Middle-East Journal of Scientific Research 23 (9): 2051-2059, 2015
ISSN 1990-9233
© IDOSI Publications, 2015
DOI: 10.5829/idosi.mejsr.2015.23.09.22400

Incorporation of Security Features in Agonistic Application Protocol in the Web Search

[1]Dr. A. Rengarajan, [2]S. Rajasekaran and [3]P. Kumaran

[1]Professor, CSE, Veltech Multitech Engineering College, Chennai, Tamilnadu, India
[2]Junior Research Fellow, Veltech Multitech Engineering College, Chennai, Tamilnadu, India
[3]Ph.D. Scholar, CSE, Anna University, Chennai, Tamilnadu, India

Abstract: The Internet of Things (IoT) is an outline in which objects, animals and people are handover with exclusive identifiers and the capability to transfer data over a grid without demanding human-to-human and human-to-computer interaction. Based on the performance scenario, such communication may take consign over a public network such as the Internet, which is based on the TCP/IP stack. However, dissimilar research functioning group's dispute that a few of these stack protocols such as the Hyper Text Transfer Protocol (HTTP) might not be fitting for controlled devices. Therefore, the IETF Constrained RESTful Environments (CoRE) WG has proposed the Constrained Application Protocol (CoAP); an application layer protocol for constrained devices in the Internet of Thing. The CoRE WG examined with IPSec or DTLS is to maintain the CoAP communication at different levels of the protocol stack. On the other hand it is used to examine the opportunity of such a proposal. For this thing are apply the X.805 security standard to judge tentatively the security aspects of such performance. The investigation things to see the major security drawbacks and to argue the need of a new incorporated security solution.

Key words: CoAP (Constrained Application Protocol) · IPSec (Internet Protocol Security) · DTLS (Datagram Transport layer security) · S-CoAP (Secure Constrained Application Protocol) · X.805 · 6LBR

INTRODUCTION

A article, in the Internet of Things, can be a human being with a heart monitor embed, a farm living thing with a biochip transponder, an automobile that has built-in sensors to attentive the driver when tire force is low or any other usual or man-made thing that can be assign an IP address and provide with the aptitude to shift data over a network. There is an continuing tendency in integrate machine-to-machine (M2M) and wireless sensor network (WSN) solution with further established Internet services by existing Internet protocols (IPs). This tendency has brought the standard IP-centric protocols keen on the kingdom of smart devices and smart objects. Solutions such as the iDigi Connect effort to bring standard connectivity to entrenched devices through TCP/IP stacks and Web services during which data acquired can be aggregated over the Internet for visualization and analysis [1]. The IoTs adaptation has turn into more obvious, realistic and significantly facilitate through the deployments of IPv6, as the large address space provided in IPv6 enables more machines to be available over the Internet and thus, converse with each other. According to World Wide Web Consortium (W3C) [2], there are two approaches to understand and control web services, the Representation State Transfer (REST) and the arbitrary Web Services. REST (Representational State Transfer) is an architectural approach, an approach to communications that is frequently used in the expansion of Web services. The use of REST is often favoured over the further hardwearing SOAP (Simple Object Access Protocol) method because REST does not influence as much bandwidth, which makes it an enhanced fit for utilize over the Internet. The SOAP approach requires inscription or a provided server program (to serve data) and a client program (to request data). Clients communicate through the server synchronously in a request /response technique using methods of a transfer protocol such as the Hypertext Transfer Protocol (HTTP) [RFC 2616]. HTTP is the essential protocol used by the World Wide Web. HTTP define how communication are formatted and transmitted, with what actions Web servers and browsers should obtain in response to various commands. For example, when you come in a URL in your

Corresponding Author: A. Rengarajan, Veltech Multitech Engg College, Chennai, Tamilnadu, India.

browser, this actually sends an HTTP command toward the Web server directing it to bring and transmit the requested Web page. The IETF Constrained RESTful Environments (CoRE) workgroup is working on the application-layer protocol for introducing the Web-services pattern in the IoT [3]. The CoRE group has defined a REST-based Web transfer protocol called the Constrained Application Protocol (CoAP). This protocol includes more than a few HTTP functionalities but has been redesigned to report for the low dispensation power and energy utilization constraints of IoT devices. CoAP, like HTTP, identifies resources with a universal resource identifier (URI) and allows the resource to be precious using similar methods such as GET, PUT, POST and DELETE. But CoAP is not just a blind density of HTTP. The further main standard is that it controls the World Wide Web works in HTML, which covers how Web pages can be formatted and displayed. However, incorporate the IoTs into traditional Internet and it is not an uncomplicated operation; this is due to the differences between the execution model of the IoTs-based applications and the present applications in the Internet. For illustration, most Internet applications that utilize HTTP protocol depend on the Pull Model [4]. For data exchange, while inactive nodes in the IoTs will be placed to sleep mode to keep the battery life and expand the life cycle of the node. These nodes just wake up to perform specific responsibilities when required. Simple Object Access Protocol (SOAP) [5] and Extensible Markup Language (XML) [6] are not suitable for such facilitation. SOAP (Simple Object Access Protocol) is a messaging protocol that allows programs to run on dissimilar operating systems (such as Windows and Linux) to communicate with Hypertext Transfer Protocol (HTTP) and its Extensible Markup Language (XML). This has been realized by dissimilar research groups, which contain proposed mechanisms to assist the communications between constrained devices within the IoTs model. Examples of these novel technologies (enablers) are the IPv6 over Low Power Wireless Personal Area Networks (6LowPAN) [7] because a network level enabler, the Routing over Low-power and Lossy networks (ROLL) [8] to give a routing mechanism optimized for constrained networks and the Constrained Application Protocol (CoAP) as an application layer enabler [3], [9].

Research Motivation: The IETF Constrained RESTful Environments (CoRE) performance group [10] has projected the CoAP as a new application-level protocol used for constrained devices. In order to build the protocol appropriate to IoT and M2M applications, various innovative functionalities have been added. The core of the protocol is individual in RFC 7252; significant extensions are in various stages of the homogeny process. To begin with, the CoAP has no security features; recent research mechanism has proposed deploy the DTLS or IPSec protocols to provide a secure CoAP implantation. We investigate the efficiency of this proposal and use the X.805 security standard toward analyzes the security of the resulting scheme of implementing both of the DTLS and IPSec and not as an independent manuscript.

Constrained Application Protocol (Coap: an Overview): As HTTP, COAP is also a network-oriented protocol, except low transparency, multicast, etc. We all know HTTP protocol has long-term victory; it may use small characters to integrate different possessions and services. HTTP engaged in Application level and provides Interoperation which is the key point of IoT. However, HTTP is point to point (p2p) communication model which is based on TCP protocol which is not suitable for notification drive services. Also, HTTP is much complex for mortified devices [11]. The COAP is an application layer protocol that has been advanced with restful-integrated interface as well as utilized over constrained networks. For 6LowPAN networks it has been generally targeted as constrain network. To serve from existing web-based technologies, the COAP has been studied to be HTTP-suited and uses related methods as HTTP does; i.e. GET, POST, PUT. However, as a transport layer COAP is deployed of User Datagram Protocol [RFC 768] (UDP) that is one of the key features rather than TCP. Because of the connectionless quality of the UDP, the COAP be capable to make available a lightweight reliability mechanism by splitting COAP protocol keen on 2-layers as shown in the subsequent sketch:

The Request/Response layer is accountable for operating the possessions by determining methods (i.e. GET, POST, DELETE and PUT) When providing message replication detection and processing messages, the transaction layer realizes the consistency mechanism. In the Transaction layer, a message might be any one of the four types:

- Confirmable (Requires Acknowledgment).
- Non-Confirmable (Requires no ACK).
- Acknowledgment (to ACK. CON. Messages).
- Reset (message is received but could not be processed).

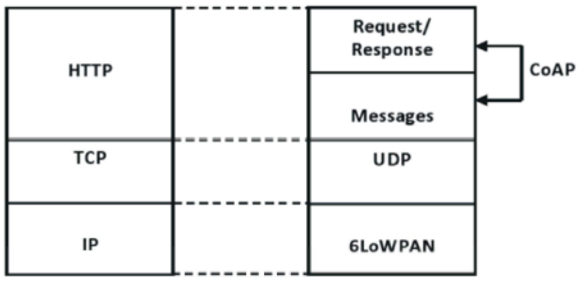

Fig. 1: HTTP and CoAP protocol stacks

In contrast with above mentioned capabilities, COAP has the subsequent important features that are prime necessities in IoTs environments, for example: Low header overhead, multicast support, URI, simple parsing process, asynchronous message exchanges and content-type support.

COAP Communication Model: A token is used to equals each response to its similar request. Messages will be swapped in an asynchronous form and will carry the requests, responses and the semantics. Since COAP is enclosed with UDP, density has been integrated into COAP and conveniently used. And for Reliability, the mechanism is light-weight and has the subsequent features:

- Simple Stop-and-Wait Re-transmission with exponential back off, for CON messages.
- Duplicate detection for CON and Non-CON messages.

Message Layer Model: It supports 4 types message: CON (confirmable), NON (non-confirmable), RST (Reset), ACK (Acknowledgement) [3], [9], [12].

Reliable Message Transport: It preserves retransmission until get ACK with the same message ID (like 0x8c56 in Fig. 2). When transmitting CON employing default time out and decreasing counting time exponentially. It responses by substituting ACK with RST, if recipient give out to process message. Fig. 2 depicts a reliable message transport.

Unreliable Message Transport: In case of retransmission, when transporting with NON type message, it doesn't need to be ACK, but has to contain message ID for supervising. Server replies RST, if recipient give out to process message. Fig. 3 shows unreliable message transport.

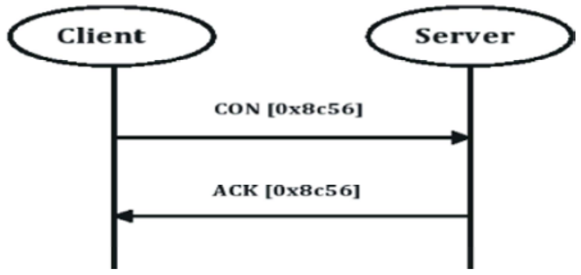

Fig. 2: Reliable Message Transport

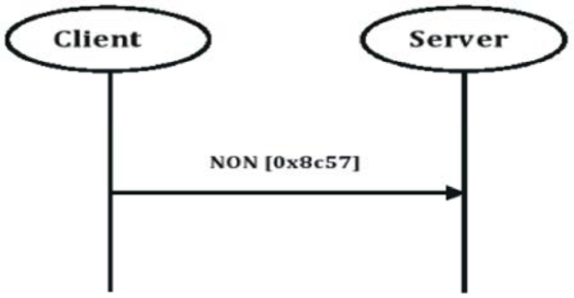

Fig. 3: Unreliable Message Transport

Request/Response Layer Model

Piggy-Backed: Client sends request applying CON type or NON type message and receives response ACK with confirmable message at once. In Fig. 4, ACK contain response message (identify by using token) for successful response, ACK contain failure response code for failure response. Consequently the messages will be swapped either reliably or non-reliably, turning on option defined in the GET request header. If the resource is accessible, then the server will send the response in a piggybacked fashion with the ACK message, in the time of processing the CON-Request, showed in the subsequent diagram:

Separate Response: If server receive a CON type message but could not able to response this request at once, it will send a blank ACK in case of client resend this message. When server prepares to response this request, it will send a new CON to client and a reply of confirmable message will be send by a client with acknowledgment. No matter CON message carry request or response, ACK is due to confirm CON message (Fig. 5). Because of the lack of appropriate respond, the server cannot respond once to CON request message, it simply acknowledges the request with a blank ACK message. The server will send the respond in a new CON message once the resource becomes available, which in turn will be acknowledged by the client. The following diagram represents this:

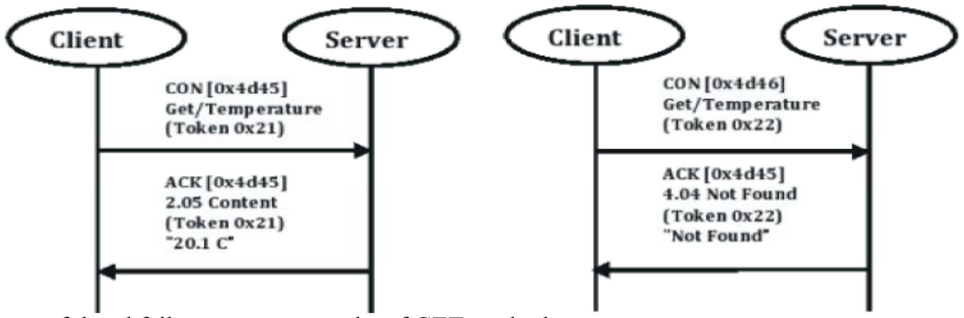

Fig. 4: The successful and failure response results of GET method

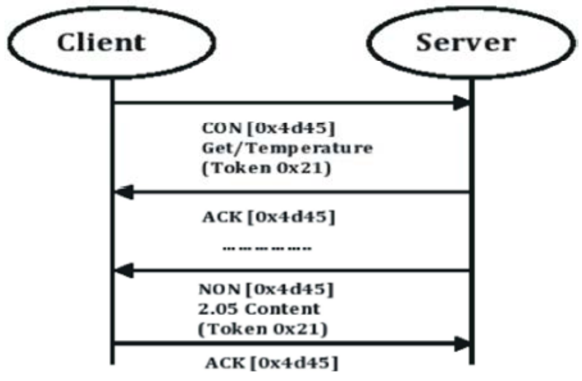

Fig. 5: A Get request with a separate response

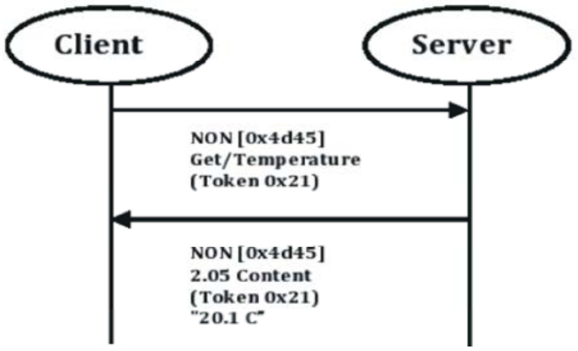

Fig. 6: Non confirmable request and response

Non Confirmable Request and Response: Disparate Piggy-backed response carry confirmable message, Non confirmable request client send NON type message denote that Server do not need to get confirm. NON type message with response will be resend by the server (Fig. 6).

The earliest design of the COAP has no security features [3], [9], only lately, researchers have aspect to inquired the security of COAP implementations. The COAP Internet draft stated two security protocols that can be used to secure COAP network and its traffic, namely, DTLS and IPSec. Later, Internet draft has been formed by extracting the IPSec protocol from COAP draft [13]. Additionally, few proposals have been issued with regards to COAP security and they are either IPSec-based or DTLS-based [14], [15], [16].

The X.805 Overview: As portrayed in [17], the X.805 standard outlines three security layers (services, application and infrastructure), three security planes (control, end user and management) which are distinguished based on the activities accomplished over the network and also 8 security dimensions to address general system vulnerabilities (Access Control, Data Integrity, Authentication, Security, Availability, Non-Reputation, Data Confidentiality, Communication and Privacy). A complete set of end-to-end view of network security and top-down systematic approach provided by security architecture that can be applied to network elements, application and services in order to predict, detect and correct security vulnerabilities. Fig. 7 depicts the complete architecture of the X.805 standard including Security Layers, Planes and Dimensions:

The X.805 standard could refer to various technologies such as wire-lined, wireless and optical networks. It could also be used over different types of networks such as service provider networks, governmental networks, enterprise networks and data centre networks [18]. The major network threats could be classified as Removal, Destruction, Disclosure, Corruption and Interruption of data or network's resources. To validate security mechanisms we have used X.805 standard in this paper; for an IToS constrained network especially DTLS and IPSec protocols that's running COAP. The following sketch shows one of the practicable scenarios for COAP networks:

It is apparent from the above diagram that COAP interactions could be machine-to-machine or in any other cases would be a client/server approach. COAP server will act as a gateway proxy in some implementations to relay HTTP's client request to penetrate a resource on COAP server. Here COAP proxy will map or translate between

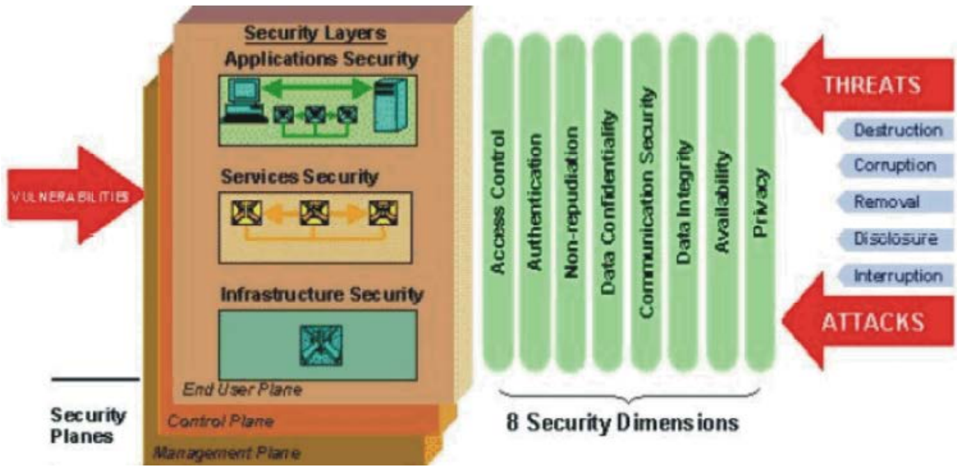

Fig. 7: X.805 – Security Standard Architecture

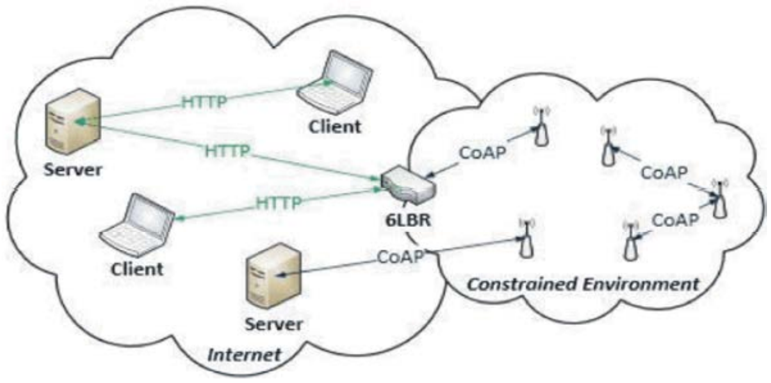

Fig. 8: CoAP network extended to Internet [15]

COAP and HTTP and converse. We would conclude that COAP network can be a closed network that does not extend to the Internet, thus COAP traffic will be internally swapped between COAP clients and COAP server(s). On the other hand, with the aid of the 6LowPAN and HTTP/COAP mapping process, COAP network can be stretched and integrated into custom Internet. In this case, traversing traffic would be direct since COAP defines a subset of HTTP protocol in its system. Though, security still a problem and must be addressed. As described earlier, COAP authors stated either DTLS or IPSec to defend COAP interactions. A security analysis for DTLS and IPSec will be carried out in the coming section by applying X.805 threat model.

COAP-DTLS Security: An enhanced version of the widely used Transport Layer Security (TLS) protocol is the DTLS protocol [RFC 5246]. The major difference is instead of TCP, DTLS runs on top of UDP to secure major UDP renowned applications such as Voice over IP /Session Initiation Protocol (VoIP/SIP). DTLS delivers authentication, confidentiality, data integrity and automatic key management. To make it a potential security protocol candidate DTLS reinforces wide range of dissimilar cryptographic algorithms. In order to achieve security services required, COAP defines four security modes in conformity with COAP's draft. These modes are: No Sec, Certificate, Raw Public key and Pre Shared Key. Over Ip, packets are sent normally as UDP datagram's in No Sec mode and denoted by COAP scheme as coap://. In the other 3 security modes, security is accomplished by DTLS and the scheme gets coaps: //. The following two designs simply depict message exchange for COAP with and without DTLS; Fig. 9 depicts the CoAP Request/Response, 1 round trip without DTLS.

- First, DTLS protocol does not support multicast communications, which is an important part of COAP protocol and prime criterion in IoTs.

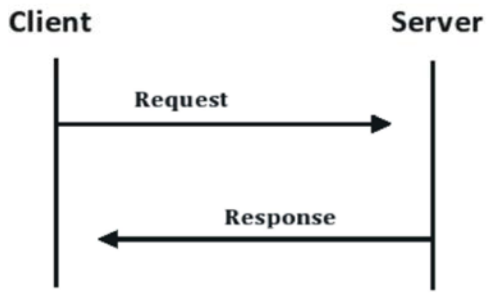

Fig. 9: CoAP Request/Response, 1 round trip without DTLS

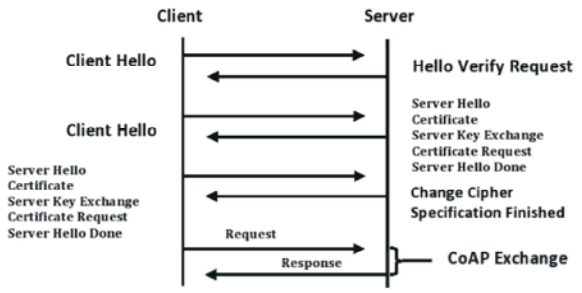

Fig. 10: CoAP Request/Response with DTLS, 4 round trips

- Second, with the stateless cookies, the DTLS handshake protocol could induce (prone to) exhaustion attack of the resources of battery-powered device. As a result of this, the nodes could lose their roles in the network and cause disruption to the entire communication.
- Third, however DTLS could secure against replay attack by using bitmap window, nodes have to process the received first and even forward them sometimes. Without filtering proxy namely 6LoWPAN Border Router (6LBR), the capability of this attack could lay down the network flooded. On all scenarios managing such filtering on a 6LBR cannot by guaranteed. Besides, the processing of replied packets is energy consuming.
- Fourth, since no end host has been authenticated to the other end-host, handshake phase is securely liable. And its messages fragmentation is still a problem while a friendly solution was suggested without a validation [19]. Furthermore, to validate the handshake messages, the hash function is claimed to be executed on all messages which denotes some nodes needs large buffer and this is not useful in each case.
- Fifth, DTLS security features do not suited well for COAP. For example, the lost in-flight messages need the retransmission of all messages. Similarly, large buffer is needed if the resource is more when all messages in-flight are transmitted together in a single UDP packet. Moreover, COAP/HTTP mapping process is required when a client needs to access a internet and then DTLS handshake process abides a challenge. Most importantly it is still unsure whether we can perform a partial mapping between TLS and DTLS or not. This issue is more complicated only because a COAP client would not be able to understand which device has constituted the request.
- Finally, COAP messages cost the network only in 2 transactions (1 round-trip); one message from the client (request) and the other from server (Response). 4 round trips are required if DTLS is used; 3 round trips for DTLS (~ 40-50 Bytes) plus 1 round trip for COAP just before COAP's actual contents are swapped.

COAP-IPSEC Security: As stated in [13], in addition to DTLS security protocol as a security means for COAP, different implementations and applications could use IPSec. IPSec is a layer 3 protocol resolved to be used with IPv6 but has been changed to be IPv4 compatible. It is an application independent protocol that can defend application and applications that resides on transport layers. IPSec is integrated into the kernel; so it is transparent to applications. Because of its transparency, IPSec can also use TLS and Secure/Multipurpose Internet Mail Extensions security protocols [RFC 3851] (S/MIME). IPSec provides the following security services: Confidentiality, Connectionless Integrity, Data origin Authentication, Anti-Replay mechanism, Access control and Limited Traffic Flow Confidentiality. To safeguard COAP transactions with IPSec, a method is to use Encapsulating Security Payload Protocol [RFC 2406] (IPSec-ESP), particularly if the hardware supports encryption at layer 2, as it is the case with some IEEE 802.15.4 radio chips. To make use of IPSec with AH [RFC 2402] or ESP authors in [16] proposed a 6LowPAN.

- First, Originally, IPSec & DTLS were not arranged to deal with constrained environment – constraints were not taken into IPSec/DTLS designs' objectives.
- Second, IPSec have Known problems if it uses Network Address Translation (NAT) and/or Port Address Translation (PAT).
- Third, the encryption process of IPSec generates a large overhead, when transmitting small packets, thus depreciating the performance of the network.

Table 1: Security Services Provided by Ipsec and Dtls Using X.805 Security Standard Architecture

Security Dimension	IPSEC	DTLS
Access Control	No	No
Authentication	Yes	Partially-Server Only
Non-Repudiation	Yes/No; Depends on Authentication method. PKI not supported by constrained devices	Yes/No; Depends on Authentication method. PKI not supported by constrained devices
Confidentiality	Yes	Yes
Communication Security	Yes	Yes
Integrity	Yes	Yes
Availability	Mitigation- No full protection	Yes- Stateless cookie
Privacy	No	No

- Fourth, the mobility is a problem in IoTs when it comes to Security Associations (SA). SA is individually identified by 3 parameters: Security Parameter Index (SPI), Security Protocol Identifier and Destination IP Address. After the creation of the SA, a node changes its IP address, then another SA needs to be created then the newly created node will contribute to performance degradation.
- Fifth, IPSec require kernel level for any changes or modification needs to be performed since it is embedded in the IP stack.
- Sixth, Configuring/Managing/Troubleshooting IPSec and Internet Key Exchange (IKE) are intricate tasks giving the enormous number of constrained devices participating in the network. Due to this mis configuration security parameters of IPSec could lead to security holes or performance problems.
- Seventh, IPSec will not support all scenarios/nodes. For example, there is no security service provided by IPSec. (i.e.) when operating on two different environments and administrative policies, one side does not support IPSec.
- Finally, IPSec support for Multicast communication is difficult.

Finally, COAP's draft stated that [9], in layer-2 encryption hardware it is viable to use IPSec (ESP) that supports the use of AES-CBC (128-bit keys). Though this approach is applicable to some devices, IOTs, classes, specifically class 2 devices (RAM size = ~50 KB, Flash size =~250 KB) and the devices/things' capabilities will vary in some scenarios and consequently not all devices can process IPSec which leads to operations complexities. A concise of the analysis is depicted in the Table 1:

More over the earlier mentioned concerns, it is accepted that IPSec and DTLS are not the most optimized solutions to secure COAP for the subsequent reasons:

- First, extra messages are required to negotiate the security parameters and set up the security associations (SAs) in IPSec and DTLS. Due to this, there will be an increase in overhead and the resources of the constrained devices are drained out. When seeing the mobile nature of devices in ITOS, this problem is more difficult as it needs to establish every time the device moves.
- Second, if we consider the outline of the communication between two different networks, the proposed security solution is based on either IPSec or DTLS, which states the presence and support of these protocols in both the source and destination networks. Since IPSec protocol has a compatibility problem with firewalls across networks, this assumption will not work well in all situations.
- Third, for setting up the secure association, IPSec and DTLS rely on protocols like the Internet Key Exchange (IKE) and the Extensible Authentication Protocol (EAP). This also implies the protocols like IKE and EAP need to be supported by all constrained devices vendors.
- Fourth, IPSec and DTLS have initially been implemented to secure connections between two static and remote devices. Also it attempt to attain the most possible secure connection between the two ends, not considering the QOS, the network dependable or any other drawbacks on the end devices. Though, dynamic and sensible measures are needed when we consider the security in constrained environment, which when negotiating the security parameters consider the constrained nature of the end devices.
- Fifth, IEEE 802.15.4 specification states that the payload to be 127 bytes as whole. To protect COAP exchanges, we can utilize DTLS as security protocols, 13 bytes (out of the 127 bytes of IEEE 802.15.4 frame) will be allocated for DTLS record. For link layer addressing information 25 bytes is used, 10 bytes for 6LowPAN addressing, in addition to the 4 bytes of COAP header. As a result 75 bytes left for application layer payload, which is not much space for transferring actual data. Hence, more resource will be used from the nodes for one big chunk of data

(bigger than 75 bytes) and the network itself will be chunked and sent twice. So, whenever it is possible some header compression mechanisms have been proposed. Because of compressing and decompressing the requirements, the compression mechanism adds more constraints to the nodes and network resources.

- Sixth, some DTLS applications might require security services to flexibly customize according to the application or scenarios needs. For example, some applications want to secure the message depending on their messages types. This process is not possible with DTLS protocol because after completing the DTLS handshake protocol, the nodes would have already agreed on security policies/cipher posse to protect all logical messages, which will be done clearly. DTLS protocols would contribute to reduce the usage of resources available if we apply security according to the needs of the application or scenario and its highly likely would increase the network performance.
- Finally, the author in internet draft [19] mentioned that if we utilized constrained environment there will be 7 potential problems related to DTLS protocol. Therefore to tackle this problem the authors have proposed some work. Though much work is needed to adjust DTLS to be mainly potential security solution for IOTs and it is known that there is no studies exist to depict whether or not these work would effectively be appropriate.

The authors propose the Secure COAP (S-COAP), a secure version of COAP, in order to address the highlighted issues. This proposal is same as some well-known earlier works such as Transport Layer Security over Stream Control Transmission Protocol (SCTP-S) [RFC 3436] or Domain Name System Security (DNSSEC) [RFC4033]. In all these works security mechanism has been an integrated part of the protocol itself. Since security measures of S-COAP have been built-into the plain COAP transactions, it will have its own negotiation stage that includes the limitations of the constrained device. In case of normal connection setup and mobility, S-COAP needs to provide security for it. Since security will be part of the COAP protocol, which is not provided by any other standards, conveys that the S-COAP should be able to operate across multiple sites and networks.

CONCLUSION

In this paper, we have proposed IPSec and DTLS protocols to secure the COAP in IoTs. To do so we have explored these proposed protocols and its implementation. But if we analysis the fact these protocols failed to satisfy some security requirements. Additionally, if we try to deploy DTLS and IPSec with constrained devices in IOTs, it will leads to the issue of usability. Therefore, for a secured version of COAP, this paper argues for the need of new lightweight, integrated security mechanism. Work is already in progress within our group to implement the new security protocol which will be verified using formal methods approach.

ACKNOWLEDGEMENT

Authors deliver their graduate to SERB (Young Scientist Scheme, No. SP/FTP/ETA-51/2013) Govt. of India for Financial Assistance and FIST F.NO:SR/FST/College-189/2013.

REFERENCES

1. (Digi: http://www.digi.com/products/wireless-wired-embedded-solutions/solutions-on-module/digiconnect/
2. Booth, D., H. Haas, F. McCabe, E. Newcomer, M. Champion and C. Ferris, 2004. “Web Services Architecture”. W3C Working Group Note 11 February 2004. Available: http://www.w3.org/TR/ws-arch/
3. Shelby, Z., K. Hartke, C. Bormann and B. Frank, 2011. “Constrained application protocol (coap),” draft-ietf-corecoap-07, 2011.
4. Shelby, Z., 2010. “Embedded web services,” IEEE Wireless Communications, 17(6): 52-57.
5. Gudgin, M., M. Hadley, N. Mendelsohn, Jean-Jacques Moreau, H. Nielsen, A. Karmarkar and Y. Lafon, “Simple Object Access Protocol (SOAP)”. Version 1.2 Part 1: Messaging Framework (2nd Ed.). Available: http://www.w3.org/TR/soap12-part1/
6. (Extensible Markup Language (XML) Available: http://www.w3.org/XML/
7. Kushalnagar, N., G. Montenegro, C. Schumacher, 2007. “IPv6 over Low Power Wireless Personal Area Networks” [R FC 4944]. Aug 2007. Available: http://tools.ietf.org/html/rfc4944.

8. "Routing Over Low power and Lossy networks" ROLL Working Group. Available at: http://datatracker.ietf.org/wg/roll/.
9. Shelby, Z., B. Frank and D. Sturek, 2013. "Constrained Application Protocol (CoAP)". Internet-Draft (work in progress draft-ietf-core-coap-018), Sensinode, SkyFoundry, Pacific Gas and Electric, June 2013.
10. Constrained RESTful Environments (CoRE) Working Group. Available at: https://datatracker.ietf.org/wg/core/charter/
11. Xi Chen, chen857 at wustl.edu (A paper written under the guidance of Prof.Raj jain.
12. Shelby, Z., Sensinode and K. Hartke, "Constrained Application Protocol (CoAP)," draft-ietf-core-coap-18. [2013-06--28] http://tools.ietf.org/html/draft-ietf-core-coap-18.
13. Bormann, C., 2012. "Using CoAP with IPSec". Internet-Draft (work in progress draft-bormann-core-ipsec-for-coap-00). Universitaet Bremen TZI, December 06, 2012.
14. Kothmayr, T., C. Schmitt, W. Hu, M. Bruenig and G. Carle, 2012. "A DTLS Based End-To-End Security Architecture for the Internet of Things with Two-Way Authentication". In Proc. of IEEE SenseApp, 2012.
15. Raza, S., D. Trabalza and T. Voigt, 2012. "6LoWPAN Compressed DTLS for CoAP". DCOSS, pp: 287-289. IEEE, (2012).
16. Raza, S., D. Trabalza and T. Voigt, 2011. "6LoWPAN Extension for IPSec". Interconnecting Smart Objects with the Internet Workshop, 25 March 2011, Prague, Czech Republic.
17. Zeltsan, Z., 2013. "ITU-T Recommendation X.805 and its application to NGN". http://www.itu.int/ITUT/worksem/ngn/200505/presentations/s5-zelstan.pdf (last accessed 14/10/2013.
18. Bell Labs, 2013. "The Bell Labs" Security Framework: Making the Case for End-to-End Wi-Fi Security. http://www.webtorials.com/main/resource/papers/lucent/paper90/wireles s3.pdf (last accessed 14/10/2013).
19. Hartke, K. and O. Bergmann, 2012. "Datagram Transport Layer Security in Constrained Environments" Internet-Draft (work in progress draft-hartke- core-codtls-02). Universitaet Bremen TZI, July 2012.