Syst., 2012,doi: 10.1002/dac.2444.
6. M. Brachmann, S. L. Keoh, O. G. Morchon, and S. S. Kumar, "End-toend transport security in the IP-based internet of things," inProc. 21st ICCCN, Aug. 2012, pp. 1–5.
7. T. Kothmayr, C. Schmitt, W. Hu, M. Brunig, and G. Carle, "A DTLS based end-to-end security architecture for the internet of things with two-way authentication," in Proc. IEEE 37th Conf. Local Comput. Netw. Workshops, Oct. 2012, pp. 956–963.
8. J. Granjal, E. Monteiro, and J. S. Silva, "On the feasibility of secure application-layer communications on the web of things," in Proc. IEEE 37th Conf. LCN, Oct. 2012, pp. 228–231.
9. S. Keoh, S. Kumar, and O. Garcia-Morchon. (2013, Feb.). Securing the IP-Based Internet of Things with DTLS, [Online].