2015 29th International Conference on Advanced Information Networking and Applications Workshops

LESS: Lightweight Establishment of Secure Session

A Cross-Layer Approach Using CoAP and DTLS-PSK Channel Encryption

Abhijan Bhattacharyya, Tulika Bose, Soma Bandyopadhyay, Arijit Ukil, Arpan Pal
Innovation Labs
Tata Consultancy Services Ltd.
Kolkata, India
{abhijan.bhattacharyya, tulika.bose, soma.bandyopadhyay, arijit.ukil, arpan.pal}@tcs.com

***Abstract*— Secure yet lightweight protocol for communication over the Internet is a pertinent problem for constrained environments in the context of Internet of Things (IoT) / Machine to Machine (M2M) applications. This paper extends the initial approaches published in [1], [2] and presents a novel cross-layer lightweight implementation to establish a secure channel. It distributes the responsibility of communication over secure channel in between the application and transport layers. Secure session establishment is performed using a payload embedded challenge response scheme over the Constrained Application Protocol (CoAP) [3]. Record encryption mechanism of Datagram Transport Layer Security (DTLS) [4] with Pre-Shared Key (PSK) [5] is used for encrypted exchange of application layer data. The secure session credentials derived from the application layer is used for encrypted exchange over the transport layer. The solution is designed in such a way that it can easily be integrated with an existing system deploying CoAP over DTLS-PSK. The proposed method is robust under different security attacks like replay attack, DoS and chosen cipher text. The improved performance of the proposed solution is established with comparative results and analysis.**

Keywords*— *CoAP; DTLS; IoT; lightweight; M2M; pre-shared-key; secure session.

I. INTRODUCTION

IoT/M2M applications often exchange sensitive information such as physiological information of individuals, energy consumption patterns of households, location data of vehicles and so on and so forth over the Internet. Hence, means for secure communication over the Internet for IoT/M2M systems is an utmost necessity similar to the conventional sensitive Web applications. However, the usual constituents of such systems are resource-constrained (in terms of available energy, processing power, memory, communication bandwidth) sensors or sensor gateways. As a result the concerned systems are usually resource-constrained as well. So, the conventional protocols like HTTP and its secure version HTTPS [7] [8] prove sub-optimal and resource hogging for the concerned applications. Ref. [6] compares the different application layer options for 'Web-enablement' of constrained devices and establishes (CoAP) [3], being standardized by the Internet Engineering Task Force (IETF), as a promising candidate. CoAP is originally designed to use User Datagram Protocol (UDP) as the transport. As a natural consequence, the secure version of CoAP, CoAPs, is mandated to use DTLS [4] for secure communication. CoAP proposes to use three possible security modes for DTLS: (a) pre-shared key mode, (b) raw public key mode and (c) certificate mode. Pre-shared key mode is the most low-overhead option as it uses symmetric key based encryption. But it needs pre-provisioning of a long-lived key which is mutually agreed between the communicating end-points. Yet, DTLS, originally "designed to be as similar to TLS as possible" [4], is not suitable for many IoT/M2M applications. At present, efforts are being made to modify standard DTLS to support the use of DTLS in constrained environments [9]. As discussed in [9], a significant overhead of DTLS attributes to the DTLS handshakes which may even lead to undesirable uncontrolled message fragmentation at the lower layers. This is true even for the PSK mode of DTLS (DTLS-PSK).

Present paper is a salient attempt to fill in the gap, as described above, with a very lightweight yet robust practical solution. The paper builds over the initial approach published in [1], [2] and [26]. It presents an alternate lightweight cross-layer approach to provide with optimized handshakes between the end-points during secure-session establishment followed by symmetric key based channel security for the whole application layer message. The session establishment process involves both mutual authentication and agreement on the session-key pairs.

The contributions of the present work can be summarized as below:

- The proposed method partitions the responsibility of secure communication such that secure session establishment is performed at the application layer and transfer of full application layer message over secure channel is performed by the transport-layer security.
- Secure session establishment is implemented as a novel lightweight challenge-response scheme and deployed on CoAP as simply two pairs of encrypted request/responses. Thus secure session establishment is offloaded to application layer enabling the application layer with greater control while dealing with constrained environments. Also, enables CoAP with inherent capability to provide authentication and optional "object security" to application layer payload.
- Once the session is established, the session parameters are mapped to the DTLS session parameter structure without requiring any modification in the existing

session format for DTLS-PSK mode. Thus it protects the full application layer message using the encryption mechanism of DTLS-PSK which has inherent protection against replay attacks [4].

- The same standard AES-128-CCM-8 [10] scheme is used for session establishment messages (unlike AES-CBC as used in [1], [2]) and data traffic. Thus only a single encryption routine is required.
- The method maintains separation in session-key space between the server to client and the reverse channel unlike the previous works.
- The proposed lightweight solution is immune to traditional IoT security attacks like replay attack, DoS (Denial of Service) attack and chosen cipher text attack.
- The scheme can easily integrate with existing system deploying DTLS-PSK without requiring any significant additional modules and reuse existing modules with minimal additional code.

The paper is organized as follows. Section II presents a study of the related state of the art. Section III provides a brief overview of CoAP. Section IV presents the core algorithm for secure session establishment and its implementation on CoAP. Section V describes the full design incorporating the DTLS session parameters. Section VI presents a security analysis of the proposed solution. Section VII presents the results and analyses them. Finally, section VIII concludes with discussion on future directions of the research work.

II. A Study of the State-of-the-art

Quite a few lightweight secure session establishment proposals can be found in literatures. The work in [18] presents an approach for end to end security on DTLS. But it uses RSA like public key cryptosystem and does not talk about a payload embedded scheme on CoAP using symmetric key crypto system as the present work. A server initiated challenge-response based session key establishment scheme is proposed as an experimental RFC in [19]. But that is applicable for layer 2 security for WLAN environment. Another challenge response based key sharing scheme using AES-CBC encrypted exchanges for layer 2 and layer 3 security is disclosed in [20]. Similar challenge response based schemes are discussed in [21] and [22]. Ref. [22] proposes the scheme for securing Web-traffic over GSM network. However, none of the above works deal with implementation on CoAP and neither do they propose any cross layer architecture and comparative performance measurement as discussed in the present work.

As mentioned earlier, [1] and [2] seeds the present work. However, the present work enhances [1] and [2] in quite a few aspects. The modified handshakes enable AES-CCM-8 encryption which is the default choice for CoAP on DTLS-PSK [3]. Thus rather than keeping AES-CBC for session establishment and AES-CCM for DTLS encryption a single cryptographic algorithm may be used. The modified handshake also takes care of protecting the session keys from potential cracking through traffic analysis. The present scheme enables separation in the key-space between the channels unlike [1] and [2]. Most importantly, the present paper discusses in detail how the CoAP based session establishment scheme may be interfaced with DTLS-PSK session parameters to provide the final channel security for application layer message without requiring any modification in existing structure for DTLS session parameters. Thus, the present paper presents a salient, practical and complete solution.

III. About CoAP

CoAP [3] is being designed by the Constrained RESTful Environments (CoRE) working group of IETF. The aim is to enable the use of RESTful architecture, for the most constrained sensor nodes (e.g. 8-bit microcontrollers with limited RAM and ROM) and networks (e.g. 6LoWPAN). CoAP empowers these constrained nodes to perform Web transfer for IoT or M2M communication.

CoAP is 'not unlike' HTTP [12]. The basic client/server interaction model for CoAP resembles HTTP. However, in case of CoAP a machine can act both as a client and a server. Unlike HTTP the message exchanges in CoAP is asynchronous in nature over a datagram-oriented transport such as UDP. Since UDP is unreliable unlike TCP, an optional conceptual request/response layer is added along with the main CoAP messaging layer which deals with both the UDP and the asynchronous interactions (Figure 1(a)). CoAP reduces the packet overhead significantly by specifying only 4 bytes of mandatory header field (Figure 1(b)). CoAP defines similar RESTful methods like HTTP, viz., GET, PUT, POST, DELETE. CoAP also supports a list of response codes to indicate the state of execution against the request from the client.

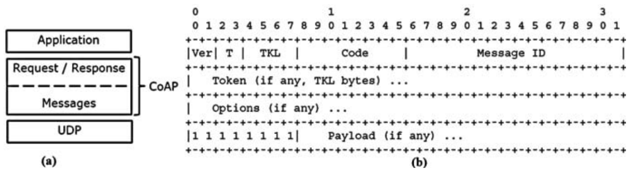


Fig. 1. (a) CoAP stack, (b) CoAP message structure.

CoAP supports four types of messages: Confirmable (CON), Non-Confirmable (NON), Acknowledgement (ACK), Reset (RST). A confirmable message ensures reliability and is replied with an ACK by the Server with the same message ID. A confirmable message is retransmitted using a default timeout and exponential back-off between retransmissions, until the recipient sends the ACK message. Non-confirmable messages are for unreliable exchanges and are not replied with any ACK. If the recipient is not able to process a Non-confirmable message it may reply with an RST message.

IV. LESS: The Algorithm and Implementation on CoAP

The core algorithm of LESS is built over the initial approach published in [1] and [2]. But, as clarified in the state-of-the art section, the LESS algorithm improves on quite a few aspects. For example: protecting the session key during the exchanges, enabling integrity during session establishment through AES-CCM rather than AES-CBC encryption and enabling separation in key used in reverse paths of communication. Table I lists the steps of LESS algorithm. Figure 2 illustrates the algorithm with a timing diagram and Table II explains the notations used.

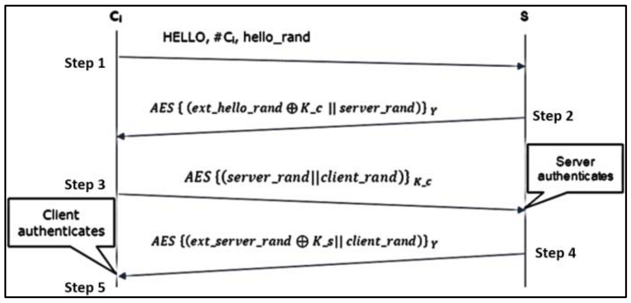
TABLE I. LESS: LIGHTWEIGHT ESTABLISHMENT OF SECURE SESSION: ALGORITHM STEPS

Step No.	Step Details	
	Action	***Description***
0	Pre-sharing secret (provisioning)	Secret Y= $\{0,1\}^{128}$ is shared between ith client Ci and server S offline at provisioning phase. The provisioning process is beyond the scope of the present paper.
1	Session initiation	Ci sends a "HELLO" containing #Ci and hello_rand to S at the time of session initiation. #Ci is the unique client ID and is pre-registered with the server. hello_rand = $\{0,1\}^{96}$is a unique random number.
2	Server challenge	S responds as: *AES {(ext_hello_rand⊕K_c || server_rand)}*$_Y$, where *K_c* = $\{0,1\}^{128}$,*server_rand*= $\{0,1\}^{96}$ and *ext_hello_rand = hello_rand || hello_rand[0:31]*. **K_c is the session key to be used for client-side encryption**. K_c is obfuscated with 'ext_hello_rand' as an extra protection since it is encrypted with Y which is common across sessions. 'server_rand' is the challenge to the client by the server.
3	Client response and challenge	Ci returns AES *{(server_rand || client_rand)}*$_{K\_c}$, where *client_rand* = $\{0,1\}^{96}$. 'client_rand' is the challenge by the client to the server.
4	Client authentication & server response	S verifies 'server_rand'from client with its own copy and thus authenticates the client and returns: *AES {(ext_server_rand⊕ K_s || client_rand)}*$_Y$ where, *ext_server_rand = server_rand || server_rand[0:31]*. *K_s* = $\{0,1\}^{128}$. **K_s is the session key to be used for server side encryption.**
5	Server authentication	Ci verifies the 'client_rand' from server with its own copy and verifies the server.

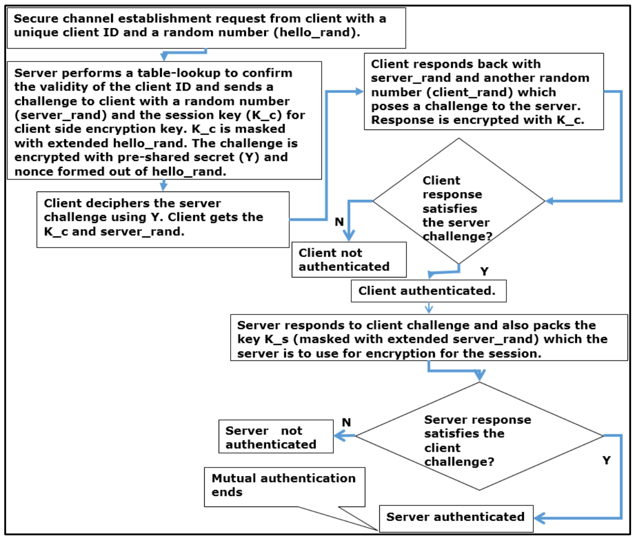
TABLE II. THE SYMBOLS AND NOTATIONS

Notation/ Symbol	Description
Y	Shared secret between the client and the server.
$\mathrm{AES}\{.\}_K$	AES-CCM-8 operation on plaintext using key K.
⊕	XOR operation.
||	Concatenation.
$\{0,1\}^n$	Binary bit-string of length 'n'.

Note: (1) The AES-128-CCM-8 encryption would need 12 bit nonce for each encryption and an additional data. (2) *'hello_rand', 'server_rand'* and *'client_rand'* serves as the required nonce values in steps 2, 3 and 4 respectively. (3) The 'additional data' is served by the header of each application layer message. *'hello_rand', 'server_rand', 'client_rand'* may be generated using the nonce generation process as described in [1]. (4) Generation of *K_c* and *K_s* is beyond the scope of the present paper.


Ci
S
HELLO, #Ci, hello_rand
Step 1
AES { (ext_hello_rand ⊕ K_c || server_rand)} Y
Step 2
AES {(server_rand||client_rand)} K_c
Server authenticates
Step 3
Client authenticates
AES {(ext_server_rand ⊕ K_s|| client_rand)} Y
Step 4
Step 5

(a)


Secure channel establishment request from client with a unique client ID and a random number (hello_rand).
Server performs a table-lookup to confirm the validity of the client ID and sends a challenge to client with a random number (server_rand) and the session key (K_c) for client side encryption key. K_c is masked with extended hello_rand. The challenge is encrypted with pre-shared secret (Y) and nonce formed out of hello_rand.
Client deciphers the server challenge using Y. Client gets the K_c and server_rand.
Client responds back with server_rand and another random number (client_rand) which poses a challenge to the server. Response is encrypted with K_c.
Client response satisfies the server challenge?
N
Client not authenticated
Y
Client authenticated.
Server responds to client challenge and also packs the key K_s (masked with extended server_rand) which the server is to use for encryption for the session.
Server response satisfies the client challenge?
N
Server not authenticated
Y
Mutual authentication ends
Server authenticated

(b)

Fig. 2. The timing diagram with mapping to steps in Table I (a) and the flow-chart (b) corresponding to the handshakes involved in LESS.

A. *Implementing LESS as CoAP Exchanges*

Implementation of the algorithm on CoAP follows the same payload embedded technique as in [1] and [2]. However, the individual message contents change in accordance with the modified algorithm described in Table I. Present method reuses the CoAP header options proposed in [1] [2]. The options are reproduced in Table III for ready reference. The handshakes required for implantation on CoAP is illustrated in Figure 3.

The maximum possible CoAP payload in the whole set of exchanges can be 36 bytes (exchanges in step II and IV of Table I) with the components given below:

(16 bytes of obfuscate session key + 12 bytes of the random challenge) + 8 bytes of additional data for CCM encryption required for integrity check = 36 bytes of application layer payload.

Hence, the CoAP message size during the session establishment process is far less than 64 bytes which is the minimum CoAP message block [11]. Thus the scheme guarantees no fragmentation of the application layer payload which leads to uncontrolled traffic in the network.

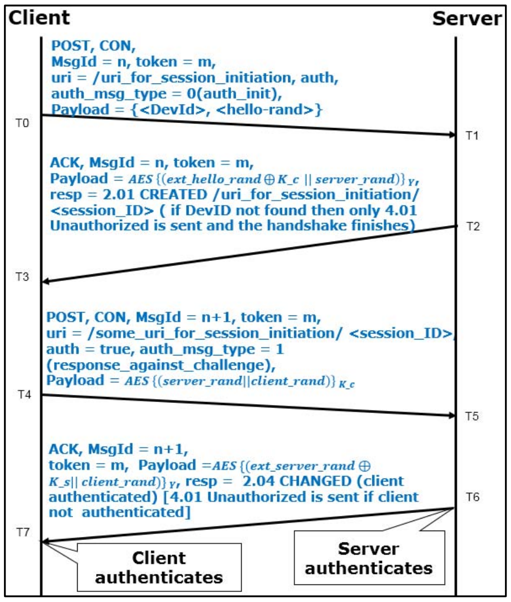

Client
Server
POST, CON,
MsgId = n, token = m,
uri = /uri_for_session_initiation, auth,
auth_msg_type = 0(auth_init),
Payload = {<DevId>, <hello-rand>}
T0
T1
ACK, MsgId = n, token = m,
Payload = $AES\{(ext\_hello\_rand \oplus K\_c \parallel server\_rand)\}_{Y}$,
resp = 2.01 CREATED /uri_for_session_initiation/ <session_ID> (if DevID not found then only 4.01 Unauthorized is sent and the handshake finishes)
T2
T3
POST, CON, MsgId = n+1, token = m,
uri = /some_uri_for_session_initiation/ <session_ID>,
auth = true, auth_msg_type = 1 (response_against_challenge),
Payload = $AES\{(server\_rand \| client\_rand)\}_{K\_c}$
T4
T5
ACK, MsgId = n+1,
token = m, Payload = $AES\{(ext\_server\_rand \oplus K\_s \| client\_rand)\}_{Y}$, resp = 2.04 CHANGED (client authenticated) [4.01 Unauthorized is sent if client not authenticated]
T6
T7
Client authenticates
Server authenticates

Fig. 3. Implementing LESS as CoAP handshakes using the confirmable (CON) message type.

TABLE III. COAP HEADER OPTIONS FOR LESS IMPLEMENTATION

Name	Format	Description
AUTH	empty	If the option is present it indicates that the present request relates to the authentication process.
AUTH_MESSAGE_TYPE	uint	0: auth_init (Session initiation request – indicates the "hello" message from client);
		1:Indicates a response from client against the server challenge

V. CHANNEL ENCRYPTION: MAPPING SESSION PARAMETERS AS DTLS SESSION

Once the session is established, the control switches from CoAP to DTLS in order to provide the channel security to the full application data comprising the application layer header and the information payload. In order to perform channel encryption DTLS-PSK needs the following session-parameter tuple [4] at both the client and the server ends:

{client write key, server write key, client initialization vector, server initialization vector}.

'Client write key' is 128 bit key for client side encryption and 'server write key' is the 128 bit key for server side encryption. Client and server initialization vectors (IV) are each of 4 bytes in length and are required as the IVs for AES encryption. The overall scheme is illustrated as a layered representation in Figure 4.

The client and server side encryption keys are derived during the session establishment process. The IVs are derived by XORing the 'client_rand' and 'server_rand' (exchanged during the session establishment process) followed by truncation. The IV generation process is illustrated in Figure 5.

Once the encryption parameters are computed the derived tuple is mapped to the DTLS session parameter structure for each record encryption as illustrated in Figure 6. It is to be emphasized that no modification in the existing structure is required. Mapping can be done seamlessly into the existing structure.

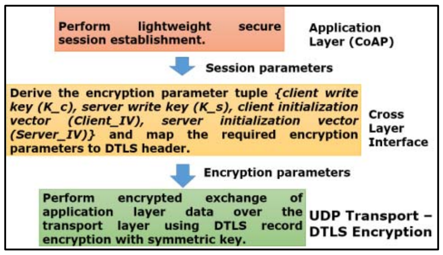

Perform lightweight secure session establishment.
Application Layer (CoAP)
Session parameters
Derive the encryption parameter tuple {client write key (K_c), server write key (K_s), client initialization vector (Client_IV), server initialization vector (Server_IV)} and map the required encryption parameters to DTLS header.
Cross Layer Interface
Encryption parameters
Perform encrypted exchange of application layer data over the transport layer using DTLS record encryption with symmetric key.
UDP Transport – DTLS Encryption

Fig. 4. A layered representation of the cross-layer implementation. The interface layer derives the one-time parameters required for DTLS record encryption for a given session.

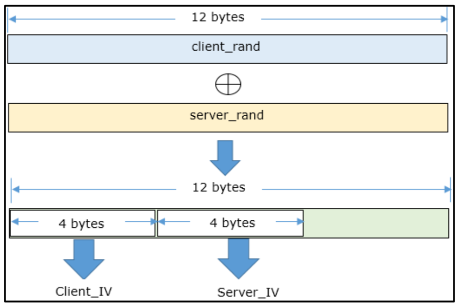


Fig. 5. The client and server IV generation process. Client_IV and Server_IV are required for both client side and server side encryption. This algorithm is identically executed at both client and server. Note that the required parameters to perform the calculation is available to both client and server after the session establishment process.

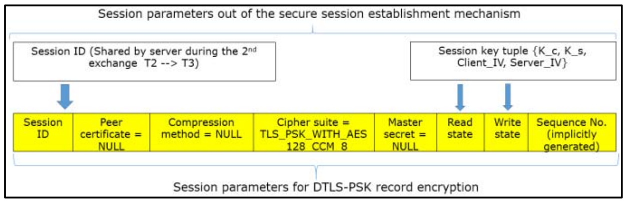

Session parameters out of the secure session establishment mechanism
Session ID (Shared by server during the 2nd exchange T2 --> T3)
Session key tuple {K_c, K_s, Client_IV, Server_IV}
Session ID
Peer certificate = NULL
Compression method = NULL
Cipher suite = TLS_PSK_WITH_AES 128 CCM 8
Master secret = NULL
Read state
Write state
Sequence No. (implicitly generated)
Session parameters for DTLS-PSK record encryption

Fig. 6. Fig. 6. Illustrating the mapping of the session parameters to the DTLS-PSK session parameter strycture. The fields are mapped as below:

Session ID => generated by the server and shared to the client through the response between T2 -> T3 of Figure 3;
Peer certificate & Compression method => NULL for DTLS-PSK [4];
Cipher suit => TLS_PSK_WITH_AES_CCM_8 as suggested by CoAP spec [3];
Master secret => The pre-shared secret maps here;
Read & Write state => Derived from the session key tuple from the interface;
Sequence No. => Generated implicitly by the DTLS transaction logic.

VI. SECURITY CONSIDERATIONS

A detail mathematical security analysis is performed in [2]. In this section we discuss resilience against some practical active and passive attacks. The definition of attacks considered are within the purview of [23].

A. Resilience to Passive Attack Due to Traffic Analysis

It is, in theory, possible to launch an offline passive attack by capturing the exchanged traffic over several sessions and predicting the encryption parameters. The attack can be mitigated by frequently changing the session parameters as the challenge response depends on several unique random numbers. However, in the proposed mechanism, the initial exchange (T2 -> T3 of Fig. 3) to establish the session parameters use the same pre-shared key across sessions. So, it can be argued that it is theoretically possible to launch a passive attack to get the session keys shared by the server. But, the server obfuscates the shared keys using the random number shared by the client along with the hello message. Thus, even if a theoretical attacker gets to predict the pre-shared secret, rightly guessing the each session key would need the attacker to try 2^{256}, or approximately 1.16×10^{77} combinations for each session. Accordingly, traffic-padding based traffic analysis and snooping attacks can be mitigated without incurring any additional bandwidth or latency cost.

B. Resilience to Denial of Service (DoS)

DoS has two aspects. Firstly, an attacker can consume resources on the server by transmitting a series of session initiation requests. Secondly, the attacker may use the server as an amplifier by issuing session initiation requests with forged source and causing message flooding.

Both the possibilities are countered through the challenge/ response mechanism. The client has to be able to properly decode the server challenge at the first place to carry on the next interactions. It is evident from the discussion in the above subsection that it is practically impossible for an attacker to get hold of the session keys shared by the server. Thus, any DoS attack would be ineffective just after the server challenge (step 2 in Table I).

C. Replay Protection

An attacker may simply replay the captured traffic to unnecessarily consume resources at the end-points. The proposed system circumvents this since it re-uses DTLS record encryption technique for securing the application data over the transport channel. DTLS inherently protects against replay attack through proper sequencing of the encrypted records in a session and a sliding receiving window [4].

VII. Results and Analysis

Experiments are performed to compare the performance of the LESS algorithm against an existing DTLS-PSK implementation. Californium [13] [14] is used as the off-the-shelf CoAP implementation with DTLS. The californium code base is modified to include the solution presented in this paper. Performance is measured in terms of average latency and bandwidth for the session-establishment process and also in terms of the rate of successful session establishment attempts under different packet error conditions. WANEM [15] is used as the Internet emulator. WANEM provides user handle to modify network parameters like bandwidth, packet loss ratio, etc. Wireshark [16] is used to analyze the resulting traffic. Figure 7 illustrates the experimental setup and Figure 8 shows the obtained results.

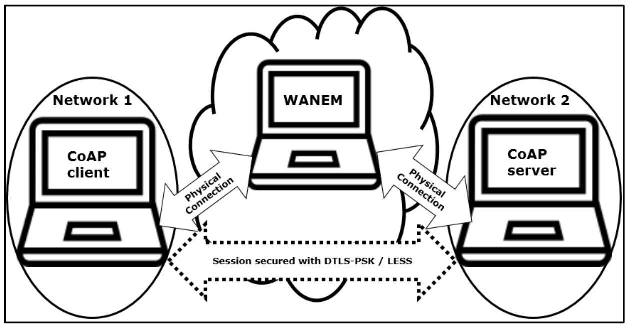
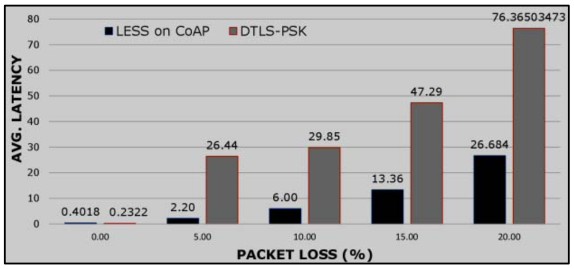
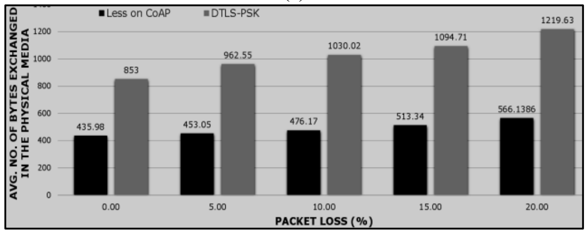


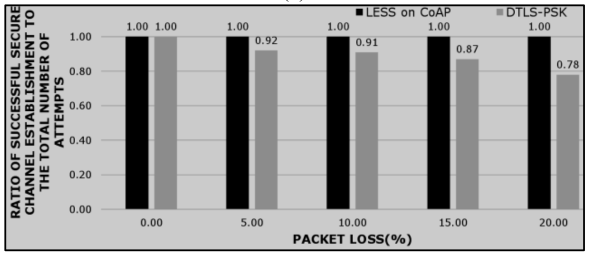
Fig. 7. The experimental setup for comparing the performance of original DTLS-PSK handshakes vs LESS. Two separate networks are formed. Each consist of one computation system. The system in the 1st network runs the client implementation and the system in the 2nd network runs the server implementation. The computers are physically connected over Ethernet to a 3rd system acting as a gateway. The gateway system runs WANEM and emulates the Internet between the two networks. The dotted line shows the logical secure session between client and the server across the networks. The session is established using both DTLS-PSK and LESS algorithm under different network losses configured using WANEM. The network bandwidth was configured as a nominal 9.6 kbps.



(a)



(b)



(c)

Fig. 8. The comparative results (avg. of 1000 iterations): (a) the average latency (in sec.) in a secure session establishment against different packet loss rates; (b) the average bandwidth consumed in terms of the bytes exchanged over the physical media for the secure session establishment process against different packet loss rates; (c) ratio of successful secure session establishment to the total number of session establishment attempts for different packet losses.

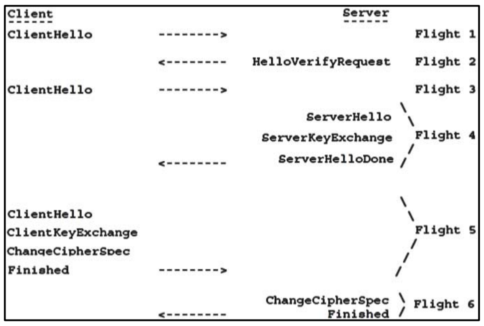


Fig. 9. Session-establishment in standard DTL-PSK [4][5].

A. Analysis of the Results

The results establish, LESS on CoAP shows consistent performance even in lossy conditions. LESS outperforms DTLS-PSK handshake in all aspects for obvious reasons. Firstly, LESS performs the full session establishment in just two request/response pairs against six 'flights' in DTLS (Figure 9). Again, each 'flight' may be composed of more than one messages and may lead to physically fragmented datagrams being transmitted. On the contrary LESS is implemented as small CoAP messages guaranteeing no fragmentation. LESS leverages CoAP's inherent reliability feature through CON mode which is very efficient and simple message acknowledgement feature incorporated in CoAP's messaging layer. This feature, coupled with un-fragmented low-sized messages over simple UDP, leads to the consistent performance of LESS (Figure 8(a) & (b)). Most importantly, LESS has 100% successful session establishment (Figure 8(c)) despite heavy loss. It is due to CoAP's reliability feature. But DTLS-PSK session establishment suffers with increasing packet loss. The captured traffic shows that DTLS-PSK fails message integrity check during the 5th and 6th flights more often with increasing packet loss. However, latency of LESS under 0% packet loss is little higher than DTLS-PSK (Figure 8(a)) as LESS algorithm involves cryptographic operations. The total latency displayed incorporates this computational latency. DTLS does not involve any such computation during session establishment. But, this difference loses significance for lossy conditions.

VIII. Conclusion and Future direction

A novel lightweight cross layer approach for secure session establishment and exchange of application layer message through a secure channel is discussed. DTLS is an heir to TLS and is not originally designed for constrained environments. Efforts are on to fit DTLS into such applications. But an application-layer controlled scheme like the present one may be a sane alternative. The proposed system is implemented such a way that it reuses most of the routines available already in CoAP and DTLS-PSK and adds minimal additional code. The system needs to maintain simple state machine (not discussed) to handle both CoAP over UDP and DTLS traffic. If only 'object security' suffices, the cross-layer interface may be removed. The keys for the opposite paths derived during secure session establishment may be used to encrypt and exchange encrypted CoAP payloads over simple UDP transport resulting into a very lightweight "object security" solution. Also, enabling CoAP with inherent security may be a very futuristic approach as CoAP is poised to be used on different alternative transport which may not be DTLS compatible. Lightweight multicast security is another need for IoT/M2M systems. But, a proper solution is still awaited [17]. Present work is to be extended for multicast security solution. The present work will be augmented with the work on lightweight solution for intelligent transportation [24][25] to create a complete lightweight secure solution.

References

[1] A. Ukil, S. Bandyopadhyay, A. Bhattacharyya, A. Pal, "Auth-Lite: Lightweight M2MAuthentication reinforcing DTLS for CoAP," in PERCOM Workshops, Budapest, 2014, pp. 215-219.

[2] A. Ukil, S. Bandyopadhyay, A. Bhattacharyya, A. Pal, "Lightweight security scheme for vehicle tracking system using CoAP," in Proc. ASPI'13, Zurich, 2013.

[3] Constrained Application Protocol (CoAP), RFC 7252, June 2014.

[4] Datagram Transport Layer Security Version 1.2, RFC 6347, January 2012.

[5] Pre-Shared Key Ciphersuites for Transport layer Security (TLS), RFC 4279, December 2005.

[6] S. Bandyopadhyay, and A. Bhattacharyya, "Lightweight Internet protocols for web enablement of sensors using constrained gateway devices," in Proc. ICNC, pp. 334 – 340, January 2013.

[7] HTTP over TLS, RFC 2818, May 2000.

[8] The Transport Layer Security (TLS) Version 1.2, RFC 5246, August 2008.

[9] https://datatracker.ietf.org/wg/dice/charter/.

[10] AES-CCM Cipher Suites for Transport Layer Security (TLS), RFC 6655, July 2012.

[11] C.Bormann and Z. Shelby. Blockwise transfers in CoAP (draft-ietf-core-block-14) (http://tools.ietf.org/html/draft-ietf-core-block-14) .

[12] Hypertext Transfer Protocol – HTTP/1.1, RFC 2616, June 1999

[13] Californium (Cf) (http://people.inf.ethz.ch/mkovatsc/californium.php).

[14] https://github.com/mkovatsc/Californium.

[15] http://wanem.sourceforge.net/.

[16] https://www.wireshark.org/.

[17] A. Rahman and E. Dijk. Group Communication for CoAP (draft-ietf-core-groupcomm-25).

[18] T. Kothmayr, et al. , "A DTLS based end-to-end security architecture for the Internet of Things with two-way authentication", in Proc. LCN Workshops, Florida, 2012

[19] The EAP-PSK Protocol: A Pre-Shared Key Extensible Authentication Protocol (EAP) Method, RFC 4764, January 2007.

[20] Z. Bajic, R. Nagarajan, R. Vijayakumar, "Methods and apparatus for layer 2 and layer 3 security between wireless termination points", U.S. Patent 8281134, Oct. 2012.

[21] O. Delgado-Mohatar, et al.,A light-weight authentication scheme for wireless sensor networks. Ad Hoc Networks., Vol. 9, Issue. 5, pp. 727-735

[22] H. Liao, et al, "Method and system for secure lightweight transactions in wireless data networks", U.S. Patent 6148405 A, Nov. 2000.

[23] Internet Security Glossary, Version 2, RFC 4949, August 2007.

[24] A. Bhattacharyya, S. Bandyopadhyay, A. Pal, "ITS-light: Adaptive lightweight scheme to resource optimize intelligent transportation tracking system (ITS) Customizing CoAP for opportunistic optimization", 10th International Conference on Mobile and Ubiquitous Systems (Mobiquitous 2013), January 2013.

[25] A. Bhattacharyya, S. Bandyopadhyay, A. Pal, "Adapting protocol characteristics of CoAP using sensed indication for vehicular analytics", 11th ACM Conference on Embedded Networked Sensor Systems, SenSys, 2013.

[26] A. Ukil, S. Bandyopadhyay, A. Bhattacharyya, A. Pal, T. Bose, "Lightweight security scheme for IoT applications using CoAP," Internation Journal of Pervasive Computing and Communications, vol. 10, pp. 372-392, 2014.