Internet Engineering Task Force
Internet-Draft
Intended status: Experimental
Expires: December 13, 2013

P. Porambage
P. Kumar
A. Gurtov
M. Ylianttila
E. Harjula
CWC, University of Oulu
June 11, 2013

Certificate based keying scheme for DTLS secured IoT
draft-pporamba-dtls-certkey-00

Abstract

The IP-based Internet of Things (IoT) stands for the universal interconnection of smart objects and back end users with the help of IP protocols. Secure key management among the smart objects is an important aspect of IoT security. Due to the high levels of resource constraints of the devices in terms of memory, battery capacity and CPU power, and other network characteristics such as mobility, scalability, heterogeneity and limited bandwidth, the conventional security protocols cannot be directly deployed in IoT networks in their raw formats. We propose a lightweight DTLS-based keying mechanism for CoAP IoT smart objects which supports the scalability of the network and node mobility. The protocol consumes less device resources and minimum network bandwidth by incurring low message overhead. The smart objects can securely access the network and obtain certificates after an initial configuration irrespective of the manufacturer standards.

Requirements Language

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

1. Introduction

The IP-based Internet of Things (IoT) will enable smart objects to communicate among each other and with backend users of the Internet during different activities such as sensing, controlling, smart metering and etc. Due to the massive size networks and resource constrained nodes, IoT networks are inherently vulnerable to security attacks. In each of the application scenarios a secure object authorization mechanism and a common session key is needed between two parties for trustworthy data communication.

IoT networks and the network devices have several specific characteristics. Mainly the devices are tightly resource constrained in terms of memory, battery capacity, CPU power and bandwidth. Therefore, the standard expensive IP-based protocols cannot be deployed in such networks and inexpensive communication protocols are required. Currently IETF is contributing to the development of lightweight protocols for Low-power Lossy IoT networks. E.g. IPV6 over Wireless Personal Area Networks (6LoWPAN) and Constrained Application Protocol (CoAP). Likewise security protocols have been introduced such as DTLS, HIP-DEX and light versions of EAP. However they are still in an immature stage and not standardized correspondingly TLS, IKE or IPSec. The network can be comprised of heterogeneous devices which are manufactured by different vendors with different specifications. Therefore, it is quite challenging to define a common security protocol that is compatible with all the device specifications. The devices can also be mobile and application specific. The size of the network might be varying from hundreds to billions of nodes.

In this draft we propose a secure network access and a key management scheme for resource restricted IoT networks. Furthermore, we analyze how the new protocol supports mobility of the devices and scalability of the network. Secure network access enables the nodes to obtain authorized identity from a trusted root. The two-phase solution is formulated with Datagram Transport Layer Security (DTLS) handshaking protocol. The certificate generation and key establishment are based on Elliptic Curve Cryptography (ECC) arithmetic. The rest of the Internet-draft is organized as follows. Section 2 gives some related work and background about DTLS secured IoT networks. Section 3 describes the use cases and the problem statement. Section 4 presents the proposed security scheme. Finally, Section 5 concludes the proposed IoT security solution with future improvements.

2. Related Work and Background

DTLS protocol is a lighter version of Transport Layer Security (TLS) protocol which runs on unreliable User Datagram Protocol (UDP) connections [RFC6347]. Though DTLS uses similar messages as TLS handshaking it has some internal mechanisms to withstand against DoS attacks, replay attacks, packet losses and packet reordering. Therefore, DTLS is proposed as the main security binding for Constrained Application Protocol (CoAP) [I-D.ietf-core-coap]. Basically the DTLS secured CoAP has three modes of security.

PreSharedKey: A list of pre-shared keys is deployed in the network nodes. When a connection is formed with a new node, the system selects the appropriate key based on the new node and establishes a DTLS session using DTLS PSK mode. This implementation is mandatory to consider cipher suite TLS_PSK_WITH_AES_128_CCM_8 as specified in [RFC6655] and security considerations of [RFC4279].

RawPublicKey: The DTLS enabled devices have asymmetric key pair without an X.509 certificate. The raw public keys are pre-configured in the devices in accordance to the cipher suite TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 as specified in [I-D.mcgrew-tls-aes-ccm-ecc], [RFC5246], [RFC4492]. Each smart object calculates an identifier based on its public key. The identifiers are used to associate the endpoints with further device information and to perform access control.

Certificate: The DTLS enabled devices have asymmetric key pair with an X.509 certificate. The certificates are issued and signed by a common trust root. Sometimes a device might have one or several certificates issued by more than one certificate authority. When a device is forming a new connection with a remote device, the certificates should be verified.

The last two phases of DTLS based security modes are more dynamic and scalable. Since the nodes might be manufactured by different vendors with different specifications, it is yet an open issue to bring the security solution to a common platform. However the use of X.509 certificates is still quite expensive for resource constrained network devices such as tiny sensors, actuators and smart home appliances. Instead of using a costly explicit certificate scheme, it will be highly appropriate to replace with an implicit certificate scheme which consumes fewer resources and induces low network overhead. The same certificates are to be utilized in pairwise key establishment between CoAP nodes. Though, DTLS is considered a lighter and robust security solution, the number of message transfers to establish the secure connection (i.e. 12 messages) still introduces a large communication overhead. In [I-D.garcia-core-security-05], the authors have presented the most significant security considerations in the IP-based Internet of Things. The internet-draft [I-D.keoh-lwig-dtls-iot] proposes pervasive security architecture for the IoT in order to provide network access control to smart devices, the management of keys and securing unicast/multicast communication.

3. Use Cases and Problem Statement

Our work aims an IoT network running on 6LoWPAN/CoAP enabled smart network devices. The network devices can be stationary or mobile, battery powered and highly resource constrained in terms of memory and CPU power. The communication links might have bandwidth limitations too. In applications such as smart power metering, health monitoring and smart home, the IoT network is connected to the public internet through a number of 6LoWPAN border routers (6LBR). In defining this key establishment protocol, we consider the 6LBR is performing as the controlling entity of the IoT network. For instance take into account a particular scenario of a smart building where the lighting devices, window panes and air condition machines are controlled, monitored and billed by a central authority. The functionality of each device is controlled by the central node, based on the sensed data related to the network.

3.1. Problem Statement and Requirements

As explained in the previous section, DTLS plays a prominent role in CoAP-based IoT security. Nevertheless, it produces a reasonable message overhead to low-power lossy networks. If DTLS is not used, however, there should be a strong and inexpensive secure key establishment protocol for IoT networks. The demand for a secure lightweight keying mechanism is significant for both DTLS and non-DTLS secured IoT networks. The utilization of implicit certificates as a replacement for X.509 certificates will also be a low-cost solution. Therefore we identify two main problems with security in CoAP-based IoT networks.

- A new joining device must have secure and authorized identity to perform as legitimate nodes in the network. The nodes can claim their legitimacy by having implicit certificates granted by a common trust root (e.g. 6LBR).

- Lightweight pairwise key establishment is mandatory for mutual communication between nodes or nodes and back end internet users. The two entities should be able to use the certificates as an implicit assurance for being legitimate users of the particular network.

Additionally the solution requires being scalable and supporting mobility and heterogeneity of the network devices. Since the network might contain thousand to billions of nodes, the solution should be easily extensible. Furthermore, since the devices have to be accessed and controlled via standard IP protocols, the authorized identification should be IP supportive.

3.2. Security Requirements

We consider the Internet Threat Model in [RFC3552] where a malicious attacker can read and modify the network traffic while transmitting between devices. However, it is assumed that the devices themselves are protected and not exposed to node capture or compromising attacks.

The security scheme should be lightweight as well as strongly secured. PKC based schemes are more secured than symmetric key algorithms. Elliptic Curve Cryptography (ECC) which is an inexpensive alternative for PKC is to be used for protocol design. Random numbers are supposed to be generated as given in [NIST-800-108].

4. Design

This section provides a brief overview of the design of the protocol which consists of two phases as (i) secure network access and certificate receipt (ii) secure pairwise key establishment between communicating nodes.

4.1. Overview

The first phase associates with a new node accesses to a secure network and obtains its authorized certificate and private key construction data for deriving its own private-public key pair. When a new node is added to the network with an initial configuration of cryptographic primitives it has to generate a certificate request to the certificate authority (CA) (i.e. 6LBR). Since the smart devices have limited transmission they might not be able to access the trust root or 6LBR in single hop. Therefore the certificate requests can be sent as multiple hops by means of relaying devices. Since 6LBR is a resource rich device, it can directly transmit the certificates to all the smart devices within the network.

The second phase of the protocol supports to establish secure traffic encryption keys between any two legitimate nodes which can prove their authenticity using the certificates and other cryptographic primitives. Even a back end user of the traditional internet can establish pairwise keys with smart devices in the IoT network. However, the users should obviously possess the security parameters (i.e. certificate issued by the same trust root) similar to the other nodes it the particular IoT network. In such scenarios, the back end users also have to access the corresponding border router initially. Afterwards the secure communication can be established according to DTLS Certificate mode as explained in section 4.3.

4.2. Hash Function Selection

During both phases, a cryptographic hash function has to be used. As specified in [SEC4], hash function selection should be carefully done for low-power devices and their security algorithms. The use of SHA-1 is not recommended anymore due its security collapse shown by Wang, [Collisions-SHA1]. SHA-2 and SHA-3 functions induce a high processing overhead and memory footprint on devices which are not affordable by resource constrained network devices. In [I-D.ietf-suiteee], the author has proposed a suitable block cipher based hash function for resource constrained devices. The motivation is to use a hash function with reduced codes size, suitable for hardware implementation, reduced computational cost and less energy consumption, however with strong security. As explained in [I-D.ietf-suiteee], AES-MMO (Matyas-Meyer-Oseas) hash function provides a reasonable level of security with less resource consumption. Specifically it supports the hardware specifications of

IEEE 802.15.4 standard including AES encryption. AES-MMO provides 128-bit security level and MD-strengthening padding scheme is used for existing deployments in ZigBee Smart Energy applications which reduces padding on small messages. However, the use of AES-MMO hash function for real-time implementation requires careful considerations.

4.3. Certificate Generation

DTLS message exchange for secure network access has to be performed when a new node is joining to an existing IoT network. The certificate generation is inspired by the ECQV implicit certificate scheme presented in [SEC4]. First the node should be pre-installed with several security parameters namely, Elliptic Curve (EC) domain parameters (q, a, b, G - base point generator with n order, q - a prime), authentication key K, CAs public key (Q_CA) and a valid IPV6 address. K is common to all the smart objects and trust root of the network. Then the node can be located in the network and start exchanging messages with the corresponding trust root (i.e. CA). The simple timeout and retransmission scheme with the standard DTLS state machine is also applicable to this handshaking too.

Initially the client (i.e. smart device) sends the Client Hello message and upon receiving the message, the server (i.e. CA) verifies the message and responds with a HelloVerifyRequest. After the client verifies the server Hello message successfully, it generates a random integer r_U and true nonce N_U, creates a certificate request (EC point) R_U and sends to the server the certificate request along with its IPV6 address and MAC value.

Upon receiving the certificate request, the server checks the legitimacy of IPV6 address and verifies the MAC value. If both are successful, the server computes public key reconstruction data P_U for the client, using the request point R_U. Then the certificate is generated as an encoded version of P_U, client IPV6 address and time stamp T. The server computes an integer (r_U) value for calculating client private key construction value s. Hash value of the certificate is computed during this stage. The selection of hash function is described in section 4.1.1. CA private key (d_CA) is utilized while calculating s.

On receiving the certificate and private key construction integer, the client first verifies the integrity of the message using the MAC and analyses the certificate for further verifications.

The client calculates private key (d_U) and public key (Q_U) as depicted in Figure 1. During this stage, the client computes its public key by two mechanisms for authenticating whether the certificate is granted by the given trusted root. If the verification is successful, the client sends Finish message to the server. Finally, the server concludes the handshaking with the Server Finished message.

```
          Client                                        Server
         (Node U)                                        (CA)
         --------                                     --------

         Client Hello        ---------->
                             <---------- HelloVerifyRequest

         Certificate Request
         Generation
         generate r_U
         R_U = r_U * G
         Generate N_U
         Calculate
         MAC[R_U, U, N_U]

         Certificate Request
         R_U, N_U, U, MAC    ---------->
                                         Check validity of U
                                         Verify MAC
                                         Generate r_CA
                                         P_U = R_U + r_CA * G
                                         Cert_U = {U, P_U, T}
                                         e = H(Cert_U)
                                         s = e*r_CA + d_CA (mod n)
                                         Generate N_CA
                                         Calculate
                                         MAC [Cert_U, s, N_CA]
                                         Message
                             <---------- Cert_U, s, N_CA, MAC
         Verify MAC
         Analyze Cert_U
         e = H(Cert_U)
         d_U = e*r_U + s (mod n)
         Method 1: Q_U = d_U*G
         Method 2: Q1_U = eP_U + Q_CA
         Verify Q_U == Q1_U

         ClientFinished      ---------->
                             <---------- ServerFinished
```

Figure 1

4.4. Secure Pairwise Key Establishment

When the IoT network nodes possess valid certificates and public-private key pairs, they are in a position to communicate equivalently in the Certificate mode of DTLS secured CoAP. Two smart devices can setup a secure communication channel along with a pairwise key establishment for traffic encryption as illustrated in Figure 2. Here we consider the initiator node as the client and the responder node as the server.

```
      Client                                        Server
     (Node U)                                  Node V)
    --------                                   --------

    Client Hello        ---------->
                        <---------- HelloVerifyRequest

    Generate N_U
    Calculate MAC [Cert_U, U, N_U]
    Key Establishment Request
    Cert_U, N_U, U, MAC ---------->
                                    Check validity of U
                                    Verify MAC
                                    e = H(Cert_U)
                                    Q_U = eCert_U + Q_CA
                                    Generate N_V
                                    Calculate MAC[Cert_V, V, N_V]
                                    Response
                        <---------- Cert_V, N_V, V, MAC
    Verify MAC                      K_UV = d_V*Q_U = d_V*d_U*G
    e = H(Cert_V)
    Q_V = eCert_V + Q_CA
    K_UV = d_U*Q_V = d_U*d_V*G

    ClientFinished      ---------->
                        <---------- ServerFinished
```

Figure 2

The initial handshake is performed between the client and the server by exchanging Hello messages according to the standard DTLS protocol. The client node (U) chooses a true random nonce NU and broadcasts it along with Cert_U, IPV6 address U and MAC[Cert_U, N_U, U]. Similarly in Phase I, MAC is appended for the initial authentication. Once the server node (V) receives the message, it verifies the MAC. If the verification succeeds, it can ensure that U is an authenticated user.

Furthermore, V can have an implicit assurance that U is a legitimate user of the given cluster by computing sender public key QU using Q_CA; e = H(Cert_U) and Q_U = eCert_U + Q_CA. According to the following derivation in Figure 3, the calculation also gives exactly the same Q_U as computed by node U.

```
Q_U =   d_U * G
    =   (er_U + s (mod n)) * G
    =   (er_U + er_CA + d_CA (mod n)) *G
    =   e (r_U + r_CA (mod n)) * G + d_CA * G
    =   e(r_U * G + r_CA * G) + Q_CA
    =   e(R_U + r_CA * G) + Q_CA
    =   eCert_U + Q_CA
```

Figure 3

Then the node V generates a random nonce NV and sends it along with Cert_V , identity V and MAC[Cert_V , N_V , V]. In the meantime V computes the pairwise key K_UV from its private key d_V and Q_U, K_UV = d_V Q_U. Similar to V, upon receiving the message, node U verifies the MAC and if the verification is successful it computes QV and K_UV = d_UQ_V . Therefore, at the end of two way message transferring, both parties can derive a common pairwise key for actual secure communication.

Comparing to the standard ECDH key exchange, our scheme is more secure since it validates the legitimacy of both parties before deriving the final key. Instead of transmitting the public keys in the air, the nodes send their Cert values to derive the public keys (at the other node). This will also implicitly assure the authenticity and legitimacy of smart objects.

Finally the handshaking is concluded by exchanging Finished messages. We assume that DTLS handshaking messages are delivered reliably as explained in [RFC6347].

Likewise, the same handshaking can be performed between a smart device in the IoT network and a backend user in the Internet. However the Internet users should also possess valid certificates from the same trust root.

5. Conclusion and Future Work

In this Internet draft we have proposed a DTLS-based certificate scheme and a secure key establishment for IoT networks. The protocol is lightweight and strongly secured due to the exploitation of ECC arithmetic throughout the entire design. The implementation of the proposed protocol on TelosB mote platform shows the feasibility of deploying it on tightly resource constrained devices. The performance measurements in terms of memory foot-prints (ROM and RAM) and execution time illustrate the necessity of applying optimization techniques for ECC operations.

Our protocol supports the scalability of the network and the topology changes (i.e, location changes or mobility) of the smart objects with in the same IoT network. When a new node is added to the network, a valid node identity, keying information (i.e, K and QCA) and EC domain parameters should be stored. Then, at the bootstrapping phase, the node can send the certificate request and obtain a certificate from the CA for computing its own keys. Therefore, the size of the network is not necessary to be pre-defined during the initial deployment phase. The CA only needs to verify the validity of the sensor node IPV6 identities to issue the certificate. Similarly, the nodes do not need a prior knowledge about their neighbors. Whenever a new node is added to the network or it changes the neighboring set, it can establish the pairwise link keys, with the corresponding neighbors using the certificate.

The certificates always provide an implicit assurance for the nodes, that they are authenticated nodes in the given domain. Irrespective of the location of the devices (within the given IoT network) they can derive the pairwise keys securely without previous awareness of the new communicating nodes. If the pairwise keys between communicating nodes (i.e. node to node or node to Internet user) are pre-installed, there should be a large number of stored keys per node, which may not be desirable for large scale networks. However, in our protocol such a large scale key pre-installation is not needed. Bandwidth utilization is also preserved by restricting two message transactions for both certificate generation and key establishment scenarios.

In future we intend to extend the certificate generation phase for enabling secure multicast operations. Afterwards the multicast keys are to be established to protect CoAP messages on top of IP multicast. Moreover the protocol is to be extended for further supporting the mobility of nodes and scalability of the network. Similar implicit certificates can be incorporated with other pervasive secure handshaking mechanisms such as HIP-DEX for securing IoT networks.

6. IANA Considerations

7. Security Considerations

This document discusses different design aspects of DTLS based secure key establishment scenarios. This document is entirely focused on security.

8. Acknowledgements

The authors would like to thank Zach Shelby for his valuable comments and suggestions.

9. References

9.1. Normative References

[Collisions-SHA1]
Wang, X., Yin, Y., and H. Yu, "Finding Collisions in the Full SHA-1", in Proceedings of Crypto, 2005.

[I-D.garcia-core-security-05]
Garcia-Morchon, O., Keoh, S., Kumar, S., Hummen, R., and R. Struik, "Security Considerations in the IP-based Internet of Things", draft-gracia-core-security-05 (work in progress), March 2013.

[I-D.keoh-lwig-dtls-iot]
Keoh, S., Kumar, S., and O. Garcia-Morchon, "Securing the IP-based Internet of Things with DTLS", dtraft-keoh-lwig-dtls-iot-01 (work in progress), February 2013.

[I-D.mcgrew-tls-aes-ccm-ecc]
McGrew, D., Bailey, D., Campagna, M., and R. Dugal, "AESCCM ECC Cipher Suites for TLS", draft-mcgrew-tls-aes-ccm-ecc-06 (work in progress) (work in progress), February 2013.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, July 2003.

[RFC4279] Eronen, P. and H. Tschofenig, "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", RFC 4279, December 2005.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

9.2. Informative References

[I-D.ietf-core-coap]
Shelby, Z., Hartke, K., Bormann, C., and B. Frank, "Constrained Application Protocol (CoAP)", draft-ietfcore-coap-12(work in progress) (work in progress), October 2012.

[I-D.ietf-suiteee]
Campagna, M., "A Cryptographic Suite for Embedded Systems (SuiteE)", draft-campagna-suitee-04 (work in progress), October 2012.

[NIST-800-108]
National Institute of Standards and Technology , "NIST SP 800-108, Recommendation for Key Derivation Using Pseudorandom Functions", NIST Special Publication 800-108, , <http://csrc.nist.gov/publications/nistpubs/ 800-108/sp800-108.pdf>.

[RFC4492] Blake-Wilson, S., Bolyard, N., Gupta, V., Hawk, C., and B. Moeller, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)", RFC 4492, May 2006.

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

[RFC6655] McGrew, D. and D. Bailey, "AES-CCM Cipher Suites for Transport Layer Security (TLS)", RFC 6655, July 2012.

[SEC4] Standards for Efficient Cryptography Group , "Elliptic Curve Qu-Vanstone Implicit Certificate Scheme (ECQV), v0.97", SEC 4, March 2011, <http://www.secg.org/download/aid-785/sec4-0.97.pdf>.

Authors' Addresses

Pawani Porambage
CWC, University of Oulu
P.O. Box 4500, FI-90014
Oulu
Finland

Phone: +358 8 553 2965
Email: pporamba@ee.oulu.fi
URI: http://www.cwc.oulu.fi

Pradeep Kumar
CWC, University of Oulu
P.O. Box 4500, FI-90014
Oulu
Finland

Phone: +358 8 553 2965
Email: pkumar@ee.oulu.fi
URI: http://www.cwc.oulu.fi

Andrei Gurtov
CWC, University of Oulu
P.O. Box 4500, FI-90014
Oulu
Finland

Phone: +358 40 596 3729
Email: gurtov@ee.oulu.fi
URI: http://www.cwc.oulu.fi

Mika Ylianttila
CWC, University of Oulu
P.O. Box 4500, FI-90014
Oulu
Finland

Phone: +358 40 535 0505
Email: mika.ylianttila@ee.oulu.fi
URI: http://www.cwc.oulu.fi

Erkki Harjula
CWC, University of Oulu
P.O. Box 4500, FI-90014
Oulu
Finland

Phone: +358 40 535 0505
Email: erkkih@ee.oulu.fi
URI: http://www.cwc.oulu.fi