Aalto University
School of Science
Degree Programme in Security and Mobile Computing

Shailesh Mota

Secure Certificate Management and Device Enrollment at IoT Scale

Master's Thesis
Espoo, June 30, 2016

Supervisors: Professor Tuomas Aura, Aalto University
Professor Elena Dubrova, KTH Royal Institute of Technology
Instructor: Rajat Kandoi M.Sc. (Tech.), Ericsson, Finland

Aalto University
School of Science
Degree Programme in Security and Mobile Computing

ABSTRACT OF
MASTER'S THESIS

Author: Shailesh Mota

Title:
Secure Certificate Management and Device Enrollment at IoT Scale

Date: June 30, 2016 **Pages:** 77

Major: Security and Mobile Computing **Code:** T3011

Supervisors: Professor Tuomas Aura
Professor Elena Dubrova

Instructor: Rajat Kandoi M.Sc. (Tech.), Ericsson, Finland

The Internet of Things (IoT) is expected to comprise of over 20 billion devices connected to the Internet by the year 2020, and support mission critical applications such as health care, road safety and emergency services to name a few. This massive scale of IoT device deployment, heterogeneity of devices and applications, and the autonomous nature of the decision making process introduces new security requirements and challenges. The devices must be securely bootstrapped in to the network to provide secure inter–device communication and also, the applications must be able to authenticate and authorize these devices to provide the relevant services.

In today's Internet, Public Key Infrastructure (PKI) is widely used to provide authenticity, encryption and data integrity during network communication through the use of digital certificates. This thesis investigates the key aspects for deploying a PKI security solution in an IoT ecosystem, ranging from deploying certificates on new devices (bootstrapping) to complete life cycle management of these certificates. We believe that the current PKI can be, with suitable enhancements, used to provide the efficiency, scalability and flexibility needed for IoT security. This thesis provides a survey of key aspects for deploying PKI security solution in IoT ecosystem. We investigate different certificate management protocols and motivate the applicability of enhanced security over transport (EST) protocol for IoT PKI solution. In addition, we propose a PKI deployment model and the bootstrap mechanism to bring up an IoT device and provision it with a digital certificate. Furthermore, we provide a prototype implementation to demonstrate certificate enrollment procedure with an EST server.

Keywords: IoT, PKI, EST, bootstrapping

Language: English

Aalto-universitetet
Högskolan för teknikvetenskaper
Utbildningsprogrammet för säker och mobil kommunikation

SAMMANDRAG AV
DIPLOMARBETET

Utfört av: Shailesh Mota

Arbetets namn:
Säkra Certifikathantering och Device Inskrivning på sakernas Skala

Datum:	Den 30 Juni 2016	**Sidantal:**	77
Huvudämne:	Säker och mobil kommunikation	**Kod:**	T3011
Övervakare:	Professor Tuomas Aura Professor Elena Dubrova		
Ohjaaja:	Rajat Kandoi M.Sc. (Tech.), Ericsson, Finland		

Sakernas Internet (IoT) förväntas bestå av mer än 20 miljarder enheter som är anslutna till Internet år 2020, och stödja verksamhetskritiska applikationer såsom hälsovård, trafiksäkerhet och räddningstjänsten för att nämna några. Denna massiva omfattningen av sakernas enhet driftsättning, heterogenitet enheter och applikationer, och den självständiga karaktären av beslutsfattandet införs nya säkerhetskrav och utmaningar. Produkterna skall vara säkert stroppad in på nätet för att ge säker inter - enhet kommunikation och även måste ansökningarna kunna autentisera och auktorisera dessa enheter för att tillhandahålla de aktuella tjänsterna.

I dagens Internet, är Public Key Infrastructure (PKI) används i stor utsträckning för att tillhandahålla äkthet, kryptering och dataintegritet under nätverkskommunikation genom användning av digitala certifikat. Denna avhandling undersöker de viktigaste aspekterna för att distribuera en PKI säkerhetslösning i ett sakernas ekosystem, från distribuera certifikat på nya enheter (bootstrapping) för att slutföra livscykelhantering av dessa certifikat. Vi anser att den nuvarande PKI kan vara, med lämpliga förbättringar, som används för att ge effektivitet, skalbarhet och flexibilitet som krävs för sakernas säkerhet. Denna avhandling ger en översikt över de viktigaste aspekterna för distribution PKI säkerhetslösning i sakernas ekosystem. Vi undersöker olika protokoll certifikat förvaltning och motivera tillämpligheten av förbättrad säkerhet över transporter (EST) protokoll för sakernas Internet PKI-lösning. Dessutom föreslår vi en PKI distributionsmodell och bootstrap mekanism för att ta upp en sakernas internet enheten och tillhandahållandet det med ett digitalt certifikat. Dessutom erbjuder vi en implementering prototyp att visa certifikatregistreringsförfarande med en EST-server.

Nyckelord: IoT, PKI, EST, bootstrapping

Språk: Engelska

Acknowledgements

I will like to acknowledge my supervisors for their guidance, motivation and support. The thesis is an outcome of the research insights provided by Professor Tuomas Aura. The discussions with Professor Tuomas were highly constructive and drove the thesis to its conclusion. Professor Elena Dubrova provided useful tips for finalising the structure of the thesis. I will like to thank my friend and instructor Rajat Kandoi for highly productive brainstorming and cooking sessions. Moreover, I will like to thank Jeanne for keeping me motivated to finish the work in time. Finally, thanks to my parents for their continuous support and belief in me.

Espoo, June 30, 2016

Shailesh Mota

Abbreviations and Acronyms

3GPP	3rd Generation Partnership Project
6LoWPAN	IPv6 over Low power Wireless Personal Area Networks
ASN.1	Abstract Syntax Notation One
CA	Certification Authority
CBOR	Concise Binary Object Representation
CMP	Certificate Management Protocol
CMC	Certificate Management Messages over CMS
CMS	Content Management System
CoAP	Constrained Application Protocol
CP	Certificate Policy
CRL	Certificate Revocation List
CRMF	Certificate Request Message Format
CSR	Certificate Signing Request
DAD	Duplicate Address Detection
DER	Distinguished Encoding Rules
DHCPv6	Dynamic Host Configuration Protocol Version 6
DN	Distinguished Name
DNS	Domain Name System
DNSSL	DNS search list
DTLS	Datagram Transport Layer Security
ECC	Elliptic Curve Cryptography
EST	Enrollment over Secure Transport
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
IEC	International Electrotechnical Commission
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange
IoT	Internet of Things
JAX-RS	Java API for RESTful Web Services

JSON	JavaScript Object Notation
MAC	Media Access Control
MHT	Merkle Hash Tree
MIME	Multipurpose Internet Mail Extensions
NGE	Next Generation Encryption
NI	Node Information
NIST	National Institute of Standards and Technology
OCSP	Online Certificate Status Protocol
OWHF	One Way Hash Funtions
PGP	Pretty Good Privacy
PKI	Public Key Infrastructure
PoP	Proof Of Possession
PSN	Public Services Networks
PTC	Positive Train Control
RA	Registration Authority
RA	Router Advertisement
RDI	Revocation Data Issuer
REST	Representational State Transfer
RPKI	Resource Public Key Infrastructure
SCEP	Simple Certificate Enrollment Protocol
SCP	Secure Copy
SCT	Signed Certificate Timestamp
STH	Signed Tree Hash
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
VPN	Virtual Private Network
XML	Extensible Markup Language

Contents

Chapter 1

Introduction

The Internet of Things (IoT) is a growing network of embedded autonomous devices or sensors with an IP address for Internet connectivity. The interconnected system between IoT and other Internet capable system promises improved quality of life, increased efficiency and greater economic growth. Gartner's recent reports on IoT devices indicate that there are more than 3 billion IoT devices available in the smart environment at present and there is a continuing interest in the market towards adoption of IoT devices. However, Cyber threats are prevalent against IoT deployments, the number of attacks and tools available to potential attackers are becoming more efficient. This rise in the number of connected devices has created a pressing demand for robust security and authentication mechanisms. The thesis focuses on establishing trust across connected devices in large scale IoT deployments.

1.1 Problem statement

The rise in the number of connected IoT devices has lead to increase in the challenges to secure inter–device communication. Moreover, recent attacks on IoT environments demonstrate a lack of generic security architecture in operation of IoT devices. Public key infrastructures (PKI) is the trusted security framework which provides authenticity, encryption and data integrity to secure current Internet. In this thesis, we answer the question "What are the various aspects to deploy a flexible, scalable, efficient and trustworthy PKI for IoT environment ?". We investigate key aspects, challenges and the importance of deploying a PKI security solution for IoT ecosystem both from a theoretical and an engineering perspective.

1.2 Motivation

The work done in this thesis is motivated by the following factors :

1. Certificate management protocols facilitate PKI operations such certificate enrollment and revocation, private key generation, key renewal and update. It is worth investigating the various existing certificate management protocols and choosing the suitable for IoT device deployment scenario.

2. Cyber threats are prevalent against IoT deployment scenario leading to disabling of system operation, economic loss and threat to privacy of sensitive user data. It is worth reviewing security attacks on various IoT deployment scenarios.

3. PKI is designed to provide secure communication between client–server architecture of current Internet. It is important to understand different challenges and aspects to adopt PKI solution for providing secure communication in IoT deployment scenario.

4. Certificate management system employed for PKI in IoT deployments should support billions of certificate operations. It is significant to examine a scalable data structure solution to support certificate operations.

5. Certificate provisioning addresses identification of devices based on digital certificates. It is appropriate to investigate different aspects for bootstrapping an IoT device to provision it with a digital certificate.

6. Compromised certificate authority (CA) can end–up issuing unauthorised SSL certificates. This way an attacker can impersonate a web server and extract sensitive user data. It is worth studying how existing solutions can be used to introduce trustworthiness of CAs by making CA operations transparent in a PKI deployment.

1.3 Contribution

This thesis provides a survey of the key aspects and the importance of deploying a PKI security solution for IoT ecosystem. We examine various existing protocols and mechanisms for certificate management including SCEP, EST, CMC and CMP. We propose the use of EST protocol for secure certificate provisioning of IoT devices. Further, we motivate the applicability

of EST protocol for resource constrained environments. We identify various challenges in deploying PKI solution for IoT environment. Furthermore, we discuss various cyber attacks against IoT deployment scenario leading to disabling of system operation, economic loss and threat to sensitive user data. Moreover, we propose a PKI deployment model for IoT environment and the bootstrap mechanism to bring up an IoT device and provision it with X.509 digital certificate which ensures device authenticity, data confidentiality and integrity. In addition to the literature survey, we provide a proof-of-concept implementation which demonstrates certificate enrollment procedure with an EST server.

1.4 Structure of the Thesis

The rest of the thesis is organized as follows: Chapter 3 provides the background information about certificate management protocols including SCEP, EST, CMC and CMP. We motivate the adoption of EST protocol for resource constrained environments. Chapter 4 provides details on different IoT deployment scenarios and reviews recent attacks in IoT environments. Furthermore, we investigate various challenges and aspects to be considered in deploying PKI solution for IoT environment. Chapter 5 examines different schemes for naming and identification of IoT devices. Chapter 6 gives necessary details about merkle hash trees (MHT) which is a scalable data structure solution to support certificate operations performed by a CA. Chapter 7 contains our proposal for the bootstrap mechanism to bring up an IoT device and provision it with digital certificate. Chapter 8 explores application layer protocol used in resource–constrained environments. In addition, we demonstrate a prototype implementation for certificate enrollment procedure with an EST web–server. Chapter 9 discusses the feasibility of applying certificate transparency to increase the trustworthiness of CA operations. Chapter 10 summarizes the thesis and provides concluding remarks.

Chapter 2

Background on Public Key Infrastructure

PKI is a model wherein a set of roles, policies and procedures are needed for the creation, management, storage and revocation of digital certificates to facilitate public–key encryption [106]. The main purpose of PKI model is to enable secure communication between different entities involved in network activities. Some such activities are e–commerce, Internet banking and secure electronic mail. PKI plays a critical role in secure communication where password based authentication methods are not sufficient and a more thorough procedure is required to confirm the identities of communication parties and validation of information transferred over the network.

PKI is built on public key cryptography and digital certificates. It is an arrangement which binds the public key with the identities of different network entities such as a user, client machine or a server. The binding procedure is established with the help of a CA. CA is responsible for registration and issuance of digital certificates for the requesting clients or network entities. This in–turn allows clients to learn the public keys of other clients and hence, communicate securely.

To illustrate this, we assume there are two network entities Alice and Bob who want to securely communicate with each other. The CA is considered to be a trusted node which has a self-signed digital certificate. Furthermore, digital certificate is a data structure which binds name with public key. In this scenario, if Bob trusts a particular CA and has the public key of the CA he can further securely know the public key of Alice. To achieve this Bob has to obtain a digital certificate signed by the CA that certifies the public key of Alice. In some scenarios, Bob may not be pre–configured with the public key of CA that certified Alice's public key, hence, Bob has to obtain a chain of certificates. For example, Bob knows the public key of CA1 and hence

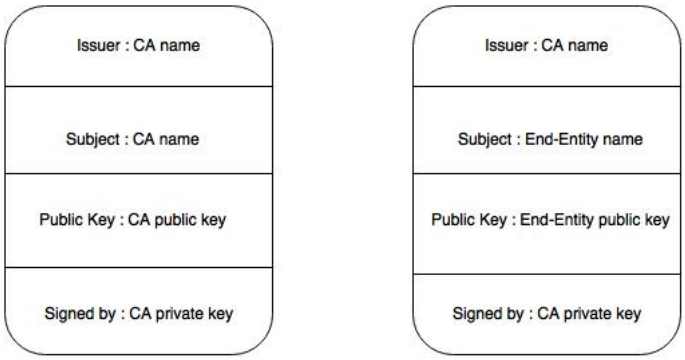

Figure 2.1: Fields in a digital certificate.

trusts CA1. Alice's public key is signed by CA3. CA3's public key is signed by CA2 and CA2's public key is signed by CA1 [89]. Once Bob obtains this chain of certificates from Alice he can trust the public key of Alice and start secure communication with her.

Figure 2.1 depicts the fields and the values used in a digital certificate. Every CA has a self-signed certificate, hence, the issuer and subject fields of certificate have the same value which is the distinguished name of the CA. Public key field has the public key of the CA and the certificate is signed using the private key of the CA. Any client certificate issued by a CA has the distinguished name of the client in the subject field and the issuer field has CA's distinguished name. The public key of the client is inserted in the public key field and the certificate is signed with the private key of the CA.

2.1 Roles in PKI model

In a generic PKI model there are multiple entities involved. Below we describe different roles in a PKI model [30, 64]:

1. Certificate Authority (CA) : The CA is responsible for storing, issuing and digitally signing the certificates.

2. Registration Authority (RA) : The RA is responsible for verifying the identity of entities requesting a digital certificate from a CA. The RA is responsible for forwarding the issuance request to the CA.

3. Central directory : A secure database or repository for storing and indexing keys.

4. Certificate Policy (CP) : A CP is a document stating different roles and duties of a PKI. In case of X.509 digital certificates, a specific field can be set to indicate associated CP [34]. This value indicates the assurance level associated with a certificate. Hence, any relying party can decide the level of trust to put in the certificate.

5. Certificate management system (CMS) : A CMS is used for performing tasks such as managing access to stored certificates and delivering certificates to be further issued to the requester [74].

Chapter 3

Background on Certificate Management Protocols

PKI is widely used to authenticate the identity of end–points such as users, devices and applications in the form of digital certificates. Digital certificates are being increasingly deployed, X.509 certificates [36] serve as the basis for authentication in the Internet Engineering Task Force (IETF) standards such as The Internet Key Exchange (IKE) [52] and IKEv2 [60], Virtual Private Networks (VPNs) [27] and Transport Layer Security Protocol (TLS) [38, 39, 41]. A CA needs a certificate issuance mechanism or a protocol for granting X.509 certificates to the end–points. Such protocols fall under the category of certificate management protocols. Through certificate management protocols a PKI client can issue requests such as certificate issuance, certificate renewal, certificate revocation, etc., to the CA. Moreover, the protocol enables PKI client to request certificate revocation status information. However, this functionality is also distinctly provided by mechanisms such as certificate revocation lists (CRLs) [114] and online certificate status protocol (OSCP) [82].

Through the efforts of IETF PKI X.509 working group two certificate management protocols have been developed – Certificate Management Protocol (CMP) [98] and Certificate Management Messages over CMS (CMC) [26]. Both the protocols offer essentially the same basic functionality described in Section 3.2 and Section 3.3. However, CMP is more comprehensive and widely deployed [15] of the two. Moreover, over the years Cisco has developed two other protocols – Simple Certificate Enrollment Protocol (SCEP) [51] and Enrollment over Secure Transport (EST) [90]. EST and SCEP facilitate certificate provisioning and enrollment. SCEP is the evolution of enrollment protocol which is widely supported in both client and CA implementations.

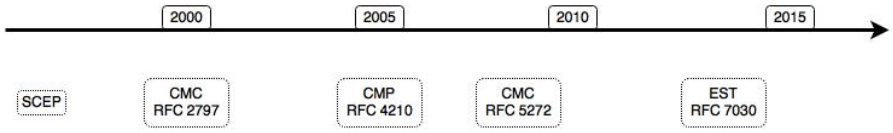

Figure 3.1: Time-line showing certificate management and certificate enrollment standards over the years.

Figure 3.1 shows the time–line for creation of the aforementioned standards.

3.1 SCEP

SCEP is a PKI communication protocol which utilizes PKCS#7 and PKCS#10 messages that are sent over Hypertext Transfer Protocol (HTTP). It is the most widely deployed protocol for certificate requests and CRL queries. However, the protocol does not support certain certificate management features such as in-band certificate revocation transactions. To support more comprehensive functionality the IETF protocol suite provides two other certificate management protocols – CMP and CMC. SCEP is not a formal standard or RFC; it is a draft which has already expired and has been published as historic. The protocol supports operations such as public key distribution through CA and RA, certificate enrollment, certificate query and CRLquery [51].

The SCEP draft defines the following entity types :

1. Requester : The requester is the client or end-point for SCEP message exchange. It is allowed to submit SCEP messages for itself or on behalf of peers. Before starting a PKI transaction, a requester is required to have an appropriate RSA key pair. This key is used for signing the SCEP pkiMessage to be sent to the server. SCEP message types are based on PKCS#7 [59] and PKCS#10 [85]. Moreover, a requester must locally configure the IP address or domain name of the CA server and identity information to be used for authentication of the CA.

2. SCEP CA : A SCEP CA is responsible for signing the client certificates. The issuer field of the certificate is set with the name of issuing CA. Before any PKI transactions can begin, the SCEP CA server obtains a CA certificate which can be self–signed or issued by a higher level CA. This certificate is provided out–of–band to the SCEP requester.

To authenticate the CA certificate the requester can use the fingerprint information in the CA certificate which is obtained after sending a *GetCACert message*. The CA is responsible for answering the CRL queries of the requester and must include *CRLDistributionPoint* information in the certificates it issues. Moreover, the CA server must be a high–availability service for answering CRL queries itself. The CA can also make the certificates available through LDAP [103]. Furthermore, the CA can also enforce policies on the client requests, reject client requests and return previously created certificates for duplicate requests.

3. SCEP RA : SCEP RA server performs authorization and validation checks for the SCEP requester on behalf of CA server. After receiving a *GetCACert* message from the requester, RA performs validation checks on the request and forwards the certification requests to the CA. Moreover, the RA certificate is also included in the response to *GetCACert* message indicating the existence of the RA to the requester. Communication between RA and CA server can be carried out with SCEP messages or other protocols such as CMC [26].

3.1.1 SCEP based certificate enrollment

The certificate enrollment request from the requester has to be authenticated by the CA/RA server before a new certificate can be issued. The SCEP protocol uses public-key cryptography to associate public keys and the identity of the requester. This prevents a *man–in–the–middle attack* and the data between the requester and CA is secured. The communication is secured by using SCEP secure message objects which specifies how PKCS#7 [59] is used for encryption and signing the messages. To perform signing, the requester uses a local certificate obtained from an alternate CA. The CA server may accept or reject the request by looking up at the policy settings . Moreover, the requester can also locally generate a self-signed certificate to perform the signing operation. The SCEP draft does not support client authentication based on a self–signed certificate. Requester authentication can also be performed by utilizing *challengePassword* attribute which is sent as part of the enrollment request by the requester. This attribute has been specified as part of PKCS#9 standard[86]. The *challengePassword* is a shared secret distributed privately (only known to the requester) by the CA server. This attribute uniquely associates an enrollment request with the requester [51].

Figure 3.2 demonstrates certificate enrollment by utilizing *challengePassword* attribute in SCEP. The SCEP CA server does not support certification revocation requests from the requester. Certificate revocation requests can

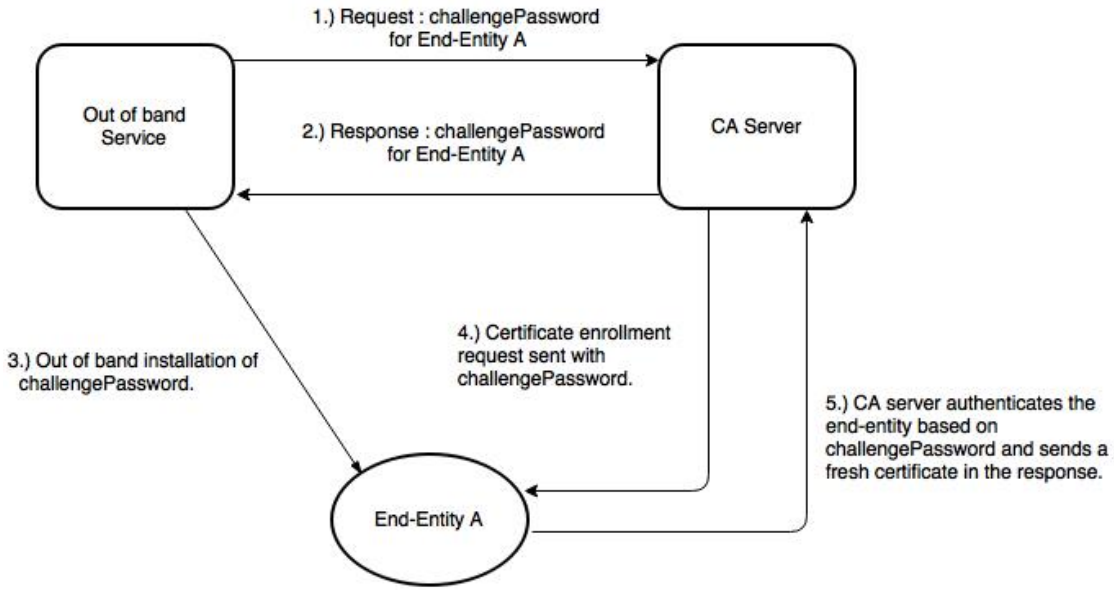

Figure 3.2: Messages exchanged during certificate enrollment in SCEP.

be performed using other certificate management protocols such as CMP and CMC.

3.2 CMP

CMP is a PKI communication protocol used by end–entities to obtain X.509 digital certificates from the CA server. CMP request and response messages are encoded in Abstract Syntax Notation One (ASN.1) using Distinguished Encoding Rules (DER) method. CMP messages are generally encapsulated over HTTP. Other possible means of transport to carry CMP messages are transmission control protocol (TCP) or any connection-oriented transport protocol such as a file over file transfer protocol (FTP) or secure copy (SCP) and through e–mail using multipurpose internet mail extensions (MIME) encoding standard. CMP uses *application/pkixcmp* as the content–type [61]. Through CMP, an end–point can communicate with the CA server to request, revoke, suspend and resume digital certificates. Any number of RAs can be present to mediate the request–response messages between the CA and the end–point.

End–entity certificate request messages need to be authenticated to the CA or RA server. This is know as end–entity message origin authentication.

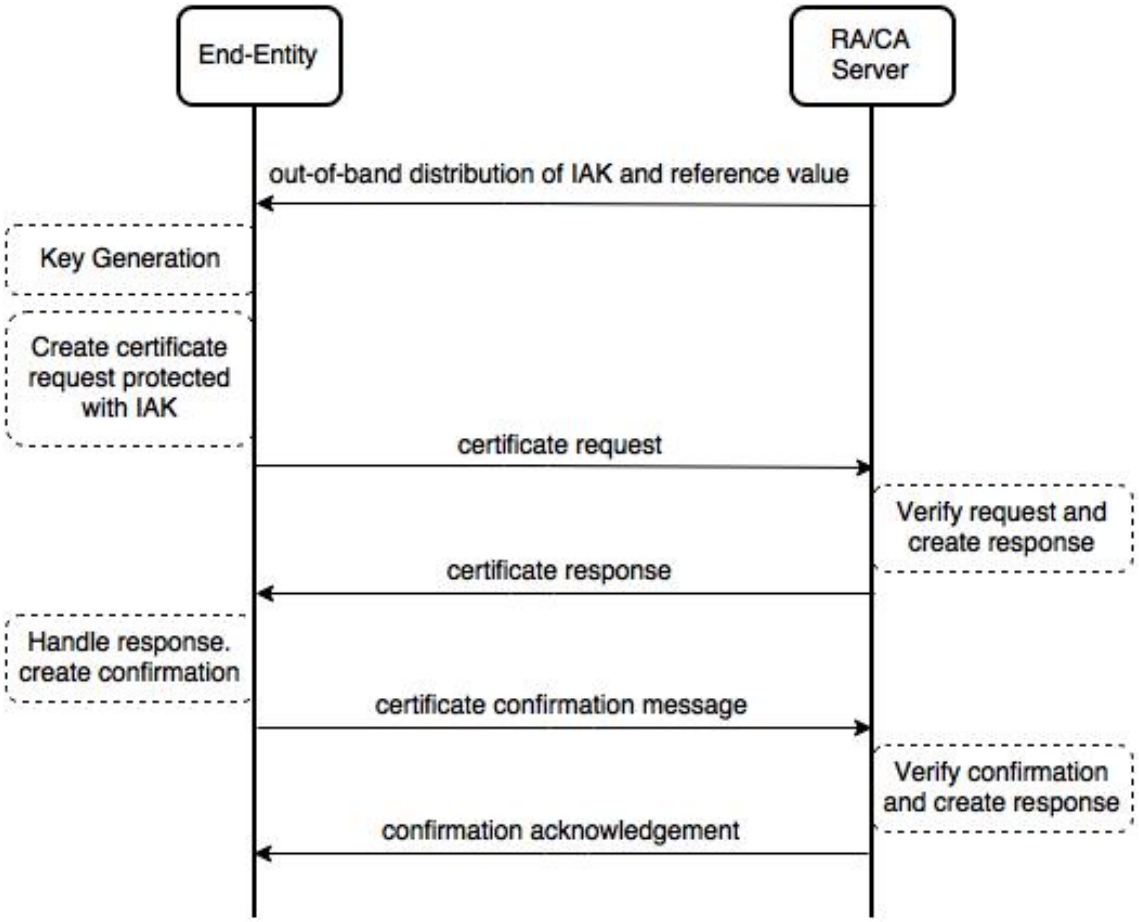

Figure 3.3: Message flow during a certificate request with CMP.

To achieve this the CA or the RA server issues the end–entity with a secret known as *initial authentication key* and a reference value which is used to identify the initial authentication key. Further, the initial authentication key is used to protect relevant PKI messages [98].

Figure 3.3 demonstrates a scenario where an end–entity initiates a new certificate request from the CA server. In this scenario, message authentication of the end–entity is required, *key generation* occurs at the end–entity and a confirmation message is sent by the end–entity after the completion of the transaction.

In case, the verification of confirmation message fails at RA or CA server, the newly issued certificate is revoked immediately.

CMP supports the following PKI management functions [98]:

1. Root CA Initialization : A new root CA produces a self–certificate with the fingerprint. Further, using out–of–band means the end–entities acquire the fingerprint securely and can verify the self–certificate of the newly created root CA.

2. Root CA Key Update : This operations supports periodically updating CA keys.

3. End Entity Initialization : To initialize end–entities two steps are necessary. First, acquiring PKI information. Secondly, verifying public key of root CA. Verification is performed out–of–band.

4. Subordinate CA Initialization : The initialization of a subordinate CA is similar to end–entity initialization. However, the subordinate CAs are also updated with the revocation lists.

5. CRL production : A newly set–up CA must periodically produce versions of each CRL.

6. PKI Information Request : This operation enables PKI entity (CA, RA or an end–entity) to acquire all the information about current status of a CA.

7. Cross Certification : To accomplish this operation the requester CA becomes the subject of the cross–certificate and the responder CA becomes the issuer.

8. Certificate Request : After the end–entity is initialized, it requests the CA for a certificate with a certification request (cr) message. The CA responds back with a newly created certificate after authenticating the end–entity.

9. Key Update : This operation enables an end–entity to request an update for an expiring key pair. The end–entity can send update requests to the CA to issue a new certificate for a new key pair or for the same key pair. The CA returns the new certificate in a key update response message.

10. Revocation operations : Scenarios such as certificate forging require revocation of the forged certificate. In this case, an authorised entity can send a revocation request to the CA server.

3.3 CMC

CMC is a certificate management protocol which is based on the existing cryptographic message syntax (CMS), PKCS#10 [58] and certificate request message format (CRMF) [97] specifications. CMC introduces a way of performing enrollment operations within a single round trip unlike CMP protocol. The protocol is designed such that the key generation occurs at the end–entity. CMC supports all the mandatory algorithms cited by S/MIME standard [91]. Moreover, CMC also supports operations such as transaction management, replay detection through tokens, deferred and pending responses to enrollment requests which requires external methods for issuing a certificate. Transport mechanisms exercised within CMC specification such as HTTP, FTP, SCP, e–mail or TCP are defined in [99]. Architecturally CMC is similar to SCEP, however, the specification supports more options and implements greater algorithmic agility. CMC defines message formats, message control, and data structures for supporting wider range of certificate management operations in comparison to certificate provisioning operations supported by SCEP [26].

An enrollment transaction in CMC is generally completed within a single round–trip. An end–entity sends a PKI enrollment request to the CA server and obtains a PKI enrollment response from the CA. Some exceptional cases such as delayed certificate issuance require more than one roud–trip time. CMC protocol defines two types of PKI requests and responses which are formed using PKCS#10 or CRMF structures [99]:

1. Simple PKI Request: This request if formed using the bare PKCS#10 structure.

2. Full PKI Request: This request consists of more than one PKCS#10 or CRMF messages structures encapsulated in a CMS as part of *PKIData*.

The two types of PKI Responses are based on *SignedData* or *AuthenticatedData* message structures :

1. Simple PKI Response: This response is a *certs-only SignedData*.

2. Full PKI Response: This response consists of content type *PKIResponse* wrapped in a *SignedData* message structure.

In CMC, RAs participate in the protocol by wrapping the end–entity PKI requests in another layer of PKI request and forwarding the expanded request to the CA server. Moreover, the CAs and RAs require the client to include

a proof–of–identity with the certification request. In CMC, the proof–of–identity is based on a shared secret between the client and CA/RA server. The shared–secret is a series of tokens which can be generated separately through a dedicated hardware device. The end–entity usually proves its identity by transferring this token in plain text along with a string identifier. Moreover, CMC also provides renewal and re–keying mechanisms. These requests are similar to any certification request wherein the end–entity needs to provide an identity proof such as existing certificates from another trusted CA [26].

Both CMC and CMP provide similar basic functionality for management of certificate life–cycle for PKI entities. CMP extensively reuses the existing CMS code which enables faster and efficient implementation of the protocol. CMP offers some additional functionality over CMP such as direct transfer of trusted root CA certificates to the end-entity. In terms of maturity level, deployment and interoperability CMP scores over CMC. Currently, CMP is supported in most of the PKI/CA products, offers extensive industry–wide interoperability and has been tested thoroughly in multi–vendor PKI environments. Due to the lack of implementations, CMC has not gone through extensive industry–wide testing and hence, will take more years to reach the stability and interoperability status offered by CMP [15].

3.4 EST

EST is a new protocol developed by Cisco to facilitate the life–cycle management of digital certificates. EST uses PKCS#10 and CMS for generating certificate requests and definitions. Moreover, the implementation is available as an open–source library developed by Cisco as *libEST* [21]. EST utilizes Cisco's Next Generation Encryption (NGE) by using Elliptic Curve Cryptography (ECC) instead of RSA encryption supported in SCEP. NGE provides high level of security and scalability through a set of algorithmic suites for authenticated encryption, elliptic-curve based digital signatures and key establishment, and cryptographic hashing. Both EST and SCEP facilitate certificate provisioning with the aim to provide certificates to the end points from a CA/RA server. However, these protocols do not aim at solving certificate management issues. Certificate management is handled separately through CMC and CMP [22].

Figure 3.4 depicts the protocol stack used by EST protocol for message transfer.

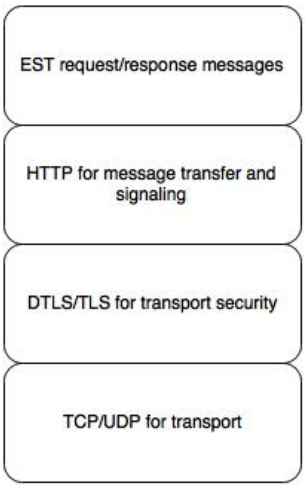

Figure 3.4: Protocol layers used with EST.

3.4.1 Operational Scenarios in EST

Prior to initiation of requests/response messages, the EST clients and server are configured with information to support mutual authentication and authorization. This information can be in the form of shared secrets, network service names, trust anchor information, enrollment keys, etc,. Moreover, based on the network management practices in an enterprise the client's might be pre–installed with trust anchors through an out–of–band secured procedure.

Figure 3.5 shows the general client–server interaction flow. Below we describe the operational scenarios with EST protocol in detail [90] :

1. Retrieving CA Certificates : Before performing any operation, the EST client requests a copy of EST CA certificates from the EST server. EST CA uses this certificate to sign objects that are being issued to the EST client such as certificates and CRLs. In order to authenticate and verify the authorization scope of the EST server the client can use different options including implicit trust anchor database, previously distributed trust anchor specific to the EST server, manual authentication performed by network administrator or a certificate–less TLS authentication based on a shared–secret specific to the EST server.

2. Initial enrollment of the client : After the client has authenticated an EST server, it can acquire a certificate by submitting an enrollment request to the EST server. Further, the EST server authenticates and

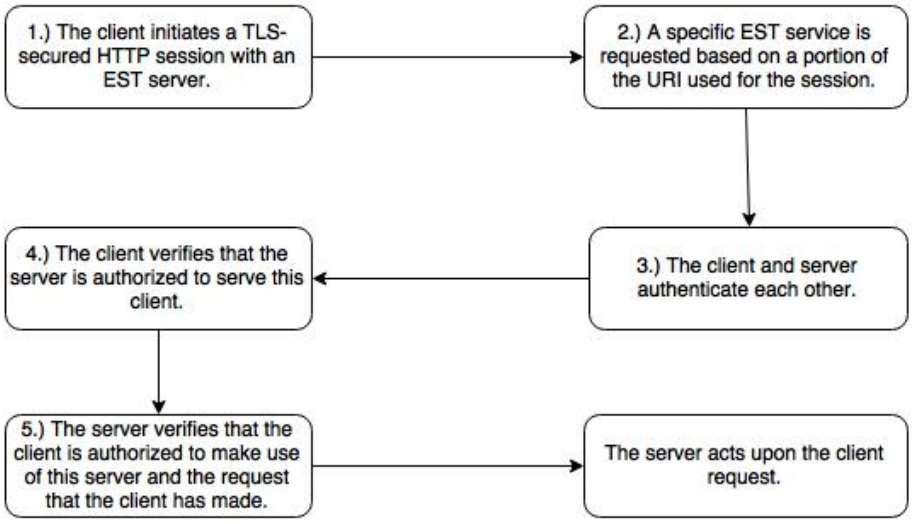

Figure 3.5: EST client and server interaction flow.

authorizes the EST client using different options such as TLS based authentication using vendor installed client certificate, certificate issued by other CAs or existing client certificate issued by the same EST server. Moreover the EST server can also perform certificate–less TLS authentication based on a shared–secret or HTTP–based authentication with username and password distributed out–of–band.

3. Certificate Signing Request (CSR) attribute request : An EST client can query the EST server for additional attributes before sending an enrollment request. These attributes are helpful in providing additional descriptive information to the EST server and the client. Information such as media access control (MAC) address of client interface, cryptographic hash function and encryption algorithm to be used can be obtained using a CSR request.

4. Re–issuance of client certificate : An EST client submits a re–enrollment request to renew its existing certificate. To authenticate itself the client can produce the existing certificate to the EST server or leverage any of the methods used for authentication during initial enrollment.

5. Server Key Generation : With the EST protocol the client can request the server to generate a key pair. The server is responsible for implementing appropriate random number and key generation as per [44]. Moreover, a CA policy determines the archiving of client key pair which

is sent to the client over a secure TLS session.

3.4.2 Comparison of EST with SCEP

EST is the recommended protocol for certificate provisioning over SCEP for the following reasons [22]:

1. SCEP messages are sent over HTTP using *pkiMessage*. To secure the *pkiMessage*, the messages are are enveloped in *pkcsPKIEnvelope* envelopes. However, EST employs Transport Layer Security (TLS) for secure transport of certificates and messages. This way there is no need to envelop the messages.

2. In case of SCEP, the certificate signing request (CSR) is authenticated through a shared secret between the client and the CA. Since EST uses TLS, the requester of a CSR is pre–authenticated and trusted, hence, the CSR is always tied to the requester. The method employed in SCEP introduces a security concern.

3. Within the PKI ecosystem, it is necessary to support automated certificate renewal and re-enrollment. The previous implementations of SCEP did not deploy such messages, however, recently submitted drafts of SCEP have added the support for automatic renewal and re–enrollment of certificates. In case of EST, automated certificate renewal and re-enrollment messages are built–in with the standard.

4. EST has also added server–side key generation through an enrollment request by the requester. In case of SCEP only private key generation at the client side is supported. Within resource constrained environments or devices server–side key generation can be an important aspect as they do not have enough entropy source and power for generation of a random private key.

5. As a standard SCEP was formulated in 1990s and did not receive wide community scrutiny during the development phase in IETF. EST was developed as a standard with the joint efforts of vendors and standards community and has received wide community scrutiny throughout the development phase. Moreover, Cisco has provided an open–source implementation of EST standard which is useful for vendors and researchers to experiment with.

6. EST supports CA rollover for refreshing trust anchors installed on the EST clients. To accomplish CA rollover, EST uses three certificates

during the transition period. Hence, all the PKI entities can be rolled to the new CA without affecting communication between the PKI entities involved. However, CA rollover with SCEP is less flexible and lacks automation. SCEP does not have a transition period for CA rollover instead it requires a *flag day*.

EST provides significant security improvements over SCEP in the following ways :

1. With EST the certificate signing request is tied to a requester which is authenticated with TLS. Hence, the certificate is rendered only to the requesting entity. The requesting entity holds a private key or a user–name and password. The user–name and password acts as a proof of possession (PoP). By enforcing PoP, requesting entities or clients can not forge a certificate for other clients. However, with SCEP a CSR is authenticated using a shared secret between a client and CA. This introduces a security vulnerability in real–world deployments. Any client with an access to the shared secret can generate a CSR for other clients. Moreover, the shared secret is not a onetime secret for each client, hence, it further complicates the distribution of a shared secret.

2. Over the years, TLS protocol has been continuously improved and its security has been proven. This ensures that EST messages are secured. However, SCEP is tightly coupled with RSA to provide data protection which introduces security concerns with technological advancements.

3. EST offers better cryptographic agility than SCEP by supporting ECC as a secure cryptographic algorithm for encryption. ECC is computationally efficient and better suited to the needs of resource–constrained environments. SCEP uses PKCS#7 methods based on RSA encryption.

3.4.3 Comparison of EST with CMC and CMP

CMC and CMP protocol support to certificate management including certificate enrollment, certificate status, certificate revocation, etc,. EST caters to certificate provisioning, hence, it can be considered as a profile of CMC that uses a secure mode of transport for key enrollment and renewal. In case of CA certificate rollover, EST follows CMP standard. In brief, CMC and CMP were defined as two standards with common goal within a short time frame by IETF. However, both the standards have to failed to gather mainstream acceptance as they are complicated to implement.

Below we briefly describe ways in which EST differs from CMC and CMP:

1. To define the ways of processing control data CMC uses multiple wrappings of CMS messages and Abstract Syntax Notation (ASN.1) structures. CMP incorporates enveloped message data for control data processing. However, the messages in EST are simple and lightweight.

2. EST incorporates TLS for secure transmission of messages.

3. With an enrollment request a server–side key can be generated in EST standard. Moreover, the private key for client can be securely transferred with TLS without the need of further encryption. This can be a vital step in case of Resource Public Key Infrastructure (RPKI) or low power devices which do not have enough entropy for generating a random private key. Server–side key generation is not included in CMP standard. Moreover, CMC does not address server–side key generation operation.

4. EST supports the renewal of CA certificates by combining renewal messages defined in CMP with the CMC specification. However, CMC does not support the renewal of CA certificates.

3.4.4 Adoption of EST

SCEP has been supported as an industry wide standard in vendor products for more than 15 years. It is included in many standards and all the CAs support it. However, the limitations of SCEP are distinguishable with resource constrained devices as described in the Section 3.4.2. CMP has been made mandatory by the 3rd Generation Partnership Project (3GPP) as part of TS 33.310 standard. However, as a part of TS 33.310 standard the utilization of CMP is limited to provisioning. Moreover, CMP is also included in the National Institute of Standards and Technology's (NIST) Special Publication 800-57. Further, CMP and CMC are also supported by some of the CA vendors and PKI products. The latest EST standard is used in IETF ANIMA WG's bootstrapping drafts. Moreover, Wi–Fi alliance has mandated the use of EST in hot–spot 2.0. Additionally, to address security concerns within power systems the International Electrotechnical Commission (IEC) created IEC 62351. IEC also mandates the use of EST as a certificate provisioning protocol. Currently, multiple CA vendors are adding support for EST such as Cisco's IOS and IOS-XE product [22].

Amongst the available protocols for certificate provisioning EST stands out for its simplicity, open–source development and advantages it provides over the other protocols. The open–source implementation in the form of a portable library is easy to use and promotes quicker adoption and increases

interoperability of EST into more vendor products. As per Industry Trade Association for Public Services Networks (PSN), "Cisco's release of its EST code into the open source community will facilitate rapid adoption by the PSN community. With the release of this code, other vendors will be able to accelerate their adoption of EST and this in turn expands the choice of encryption solutions available to public sector organizations" [16].

3.4.5 Potential Use Cases of EST

EST can be effectively deployed in a variety of use–case scenarios. One such scenario can be an enterprise with numerous network end–points which require periodic certificate renewal. In case of enterprise server's certificate expiration, EST offers automatic re–enrollment for obtaining new certificate. Subsequently, speeding up the entire procedure and this requires no manual intervention from network administrator. Moreover, EST also supports automatic redistribution of updated CA certificates. The future IoT environment is bound to support large number of end–points giving rise to highly complex certificate management issues. EST improvements can prove valuable and important in such IoT deployment scenarios. We will discuss in detail the deployment scenarios in Chapter 4.

TLS protocol has seen a lot of attacks including server–side certificate forging leading to server impersonation and discovery of bugs in the protocol such as SLOTH [32], Heartbleed [43, 75], BREACH [49], etc,. Such attacks set the enterprises, consumers and organization into panic to determine an immediate and appropriate solution which generally requires replacement of server certificates. Re–enrollment capabilities of EST will support rapid resolution of such security vulnerabilities.

Chapter 4

Security in IoT devices

IoT refers to the currently growing network of smart devices and sensors with an IP address for Internet connectivity, and the communication occurring between these smart devices and other Internet–capable devices and systems. IoT devices share information and perform actions based on user input or an automated controlling system to interact with other Internet–capable devices and systems. This interconnected system between IoT and other Internet–capable system promises improved quality of life, increased efficiency and greater economic growth. Moreover, IoT offers advanced connectivity between devices, systems and applications and introduces new protocols, specifications and applications [24].

4.1 Internet of Things market potential

Gartner's recent reports on IoT devices indicate that there are more than 3 billion IoT devices available in the smart environment at present and there is a continuing interest in the market towards adoption of IoT devices, with deployments planned on a global scale [23]. Currently, the use of IoT devices is widespread in manufacturing sectors seeking for automation, operational efficiency and resource optimization benefits [19]. However, there is a boom in the number of small–scale innovative vendors focusing on categories such as smart–connected homes, smart machines, semiconductors and IoT device security [17]. Core issues such as choosing the relevant platform and database, applying appropriate cloud services and analytics, and managing the security of IoT devices, need to be addressed to help foster the adoption of IoT devices in future [20].

Figure 4.1 demonstrates a simplified IoT network architecture. IoT applications runs as a web application on a dedicated cloud server. The server

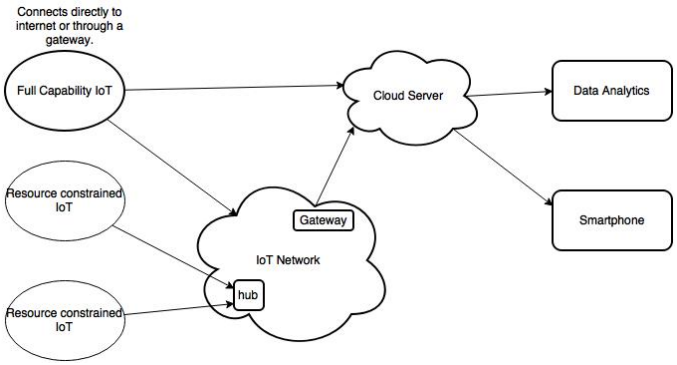

Figure 4.1: Simplified IoT system architecture.

collects device data which is fed to data analytics module to draw conclusion suited to the requirements of enterprise deploying the architecture.

4.1.1 Deployment Scenarios with IoT

A number of different markets are integrating IoT devices into their infrastructure and processes. IoT devices are fostering automation in fields including automotive, health–care, smart–cities, transportation and industries catering to daily utilities for reducing cost and increasing energy efficiency. Commercially available IoT devices such as smart meters, solar panels, etc, cater to aforesaid requirements. Moreover, IoT devices such as heart–monitoring devices, wireless insulin pumps, sensors for automobiles, smart–home appliances are also enabling development of advanced applications for daily use.

4.1.1.1 Automotive

Connected vehicle technology is an ecosystem of smart–cars, trucks or buses embedded with IoT devices or internal sensors to determine accurate speed, location and temperature of the vehicle. Moreover, this ecosystem is aimed at enabling interaction between smart–vehicles with surrounding roads, buildings, pedestrians and other smart–vehicles in order to improve road safety, reduce maintenance and insurance costs and avoid traffic congestion scenarios.

Figure 4.2 demonstrates the architecture for IoT applications in connected vehicular technology deployment scenario.

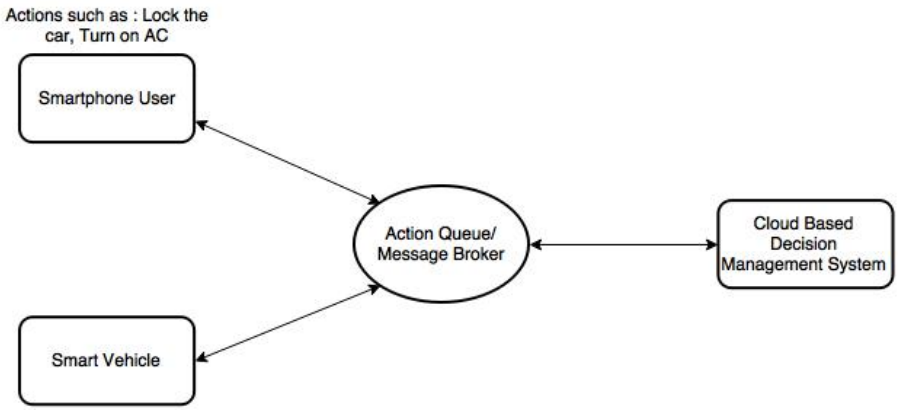

Figure 4.2: Automotive deployment scenario with Smart vehicular technology.

4.1.1.2 Health–care

IoT holds an immense potential to transform health–care into *Internet of Health–care Things*. Connected IoT devices can be used to continuously monitor patients with chronic conditions such as diabetes, heart–risks, etc,. to tackle emergency situations, avoid hospitalization and handle pre–hospitalization patient report generation to improve quality of life and reduce cost of care for patients with chronic diseases. Moreover, personalised IoT devices can minimize patient–doctor interaction by continuously recording, reporting and triggering emergency alarms generated over health data based on daily activities.

Figure 4.3 demonstrates the architecture for IoT applications in health–care.

4.1.1.3 Smart–cities

Smart city is a vision of interconnecting public spaces and infrastructure in urban cities through embedded sensors such as IoT devices. Smart city caters to market segments such energy, transportation, health–care, buildings, infrastructure and governance. This concept has driven innovation and experimentation with IoT technology into adaptive traffic monitoring and controlling, self-driven cars, smart–home meters, monitoring environmental changes, resource management within a city, smart–home controllers and security systems, energy management and security solutions in office infrastructure, improving workplace productivity, etc,. Some quick use cases where

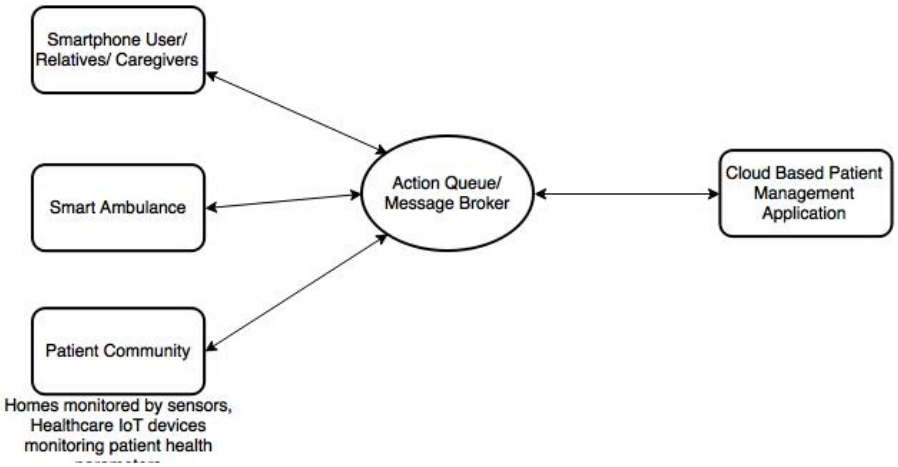

Figure 4.3: Deployment scenario with Internet of Health–care Things.

the aforementioned technology helps include adjustment in travel schedules of commuters based on real–time tracking of public transportation, improving air and water quality based on data provided through environmental monitoring, reducing electricity loss during distribution with smart–meters, etc, [78].

Figure 4.4 demonstrates the architecture for deployment of IoT applications in a smart–city.

4.1.1.4 Intelligent Transportation Systems

Intelligent transportation systems are fostering the adoption of IoT devices and embedded sensors in vehicles such as cars, trucks, ships, air–crafts and passenger trains. Moreover, IoT devices are permeating into transport infrastructure such as roadways, tunnels, bridges and railway tracks. This widespread deployment will lead to vehicle, passenger and pedestrian safety, increasing fuel efficiency, monitoring environmental pollution, routing of logistics transport, efficient selection of parking space and reducing road congestion [28].

Figure 4.5 demonstrates the architecture for deployment of IoT applications in intelligent transportation systems.

4.1.1.5 IoT utilities

IoT devices are increasingly becoming an important part of the utility industry with focus on improving energy efficiency, conserving energy and water

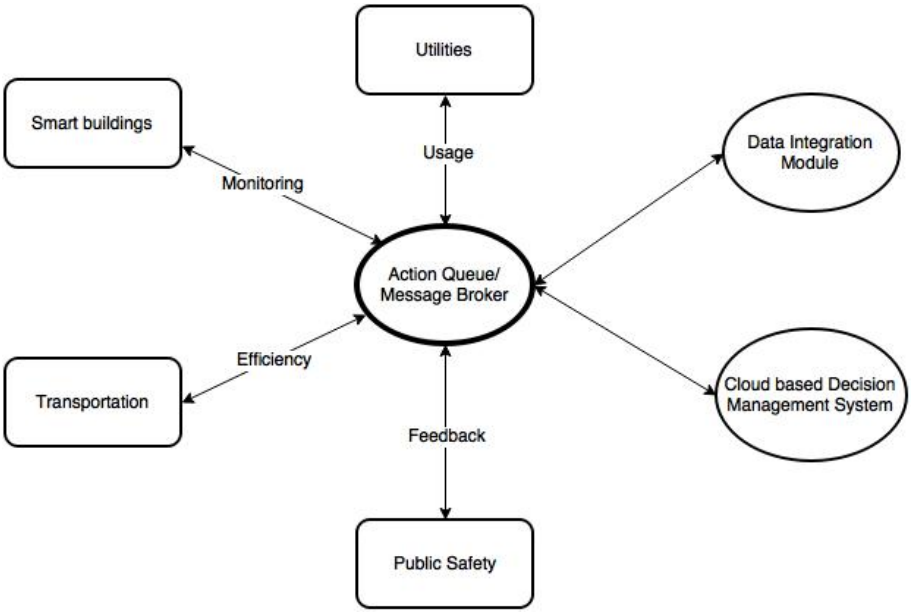

Figure 4.4: IoT deployment scenario in Smart–cities.

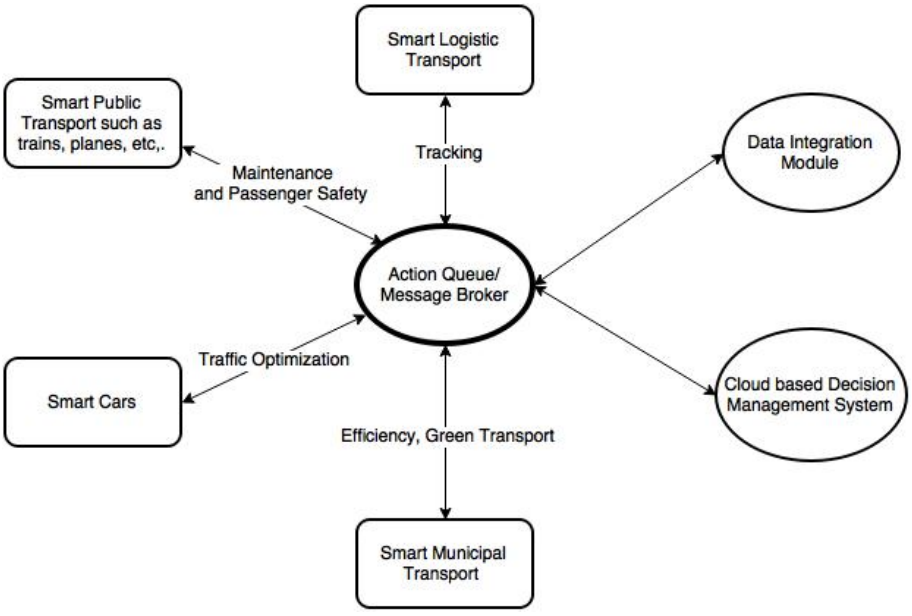

Figure 4.5: IoT deployment scenario in transportation.

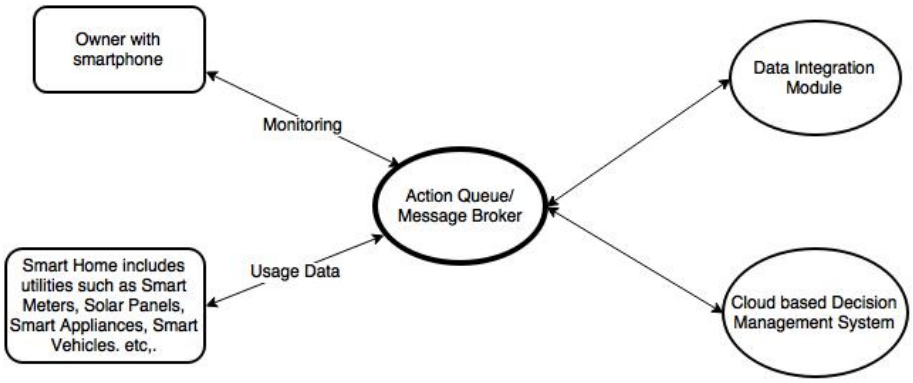

Figure 4.6: IoT deployment scenario in Smart–Home utilities.

resources, improving home safety and security, automation of daily housework and reducing carbon emissions. This has led to adoption of devices such as smart–grids, smart–meters, smart–home appliances to enhance interaction between product, product owners and product manufacturers [29].

Figure 4.6 demonstrates the architecture for deployment of IoT applications in Smart–Home utilities.

4.1.2 Impact of various IoT applications

McKinsey estimates that IoT has a potential economic impact of 3.9 trillion to 11.1 trillion dollars per year by 2025. These estimates are proposed considering factors such as IoT adoption rates, economic growth and demographic trends [78].

As shown in Figure A.6, IoT applications catering to manufacturing factories will have the greatest impact equalling to approximate $3.7 trillion per year. Moreover, smart–cities are the next largest IoT application deployment scenario with a potential impact of $1.7 trillion per year by 2025. Figure A.6 also demonstrates major applications that will be deployed for the specific IoT settings.

4.2 Security Attacks in IoT ecosystem

The rise of connected devices in IoT ecosystem has created a pressing demand for robust security mechanisms [56, 76]. The number of attacks and the tools available to potential attackers is becoming more efficient and effective as the IoT ecosystem grows [65, 100]. Moreover, Cyber threats are prevalent

against IoT deployment scenarios leading to disabling of system operation, economic loss and privacy threat to sensitive user data [33, 95]. Such security issues can hamper the deployment of IoT services and applications [48]. Moreover, hackers and attackers are showing increasing interest in exploiting the security vulnerabilities in this area [45, 107]. Hence, security and privacy are a serious concern for both device owners and manufacturers in the shift towards IoT [25, 77].

Below we list some such security attacks on various IoT deployment scenarios :

1. In 2012, a series of hacks were perpetrated against smart–meter installations in Puerto Rico costing hundreds of millions of dollars annually. The law enforcement agency expected similar threats to spread across the country as more utilities deploy smart grid technology [87].

2. In 2012, Justin W. Clarke, a security researcher from San Francisco Bay area, publicly disclosed a flaw in hardened grid and router provider RuggedCom's switches. By accessing the private key stored in Rugged operating system, the attacker can successfully decrypt all the traffic sent to or from the switch. This can further lead to compromise of the entire energy grid [102].

3. Within a positive train control (PTC) system, the level of automation and control also exposes the system to dangerous vulnerabilities. In case, a malicious user gains system–level access, it is free to execute any set of commands to trigger unwanted automated chain of actions [18]. One such incident was reported in Poland, wherein a 14–year old hacker changed the points on the city's tram system leading to derailment and injuring 12 passengers [35].

4. In 2014, Scott Erven and his team of security researchers released their study on the vulnerability of medical devices. The study reveals that drug infusion pumps can be remotely manipulated, X–rays can be accessed by attackers on hospital's network, attackers can take down critical equipment during emergency situations and reset the configuration of testing equipment [109].

5. In the recent past, cyber attacks have been launched against public utility infrastructure such as power systems and water treatment plants to affect the distribution of water and electricity [66].

6. In early 2015, a German steel mill was attacked by hackers leading to massive physical damage. The hackers managed to manipulate and

disrupt the control systems such that the blast furnace could not shut down properly [110].

7. In 2015, two well known hackers Charlie Miller and Chris Valasek remotely compromised Jeep Cherokee and gained complete physical control over the vehicle. Prior to this, their research on vulnerabilities in connected cars lead them to hacking Toyota Prius and a Ford Escape by plugging a laptop into the vehicle's diagnostic port allowing them to control headlights, steering and brakes [80].

8. In 2015, Samsung shipped a smart–fridge with a significant security flaw that can allow an attacker to gain access to login credential of the user. The application level implementation does not validate SSL server's certificates [112].

9. In 2015, Marc Rogers and Kevin Mahaffey, demonstrated vulnerabilities found in Tesla Model S that can allow an attacker to start the vehicle by connecting a laptop to the car's network cable. Moreover, the security researchers could also plant a remote-access trojan on the car's internal network which allowed them to remotely control the car [111].

10. In 2015, several vulnerabilities were disclosed in *Hello Barbie*, an iconic toy which enables children to communicate with the doll over a cloud server connection. The vulnerability allows an attacker to intercept and spoof the audio communication between the toy and children [88].

4.3 PKI as a security solution for IoT

The rise in the number of connected IoT devices has lead to increase in the challenges to secure inter–device communication. The IoT ecosystem requires proven, reliable and tested network security solutions for data confidentiality, data integrity, device authentication and device authorization. However, recent attacks on various IoT deployment scenarios as discussed in Section 4.2 demonstrate a lack of generic security architecture or standard in deployment and operation of IoT devices [73, 113].

From IoT ecosystem perspective, communications occur between the IoT devices or between IoT devices and cloud server. Data confidentiality and integrity are concerned with securing this data exchange against interception, interruption and modification. Moreover, authentication and authorization provide assurance that an IoT device is an entity it claims to be and the

IoT device is authorized to perform a particular action. As a security solution PKI provides and manages digital certificates which bind a public key to an IoT device identity or private key such that any other entity in the environment can validate this binding [101]. Currently, digital certificates used for identity management are based on X.509 certificates which are self–descriptive entities specified using ASN.1. X.509 certificates are encoded using DER standards and are stored as an ASCII string using BASE64 encoding [36].

4.3.1 Security protocols in IoT protocol stack

With TCP/IP protocol stack, secure Internet communication is provided through 3 key components :

1. Network protocols such as IPsec and TLS provide authentication and data confidentiality at network and application layer for peer–to–peer communication [40, 62].

2. Digital certificates provide a valid identity to each peer in the network.

3. PKI infrastructure supports the provisioning, management and revocation of the digital certificates.

This three component based security solution can further be applied to IoT networks. As shown in Figure 4.7 IoT protocol stack at network layer is based on IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) [53]. Hence, network layer secure communication can be provided through IPsec. Moreover, IoT uses Datagram Transport Layer Security (DTLS) to secure application layer data. DTLS protocol is based on TLS and provides equivalent security to secure peer–to–peer communications [93]. In addition, IoT protocol stack has been optimized for low power devices and hence the data overhead generated through the stack is much smaller in comparison to the TCP/IP protocol stack [63].

4.3.2 PKI consideration for IoT

PKI is designed to provide secure communication between Internet–based client–server architecture. There are various challenges to be considered prior to adopting PKI to provide secure communication in IoT deployment scenario :

1. Peer authentication in IoT : In IP networks device authentication and secure communication for application layer data is done with TLS protocol. TLS uses X.509 certificates during initial authentication phase

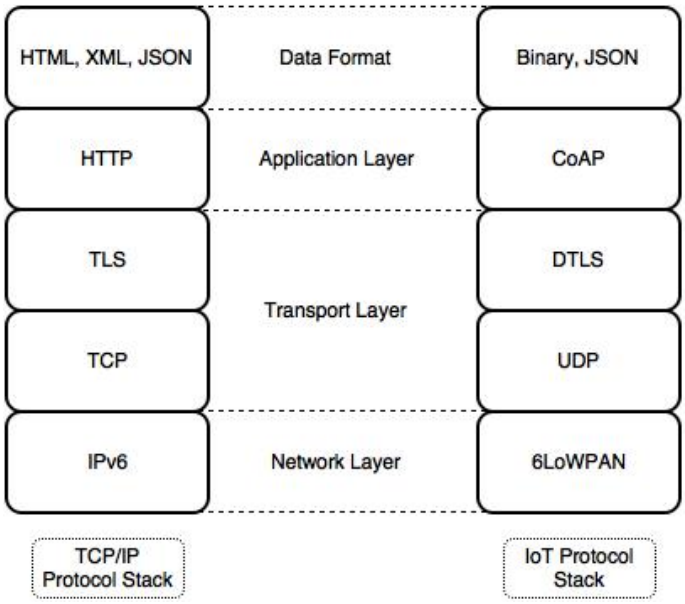

Figure 4.7: Comparison of IoT protocol stack with TCP/IP protocol stack.

between two peers to negotiate the session key. Moreover, only server presents a X.509 certificate to authenticate its identity with the client. It is optional for the client to present X.509 certificate during the initial authenticate phase. In case of IoT deployment scenario, it is necessary to configure TLS to provide client's X.509 certificate for mutual authentication between communicating IoT devices or between IoT device and the cloud server.

2. Key Generation : IoT devices are resource constrained devices and may not have enough entropy to generate random private key. Therefore, it is preferable to generate the private key at the CA/RA server. EST certificate management protocol supports generation of client keys at CA server and further secure transportation of the key over TLS and DTLS. Moreover, the PKI solution should consider whether the key should be completely removed to maintain the integrity of generated certificates or securely store them through RA/CA server for revocation or renewal cases.

3. Choice of Cryptographic Algorithm: As shown in Figure 4.7, DTLS is the proposed protocol for securing device authentication and communication for application layer data. Using asymmetric encryption based on RSA and PKI certificates, imposes high energy consumption in the

IoT deployment scenario. In consequence, ECC is a better suited approach for IoT deployment scenario in comparison to RSA as it is more energy efficient and offers same level of security with lesser key size [50]. Moreover, EST offers CA side support for generation of private keys based on ECC algorithm. Hence, EST can be used as the protocol for certificate management in PKI security solution for IoT deployment scenario.

4. Scalable Database Solution : PKI security solution should be effective in scaling for different IoT deployment scenarios. IoT devices can be deployed within a home or a building or over an entire city. Hence, there is a need for an efficient and scalable database management system for storing certificates that cater to different scale of IoT devices ranging from thousands to millions to billions.

5. Life–Span of Digital Certificates : The lifetime of a digital certificate may vary based on the type of IoT device, device manufacturer and IoT deployment scenario. Some situations may warrant that certificates and private keys can never be updated or replaced because they are permanently stored on the device. However, some other scenarios for IoT deployment may demand storing short–lived certificate on the IoT device. In such situation handling revocation or renewal at a high scale has to be considered in the design of the PKI solution.

6. Certificate Management Protocol : Currently, various certificate management protocol such as SCEP, CMC and EST, exist for supporting secure certificate provisioning. It is necessary to consider advantages and disadvantages of each before choosing the suitable one for PKI solution.

7. Managing High–Volume Issuance : Irrespective of deployment scenario, every IoT device will be installed with a vendor certificate during manufacturing. This requirement necessitates a highly available, fault-tolerant and scalable CA server.

8. Number of certificates per device : Depending on the deployment scenario the number certificates installed on the device may vary. For example, a smart vehicle will not only need a pre–installed vendor certificate but an operator certificate and other necessary certificates in order to communicate with entities such as other smart–vehicles, transportation infrastructure (traffic lights), etc., while on the move.

Chapter 5

Naming of IoT Devices

To support large scale deployment of IoT devices, naming and identification of the device is a pre–requisite. However, it is impossible to assign names to IoT devices from a single name–space considering the number of devices deployed will reach to billions. Naming of the IoT devices can be either hierarchical or flat. Hierarchical names cater to scalable name assignment and resolution while flat names, generated based on unique identifier such as MAC address, work as a self–certified IDs [57].

5.1 DNS name auto–configuration for IoT devices

Currently, network devices support automatic configuration of IPv6 address using neighbour discovery [84], default gateway configuration by employing IPv6 stateless address auto–configuration [105] and DNS server configuration with IPv6 router advertisement (RA) options for DNS configuration [54]. However, DNS names have to be configured manually and will prove to be cumbersome in IoT deployment scenario considering the growing scale of deployed devices. For our scalable and flexible PKI solution we will consider the auto–configuration scheme of DNS names [55]. The auto–configuration scheme generates a DNS name at the host, assisting easy monitoring and remote controlling of IoT devices [72]. This scheme works in unicast mode as opposed to multicast mode used in bonjour application [1] and hence, generates less network traffic. To generate the DNS name, the auto–

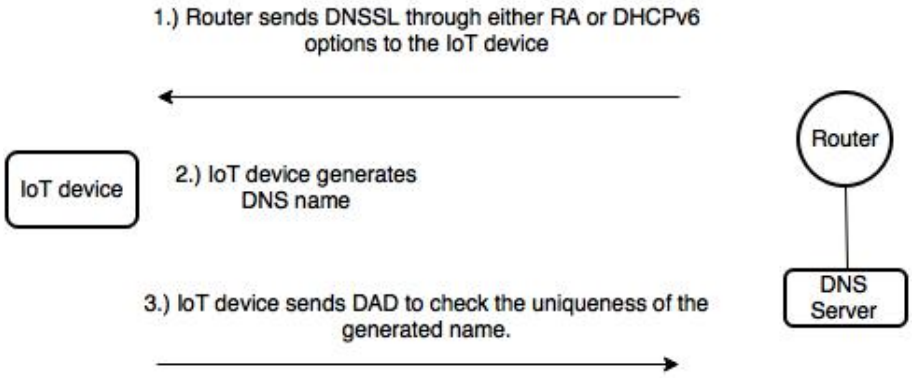

Figure 5.1: Message flow to demonstrate auto–generation of DNS name.

configuration mechanism needs to acquire the DNS search list (DNSSL) either through RA options [54] or via dynamic host configuration protocol version 6 (DHCPv6) options [42]. DNSSL is a list of DNS suffix domain names that are used by IPv6 hosts.

The auto–generated DNS name is of the format *unique_id.device_model.device_category.location.domain_name*. *unique_id* is a unique identifier in ASCII characters. It can be a sequence of numbers or set of alphanumeric characters. *device_model* is the model name for the device in ASCII characters provided by manufacturer. *device_category* is the category to which the device belongs such as refrigerator, smart–tv, smart–meter, etc,. *location* represents the physical location of the device. *domain_name* is the DNS domain name such as example.com or aalto.com [55].

The procedure for DNS name auto–configuration is completed in two phases. In the first phase, the IoT device generates it own DNS name and in the second phase the generated name is registered with the DNS server [72].

Figure 5.1 shows the first phase in DNS name auto–configuration. In this phase, the IoT device gets the DNSSL list through either RA or DHCPv6 options. It performs validity check on the DNSSL option received. If it is valid, then the IPv6 host or IoT device generates the unique DNS name after performing duplicate address detection (DAD).

Figure 5.2 shows the second phase in DNS name auto–configuration. In this phase, an IoT device receives a node information (NI) [37] query from the router in the same sub–net. It sends the NI response with the

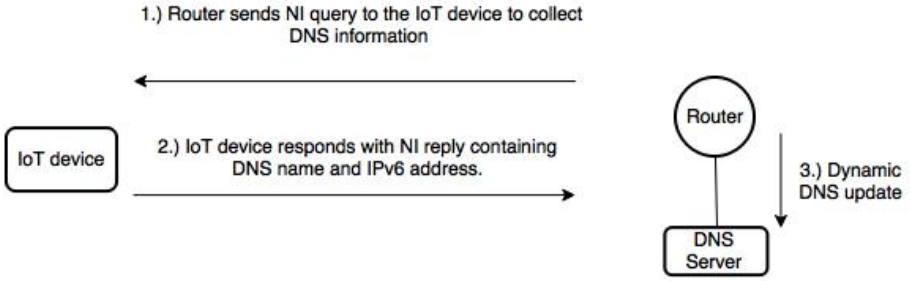

Figure 5.2: Message flow to demonstrate DNS name registration.

generated DNS name. The router then registers the DNS information with the DNS server through dynamic DNS update [108].

Chapter 6

Certificate Management with Merkle Hash Trees

Certificate management system maintained by the CA server supports insertion, deletion and searching of digital certificates. Our proposed PKI solution uses merkle hash tree data structure based on authenticated dictionary called as AD–MHT [81, 83] for supporting the aforementioned operations. Identity certificates are used to bind a public key with an identity. It is an association between a distinguished name (DN) and the requesting entity's public key. It is necessary that each requesting entity posses a unique DN. Moreover, X.509 standard defines the data and data format for identity certificates. The data fields in X.509 certificates include version, serial number, signature algorithm identifier, issuer name, validity period, subject name and subject public–key information. Serial number is generated by the CA and is used to uniquely identify a certificate. Issuer name is the DN of the issuing entity. Subject name is the DN of the entity requesting a certificate. Moreover, validity period defines the expiration time of the certificate. The certificate has to be revoked once it is expired.

PKI handles the management of certificates during the entire lifetime. With certificate revocation an issuer can revoke the binding between the identity and the public key before the certificate expires. Certificate revocation is needed in case of loss or compromise of private keys, change of access rights of the certificate owner, policy changes at CA or as a precautionary measure against certificate forging attacks [47].

Figure 6.1 demonstrates the certificate revocation mechanism. In order to revoke a certificate the issuer initiates the revocation process and

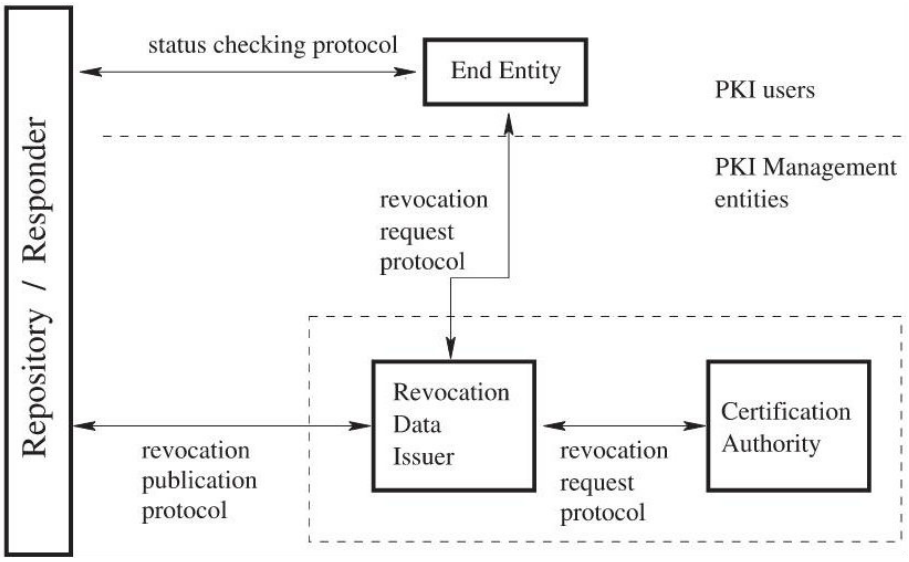

Figure 6.1: CA reference model for certificate revocation [81].

sends a revocation request to the revocation data issuer (RDI). RDI has the access to the database of the revoked certificates and is also responsible for maintaining an appropriate status data format for the revoked certificates that can be distributed to the end–entities [81]. CMP is one of the protocol that can be used for communication in revocation reference model. Moreover, the end–entities can use CRL for status checking. CRL is digitally signed by the CA and includes information such as certificate serial number, revocation reason and revocation date [36].

6.1 Merkle Hash Tree

MHT stores data only in the leaves of the tree. The contents of the leaf nodes are hashed and combined to generate nodes for the upper level of the tree. This process is recursively applied to generate the root node of the tree which is digitally signed by the CA to authenticate the entire MHT [81].

Figure 6.2 shows a sample MHT. MHT employs properties of one way hash functions (OWHF). OWHFs are 10,000 times faster to compute than digital signatures. $N_{i,j}$ denotes a node in MHT where i represents

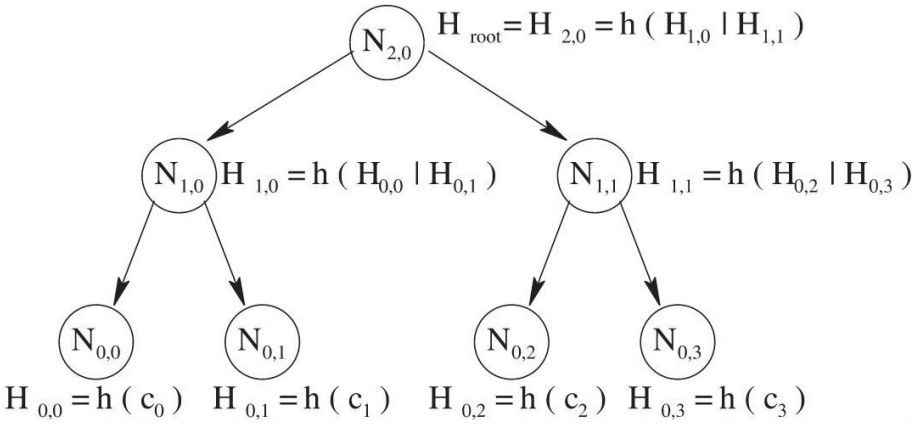

Figure 6.2: An example of MHT [81].

the depth or the level of the node and j represents the number of the node. Moreover, $H_{i,j}$ is the hash value stored by the node $N_{i,j}$. The nodes at level 0 are called leaf nodes and they store the data in the tree. The data stored is represented by the set ϕ[79].

$$\phi = \{c_0, c_1, c_2.....c_n\}$$

where c_j is the data stored by leaf node $N_{0,j}$. Hence, $H_{0,j}$ can be computed as :

$$H_{0,j} = h(c_j)$$

where h is the OWHF. In order to construct a MHT, a set of t adjacent nodes at level i; $N_{i,j}$, $N_{i,j+1}$, ..., $N_{i,j+t-1}$ are concatenated to create the node for upper level which is denoted by $N_{i+1,k}$. Hence, $H_{i+1,k}$ is obtained by applying h or OWHF to the concatenation which is denoted by the equation as :

$$H_{i+1,k} = h(H_{i,j}|H_{i,j+1}|...|H_{i,j+t-1})$$

The root node at the top level is a digest for all the data stored in the MHT. MHT shown in Figure 6.2 is a binary tree with the adjacent nodes combining in pairs to form the node in the next level. MHT–AD is a 2–3 tree structure where each internal node has two or three children. For a MHT–AD with n leaves, certificate management operations such as insertion, deletion and search can be performed in O(log(n)) time. Each leaf in the MHT-AD

represents a certificate. The leaves are ordered by the serial number of the certificate [79].

6.2 Certificate Transparency

Over the past years there have been numerous incidents where an attacker was able to compromise the private keys of intermediate CA or the root CA. This lead to issuance of unauthorised SSL certificates for domains owned by Google, Facebook, Yahoo and Microsoft. This way the attacker could impersonate a web server or act as a man-in-the-middle to get critical user information like user–name, password, bank details, download malware on victim's machine etc. In some cases, attacks are being carried out by enterprises to monitor employees or by government agencies for surveillance purposes. Below, we list some of the detected forged SSL certificate attacks :

(a) In March, 2011 Comodo Inc. experienced a major security compromise and issued 9 fraudulent digital certificates for domains such as Google, Yahoo and Skype [94]. Attacker was successful in obtaining user–name and password of Comodo's RA in Southern Europe and issue the forged certificates. RA are subordinate to CA and perform the task of authenticating the identities of certificate requesting entities. The attacker falsely attested the authenticity of the parties requesting the certificate using the stolen RA log–in information. The attacker's IP address was traced back to Iran. It was purported that the attack was carried out for political motives.

(b) In 2011, an attacker successfully took control of 8 certificate–issuing servers of Dutch CA DigiNotar [46]. The attacker was successful in obtaining wild card certificates for *.google.com and other domains such as Yahoo and Mozilla. It was purported that the attack was aimed at launching a large–scale man–in–the–middle attack against Internet users in Iran. This attack led to bankruptcy of Dutch CA DigiNotar.

(c) In early December, 2013 Google reported an unauthorized digital certificate issued by an intermediate CA [69]. The forged digital certificate was linked back to the French CA ANSSI. The certificate was used to monitor traffic on a private network.

(d) In late December, 2013 Google again reported an unauthorized wild card digital certificate issued by an intermediate CA [68]. The forged digital certificate was linked back to TURKTRUST, a Turkish CA. It was revealed by TURKTRUST that way back in August 2011, they had issued 2 intermediate certificates to an organization mistakenly.

Certificate Transparency proposed by Google aims at solving the aforementioned certificate–based threats by making CA operations such as issuance, revocation, renewal, etc,. on SSL certificates open to scrutiny by domain owners, CAs and domain users through certificate logs, monitors and auditors. Certificate transparency intends to bring public scrutiny and openness to the current PKI model. It supports early detection and faster mitigation of forged certificates, rogue CAs and offers a better oversight of the current PKI model [3].

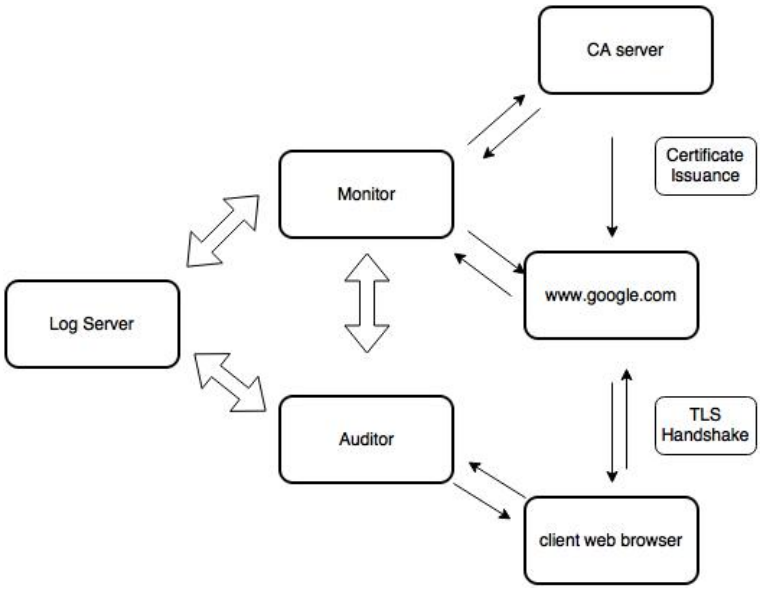

Figure 6.3: Certificate Transparency system architecture[70].

Certificate logs are at core of the architecture shown in Figure 6.3. Certificate logs are append–only and are assured using MHT proof–of–presence mechanism. By using the cryptographic hash the logs are tamper–proof, incorruptible and it makes possible to detect forged forks in the logs or outdated certificate insertion with the help of auditors. Moreover, each certificate log is publicly advertised with a URL and public key. End–points can use HTTPS

GET and POST messages to interact with the log server. When a CA submits valid certificate to the log, the log server responds back with a signed certificate timestamp (SCT). SCT is associated with a certificate throughout the lifetime and the TLS server attaches this value with the certificate during initial handshake phase [71].

Monitors are used to oversee the operation of the log server and watch the logs for forge certificates, unusual certificate extensions, certificates with false permissions, etc,. Monitors periodically fetch new entries in the log server to verify that they are publicly visible. Moreover, monitors maintain complete copy of the log server and can be used as a read–only back–up service when log server goes offline. Auditors verify the integrity of logs by periodically fetching and checking whether a particular certificate exists in the log.

Certificate transparency uses signed cryptographic hashes of MHT as log–proofs to facilitate public auditing of certificates and logs. The root hash of the MHT is signed by the log–server and is called signed tree hash (STH). Moreover, the log server periodically appends newly acquired certificates to the log and creates a new MHT with an updated STH. Through the STH, log server is able to provide consistency and audit proofs for any certificate. With the consistency proof, it is easy to verify that all certificates have been consistently appended to the log. Further, the audit proof helps in verifying that a particular certificate has been appended to the log [70].

A merkle consistency proof checks that two different versions of the logs are consistent with each other, that is, the later version is append–only consistent with the earlier version. In order to accomplish this, first, it is necessary to verify that the old merkle tree hash is a subset of the new hash value. Then, it is enough to verify that the new hash value is a concatenation of the old hash value along with all the intermediate node hashes of the newly inserted certificates. The proof is the set of minimum intermediate node hash values. A merkle audit proof verifies the proof–of–presence of a specific certificate in the log. In case, the proof–of–presence check for a certificate fails a TLS client can reject the connection. A merkle proof is the missing node hashes required to compute root hash with a given leaf hash value. A leaf or certificate exists in the log if the root hash computed with the audit path matches with the merkle tree hash value of the log [96].

Chapter 7

Bootstrapping IoT Device

This chapter describes our proposed IoT deployment model and the bootstrap mechanism to bring up an IoT device and provision it with a digital certificate.

7.1 IoT deployment model

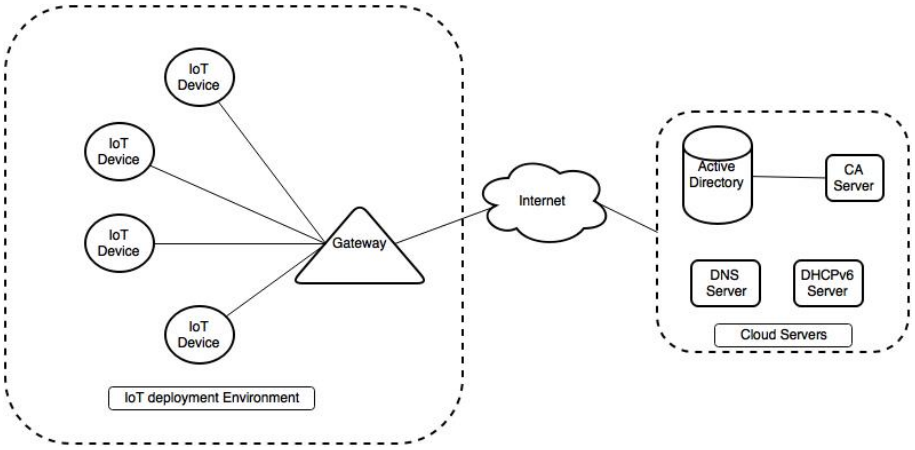

Figure 7.1: Proposed Deployment Model for IoT devices.

Figure 7.1 demonstrates the IoT deployment model that we propose. IoT deployment environment can be an enterprise, a manufacturing unit, transportation, health care services etc,. The devices are connected to an IoT gateway which connects the devices to the Internet. For simplicity of the

model, we assume that the CA server, DNS server and the DHCPv6 server are cloud–based.

7.2 Bootstrapping Procedure

In this section we will explain the proposed bootstrapping procedure for an IoT device. Figure 7.2 and Figure 7.3 demonstrate the bootstrapping procedure step-by-step.

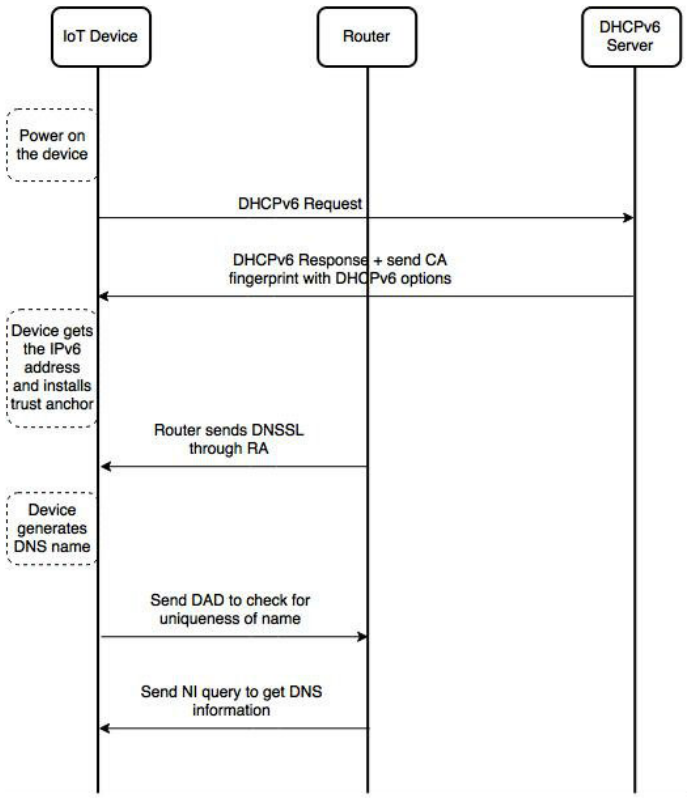

Figure 7.2: Message flow to demonstrate bootstrapping of IoT device.

The device sends a DHCPv6 request to the server after booting up. With the DHCPv6 response the device obtains an IPv6 address along with the implicit trust anchor for the CA server. After installing the trust anchor, the device can request the DNSSL from the router. Moreover, the router periodically advertises the DNSSL through RA. The device generates a unique auto–generated DNS name and sends a DAD to the router to confirm the

uniqueness of the name across the network. Thereafter, with NI query and response the router registers the device name through a dynamic DNS update with the DNS server.

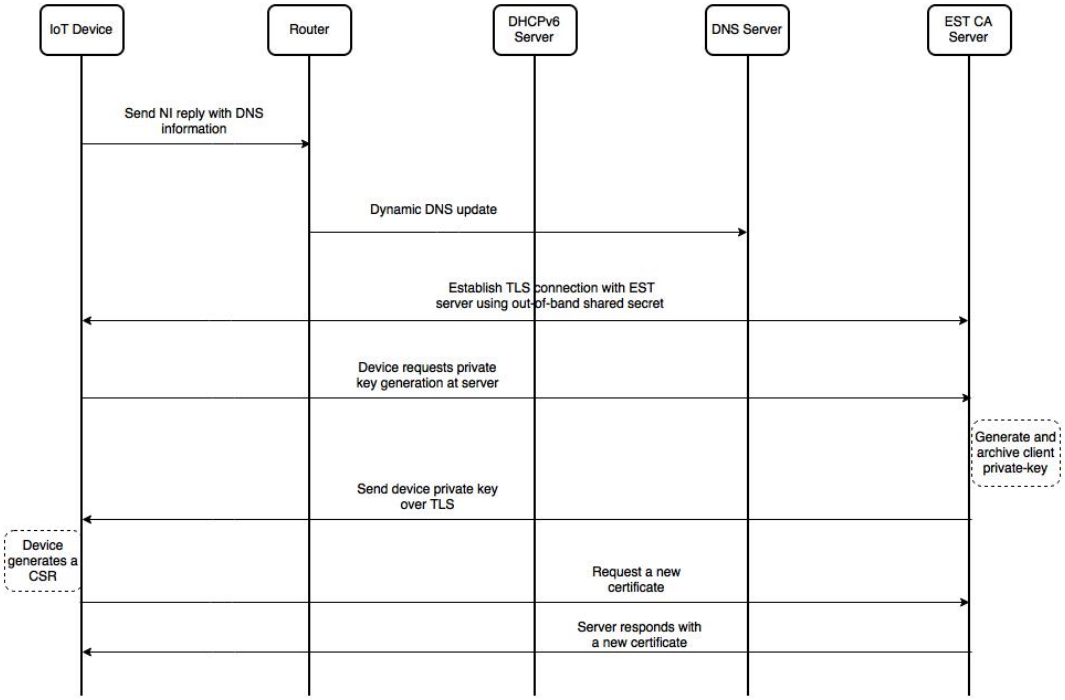

Figure 7.3: Message flow to demonstrate bootstrapping of IoT device.

Subsequently, the device establishes a TLS connection with the EST server through an out–of–band shared secret. The device and the EST server mutually authenticate each other based on the pre–shared secret key. Further, the device sends a private key generation request to the EST server. The EST server generates, archives and responds back with device's private key. Lastly, the device sends a certificate signing request to the EST server with the private key and auto–generated DNS name. After receiving and verifying the request, the EST server issues a new certificate for the device.

Chapter 8

Implementation and evaluation

This chapter examines constrained application protocol (CoAP) which is an application layer protocol intended for use in resource–constrained environments. Furthermore, we have demonstrated certificate enrollment procedure with an EST web–server based on RESTful web services implemented with portable resteasy library.

8.1 CoAP

Web APIs on Internet application depend fundamentally on Representational State Transfer (REST) architecture of the Web. CoAP is a web transfer protocol designed for realizing the REST architecture to use within constrained nodes and networks. Constrained nodes generally have 8-bit microcontrollers and constrained networks are build on 6LoWPANs. CoAP supports machine-to-machine (M2M) applications deployed in IoT scenarios such as home–automation, smart–grids, transportation, manufacturing, etc,. CoAP is based on a request–response interaction model between application end–points and supports service discovery, resource discovery employing web based URIs and media types [104].

Figure 8.1 demonstrates the abstract layering of CoAP protocol. Below is a list of features supported by CoAP [31, 67, 92]:

(a) CoAP is designed on REST based web architecture. Servers produce the available resources with a URL and CoAP clients can access the resources using methods such as GET, POST, PUT and DELETE.

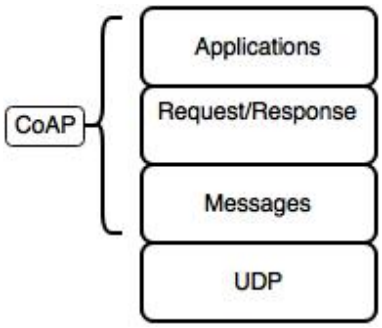

Figure 8.1: Abstract Layering in CoAP.

(b) CoAP is designed to be interoperable with HTTP. Since both protocols employ REST model, they can be mapped statelessly using cross–protocol proxies. Hence, CoAP resources can be accessed via HTTP and vice–versa.

(c) CoAP supports multiple payload formats including Extensible Markup Language (XML), JavaScript Object Notation (JSON), Concise Binary Object Representation (CBOR) and other user–defined data format.

(d) CoAP incorporates UDP binding and supports reliable unicast and multicast communication.

(e) CoAP supports security binding with DTLS.

(f) CoAP employs asynchronous message exchanges based on request–response model.

Generic implementation for CoAP based on RFC 7252 are available for multiple platforms. Some implementations are published under open–source licenses such as Apache 2.0 or MIT license. Implementation for constrained devices are generally written in C. Erbium is a REST based engine and CoAP implementation for Contiki operating system [5]. Libcoap can be used for constrained devices running Contiki or LWIP. Further, libcoap has been ported to TinyOS [12]. Tinydtls is an implementation of DTLS for constrained devices [14]. Microcoap is a C implementation for arduino and POSIX [6]. SMCP implementation offers CoAP stack for embedded environments and supports asynchronous request–response model [7].

Moreover, CoAP implementations for cloud servers, home gateways and smart–phones have also been published under open–source licenses. Cali-

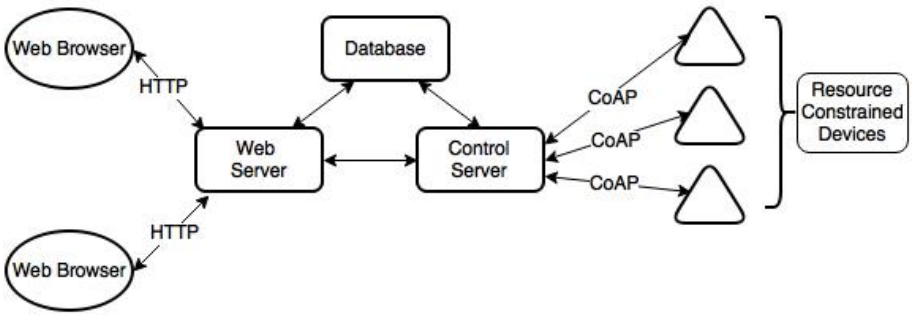

Figure 8.2: CoAP–HTTP system architecture.

fornium is java–based server side implementation of CoAP [2]. Node-coap is a client and server CoAP library written in javascript and can be installed in node.js [9]. CoAPthon is a CoAP library based on twisted framework and written in python [11]. In addition, Copper extension for Firefox enables access to CoAP resources from a web–browser [4]. Further, tools such as crosscoap are also available to implement systems with CoAP. Crosscoap is a CoAP server for translating CoAP requests to HTTP requests and HTTP responses to CoAP to be sent over to the CoAP client [8].

Figure 8.2 demonstrates a CoAP–HTTP system to facilitate HTTP client connections to CoAP resources and vice–versa.

8.2 Tools used

We provide a proof-of-concept implementation to demonstrate certificate enrollment procedure with an EST web–server. The web–server is based on Java API for RESTful Web Services (JAX-RS) specification. JAX-RS provides support for creating web–services based on REST architecture [13].

For implementation, we have used resteasy portable library. Resteasy provides implementation for JAX-RS specification, can run in any servlet container and supports tighter integration with JBoss application server [10]. Eclipse neon IDE was used for project development. Figure 8.3 shows the basic project structure.

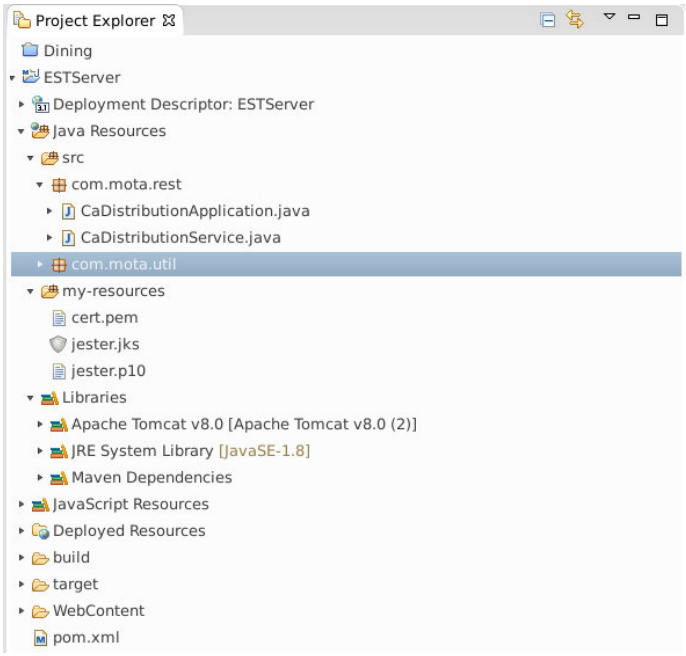

Figure 8.3: Screenshot of project structure.

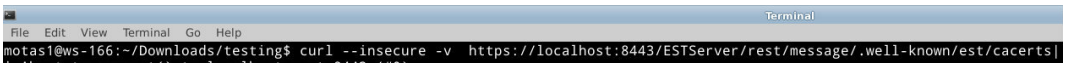

Figure 8.4: Screenshot–1 of operation GetCaCerts.

8.3 EST server prototype

The HTTPS enabled EST server listens on port 8443 for client requests. Curl is used to simulate client–side URL transfer requests.

The EST server supports following operations :

(a) Get the root certificate used by the EST web–server.

(b) Get the CA certs from the server using GET *cacerts*.

(c) Certificate enrollment request to the EST web–server using GET *simpleenroll*.

Figure 8.4 and Figure 8.5 demonstrate the operation to retrieve latest CA certs or explicit trust anchors. Figure 8.4 shows client–side request with curl

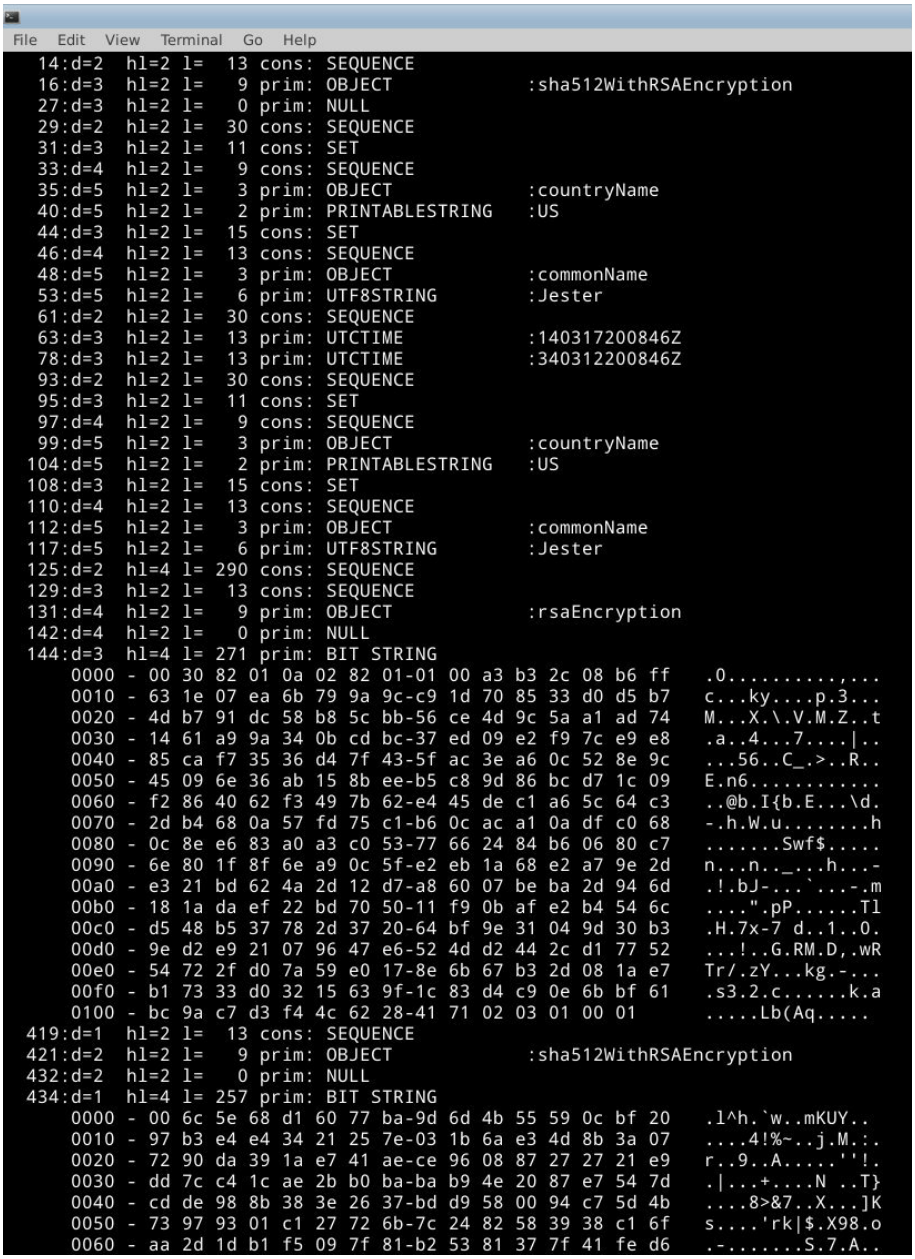

Figure 8.5: Screenshot–2 of operation GetCaCerts.

and Figure 8.5 shows the response which is decoded to base64 and converted to PEM format.

Chapter 9

Discussion

9.1 PKI challenges for IoT

PKI is the trusted solution for secure communication with current Internet and can be further adopted to provide authentication, confidentiality and integrity for IoT deployment scenarios. Moreover, management of digital certificates by a CA is the core concept of PKI ecosystem. With this thesis work, we have highlighted different certificate management protocols such as SCEP, EST, CMP and CMC. Furthermore, we have motivated the use of EST protocol for certificate management operations within IoT environment. EST utilizes Cisco's NGE by using ECC instead of RSA encryption. Besides, EST employs Transport Layer Security (TLS) for secure transport of certificates and messages. In addition, EST also supports CA server–side private key generation for resource constrained devices and this can be an important aspect as IoT devices may not have enough entropy source and power for generation of a random private key.

Moreover, this thesis has focused on various challenges to adopt PKI solution for providing secure communication in IoT deployment scenario. We have considered aspects such as naming of IoT devices, peer authentication, private key generation, choice of cryptographic algorithm, scalable database solution, life–span of digital certificates, managing high volume of issuance and number of certificates per device to support large scale deployment of IoT devices.

9.2 Certificate Transparency with IoT PKI

Certificate Transparency aims at supporting early detection of forged certificates, rogue CAs and offers a better oversight to the current PKI model. It advocates public scrutiny over CA operations by domain owners, CAs and domain users through certificate logs, monitors and auditors. Certificate logs consist of cryptographic proofs which simplify the verification of log consistency and certificate legitimacy. In order to verify the legitimacy of a certificate, a client can request for an audit proof. A log containing 1 billion certificates will require the client to fetch 28 node hashes of MHT which is approximately 30 KB data [96]. Furthermore, the efficacy of certificate transparency is vital for IoT deployment scenario. It is critical to secure end–to–end device communication through digital certificates. A compromised CA can generate fake digital certificates for IoT devices and introduce a serious vulnerability in the PKI security solution. However, considering the large scale deployment of IoT devices the CA certificate database can be handling billions of digital certificates. In order to verify the legitimacy of a digital certificate a resource constrained device will have to download additional 30–35 KB data. This will increase the performance overhead for the devices. We propose to offload this task to the IoT gateway for the deployment model in Section 7.1.

9.3 Future work

The work in this thesis can be extended by setting up an end–to–end testbed as proposed in Section 7.1 with cloud–based EST , DHCPv6 and DNS servers, raspberry pi or arduino devices and an access point. The testbed will aid in analyzing the feasibility of bootstrap mechanism proposed in Section 7.2. Additionally, we will like to implement a large scale database solution for the CA based on MHT to investigate performance, efficiency and scalability with various certificate management operations and certificate transparency as discussed in Section 4.3.2 and Section 6.2.

Chapter 10

Conclusion

This thesis has mainly focused on the deployment of an efficient, scalable, trustworthy and flexible PKI solution for ensuring penetration–resistant device–device authentication, device authorisation and inter–device communication within an IoT deployment scenario. In addition, the work done provides a comprehensive survey of the key aspects, challenges and the importance of deploying a PKI security solution for IoT ecosystem. We have reviewed various security attacks on different IoT deployment scenarios leading to disabling of system operation, economic loss and privacy threat to sensitive user data. Moreover, we have investigated the applicability of different certificate management protocols such SCEP, EST, CMC and CMP to be employed within a resource constrained environment. Subsequently, we have proposed and motivated the adoption of EST as the suitable certificate management protocol for IoT deployment scenario.

Additionally, we have proposed a PKI deployment model for IoT environment and the bootstrap mechanism to bring up an IoT device and provision it with X.509 digital certificate which ensures device authenticity, data confidentiality and integrity. Besides, the thesis has examined the need for naming and identification of IoT devices which is a critical step in the bootstrap process. We have incorporated DNS name auto–configuration scheme for IoT devices as it assists in easy monitoring and controlling the devices remotely. Furthermore, we have investigated the feasibility of applying certificate transparency over PKI in IoT deployments to introduce trustworthy CAs. Implementation and evaluation work carried out in the thesis examines CoAP which is an application layer protocol intended for use in resource–constrained environments. In addition, we have demonstrated certificate enrollment procedure with an EST web–server based on RESTful web services implemented with portable resteasy library.

Bibliography

[1] Bonjour for developers - apple developer. https://developer.apple.com/bonjour/.

[2] Californium (cf) coap framework. http://www.eclipse.org/californium/ Accessed on 20.06.2016.

[3] Certificate transparency. https://www.certificate-transparency.org/ Accessed on 09.06.2016.

[4] Copper (cu) coap user-agent - javascript coap implementation. http://people.inf.ethz.ch/mkovatsc/copper.php Accessed on 20.06.2016.

[5] Erbium (er) rest engine - c coap implementation. http://people.inf.ethz.ch/mkovatsc/erbium.php Accessed on 20.06.2016.

[6] Github - 1248/microcoap: A small coap implementation for microcontrollers. https://github.com/1248/microcoap Accessed on 20.06.2016.

[7] Github - darconeous/smcp: A flexible coap stack for embedded devices and computers. rfc7252 compatible. https://github.com/darconeous/smcp/ Accessed on 20.06.2016.

[8] Github - ibm-security-innovation/crosscoap: Coap-to-http translator proxy. https://github.com/ibm-security-innovation/crosscoap Accessed on 20.06.2016.

[9] Github - mcollina/node-coap: Coap - node.js style. https://github.com/mcollina/node-coap Accessed on 20.06.2016.

[10] Github - resteasy/resteasy: Rest and jaxrs. https://github.com/resteasy/Resteasy Accessed on 20.06.2016.

[11] Github - tanganelli/coapthon: Coapthon is a python library to the coap protocol aligned with the rfc. https://github.com/Tanganelli/CoAPthon Accessed on 20.06.2016.

[12] libcoap: C-implementation of coap. https://libcoap.net/ Accessed on 20.06.2016.

[13] Project kenai. https://jax-rs-spec.java.net/ Accessed on 20.06.2016.

[14] tinydtls — projects.eclipse.org. https://projects.eclipse.org/projects/iot.tinydtls Accessed on 20.06.2016.

[15] Cmp and cmc comparison. Discussion, 3GPP, May 2002. http://www.3gpp.org/ftp/tsg_sa/wg3_security/tsgs3_24_helsinki/docs/pdf/S3-020366.pdf.

[16] Taking encryption to the next level: Enrollment over secure transport strengthens adoption of elliptic curve cryptography. White paper, Cisco, April 2014. http://blogs.cisco.com/security/taking-encryption-to-the-next-level-enrollment-over-secure-transport-strengthens-adoption-of-elliptic-curve-cryptography.

[17] Focus on startups and small vendors as drivers for iot innovation. Tech. rep., Gartner, Inc., June 2015. https://www.gartner.com/doc/3083422/focus-startups-small-vendors-drivers.

[18] The future of smart cities: Cyber-physical infrastructure risk. https://ics-cert.us-cert.gov/Future-Smart-Cities-Cyber-Physical-Infrastructure-Risk, August 2015.

[19] Mass adoption of the internet of things will create new opportunities and challenges for enterprises. Tech. rep., Gartner, Inc., February 2015. https://www.gartner.com/doc/2994817/mass-adoption-internet-things-create.

[20] The practicalities of implementing iot. Tech. rep., Gartner, Inc., October 2015. https://www.gartner.com/doc/3143217/practicalities-implementing-iot.

[21] Cisco ios est client with libest ca server. White paper, Cisco, 2016. http://www.cisco.com/c/dam/en_us/about/doing_business/trust-center/docs/cisco-ios-est-client-libest-ca-server.pdf.

[22] Est compared to scep, cmc and cmp. White paper, Cisco, 2016. http://www.cisco.com/web/about/doing_business/trust-center/docs/public-key-infrastructure-provisioning-with-est.pdf.

[23] Iot: Key lessons to date and action plan for 2016. Tech. rep., Gartner, Inc., February 2016. https://www.gartner.com/doc/3210021?refval=&pcp=mpe.

[24] Pki: The security solution for the internet of things. Tech. rep., Digicert, April 2016. https://www.digicert.com/internet-of-things/iot-pki-whitepaper.htm.

[25] Abomhara, M., and Køien, G. M. Cyber security and the internet of things: Vulnerabilities, threats, intruders and attacks. *Journal of Cyber Security 4*, 65–88.

[26] Adams, C., Farrell, S., Kause, T., and Mononen, T. Internet x.509 public key infrastructure certificate management protocol (cmp). RFC 4210, RFC Editor, September 2005. http://www.rfc-editor.org/rfc/rfc4210.txt.

[27] Andersson, L., and Madsen, T. Provider provisioned virtual private network (vpn) terminology. RFC 4026, RFC Editor, March 2005.

[28] Authority, D. Transportation. http://www.deviceauthority.com/transportation/, June 2016.

[29] Authority, D. Utilities. http://www.deviceauthority.com/utilities/, June 2016.

[30] Benantar, M. The internet public key infrastructure. *IBM Systems Journal 40*, 3 (2001), 648–665.

[31] Bergmann, O., Hillmann, K. T., and Gerdes, S. A coap-gateway for smart homes. In *Computing, Networking and Communications (ICNC), 2012 International Conference on* (2012), IEEE, pp. 446–450.

[32] Bhargavan, K., Leurent, G., et al. Transcript collision attacks: Breaking authentication in tls, ike, and ssh.

[33] Cheng, Y., Naslund, M., Selander, G., and Fogelstrom, E. Privacy in machine-to-machine communications a state-of-the-art survey. In *Communication Systems (ICCS), 2012 IEEE International Conference on* (2012), IEEE, pp. 75–79.

[34] Chokhani, S., Ford, W., Sabett, R., Merrill, C., and Wu, S. Internet x.509 public key infrastructure certificate policy and certification practices framework. RFC 3647, RFC Editor, November 2003.

[35] ComputerWeekly. Schoolboy hacker derails poland's tram network. `http://www.computerweekly.com/news/2240084537/Schoolboy-hacker-derails-Polands-tram-network`, November 2008.

[36] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and Polk, W. Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile. RFC 5280, RFC Editor, May 2008. `http://www.rfc-editor.org/rfc/rfc5280.txt`.

[37] Crawford, M., and Haberman, B. Ipv6 node information queries. RFC 4620, RFC Editor, August 2006. `http://www.rfc-editor.org/rfc/rfc4620.txt`.

[38] Dierks, T., and Allen, C. The tls protocol version 1.0. RFC 2246, RFC Editor, January 1999. `http://www.rfc-editor.org/rfc/rfc2246.txt`.

[39] Dierks, T., and Rescorla, E. The transport layer security (tls) protocol version 1.1. RFC 4346, RFC Editor, April 2006. `http://www.rfc-editor.org/rfc/rfc4346.txt`.

[40] Dierks, T., and Rescorla, E. The transport layer security (tls) protocol version 1.1. RFC 4346, RFC Editor, April 2006. `http://www.rfc-editor.org/rfc/rfc4346.txt`.

[41] Dierks, T., and Rescorla, E. The transport layer security (tls) protocol version 1.2. RFC 5246, RFC Editor, August 2008. `http://www.rfc-editor.org/rfc/rfc5246.txt`.

[42] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., and Carney, M. Dynamic host configuration protocol

for ipv6 (dhcpv6). RFC 3315, RFC Editor, July 2003. `http://www.rfc-editor.org/rfc/rfc3315.txt`.

[43] Durumeric, Z., Kasten, J., Adrian, D., Halderman, J. A., Bailey, M., Li, F., Weaver, N., Amann, J., Beekman, J., Payer, M., et al. The matter of heartbleed. In *Proceedings of the 2014 Conference on Internet Measurement Conference* (2014), ACM, pp. 475–488.

[44] Eastlake, D., Schiller, J., and Crocker, S. Randomness requirements for security. BCP 106, RFC Editor, June 2005. `http://www.rfc-editor.org/rfc/rfc4086.txt`.

[45] Economist, T. Hacking the planet. `http://www.economist.com/news/leaders/21657811-internet-things-coming-now-time-deal-its-security-flaws-hacking`, July 2015.

[46] Fisher, D. Final report on diginotar hack shows total compromise of ca servers, October 2012.

[47] Fox, B., and LaMacchia, B. Online certificate status checking in financial transactions: the case for re-issuance. In *Financial Cryptography* (1999), Springer, pp. 104–117.

[48] Gigaom. The internet of things needs a new security model. which one will win? `https://gigaom.com/2014/01/22/the-internet-of-things-needs-a-new-security-model-which-one-will-win/`, January 2014.

[49] Gluck, Y., Harris, N., and Prado, A. Breach: reviving the crime attack. *Unpublished manuscript* (2013).

[50] Gura, N., Patel, A., Wander, A., Eberle, H., and Shantz, S. C. Comparing elliptic curve cryptography and rsa on 8-bit cpus. In *Cryptographic hardware and embedded systems-CHES 2004*. Springer, 2004, pp. 119–132.

[51] Gutmann, P., and Marcon, J. Simple Certificate Enrolment Protocol. Internet-Draft draft-gutmann-scep-02, Internet Engineering Task Force, Mar. 2016. Work in Progress.

[52] Harkins, D., and Carrel, D. The internet key exchange (ike). RFC 2409, RFC Editor, November 1998. `http://www.rfc-editor.org/rfc/rfc2409.txt`.

[53] Hui, J., and Thubert, P. Compression format for ipv6 datagrams over ieee 802.15.4-based networks. RFC 6282, RFC Editor, September 2011. http://www.rfc-editor.org/rfc/rfc6282.txt.

[54] Jeong, J., Park, S., Beloeil, L., and Madanapalli, S. Ipv6 router advertisement options for dns configuration. RFC 6106, RFC Editor, November 2010.

[55] Jeong, J. P., Jung-Soo, P., and sejunlee@skku.edu. DNS Name Autoconfiguration for Internet of Things Devices. Internet-Draft draft-jeong-homenet-device-name-autoconf-04, Internet Engineering Task Force, Apr. 2016. Work in Progress.

[56] Jiang, D., and ShiWei, C. A study of information security for m2m of iot. In *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on* (2010), vol. 3, IEEE, pp. V3–576.

[57] Kafle, V., Fukushima, Y., and Harai, H. Connecting the world through trustable internet of things. In *ITU Kaleidoscope: Trust in the Information Society (K-2015), 2015* (2015), IEEE, pp. 1–8.

[58] Kaliski, B. Pkcs 10: Certification request syntax version 1.5. RFC 2314, RFC Editor, March 1998.

[59] Kaliski, B. Pkcs 7: Cryptographic message syntax version 1.5. RFC 2315, RFC Editor, March 1998. http://www.rfc-editor.org/rfc/rfc2315.txt.

[60] Kaufman, C. Internet key exchange (ikev2) protocol. RFC 4306, RFC Editor, December 2005. http://www.rfc-editor.org/rfc/rfc4306.txt.

[61] Kause, T., and Peylo, M. Internet x.509 public key infrastructure – http transfer for the certificate management protocol (cmp). RFC 6712, RFC Editor, September 2012.

[62] Kent, S., and Seo, K. Security architecture for the internet protocol. RFC 4301, RFC Editor, December 2005. http://www.rfc-editor.org/rfc/rfc4301.txt.

[63] Khemissa, H., and Tandjaoui, D. A lightweight authentication scheme for e-health applications in the context of internet of things. In *Next Generation Mobile Applications, Services and Technologies, 2015 9th International Conference on* (2015), IEEE, pp. 90–95.

[64] Kiran, S., Lareau, P., and Lloyd, S. Pki basics - a technical perspective. Tech. rep., PKI Forum, Nov 2002. `http://www.oasis-pki.org/pdfs/PKI_Basics-A_technical_perspective.pdf`.

[65] Kizza, J. M. Computer network fundamentals. In *Guide to Computer Network Security*. Springer, 2013, pp. 3–41.

[66] Kozik, R., and Choras, M. Current cyber security threats and challenges in critical infrastructures protection. In *Informatics and Applications (ICIA), 2013 Second International Conference on* (2013), IEEE, pp. 93–97.

[67] Kuladinithi, K., Bergmann, O., Pötsch, T., Becker, M., and Görg, C. Implementation of coap and its application in transport logistics. *Proc. IP+ SN, Chicago, IL, USA* (2011).

[68] Langley, A. Google online security blog: Enhancing digital certificate security, January 2013.

[69] Langley, A. Google online security blog: Further improving digital certificate security, December 2013.

[70] Laurie, B. Certificate transparency. *Queue 12*, 8 (2014), 10.

[71] Laurie, B., Langley, A., and Kasper, E. Certificate transparency. RFC 6962, RFC Editor, June 2013.

[72] Lee, S., Jeong, J., and Park, J. Dns name autoconfiguration for iot home devices. In *2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops (WAINA)* (2015), IEEE, pp. 131–134.

[73] Li, Z., Yin, X., Geng, Z., Zhang, H., Li, P., Sun, Y., Zhang, H., and Li, L. Research on pki-like protocol for the internet of things. In *Measuring Technology and Mechatronics Automation (ICMTMA), 2013 Fifth International Conference on* (2013), IEEE, pp. 915–918.

[74] Lindemann, R. Certificate management system, March 2007. US Patent App. 11/729,735.

[75] Ltd., C. Heartbleed bug. `http://heartbleed.com/`, April 2014.

[76] Madhura, P., Bilurkar, N., Jain, P., and Ranjith, J. A survey on internet of things: security and privacy issues. *IJITR 3*, 3 (2015), 2069–2074.

[77] Mahalle, P. N., Prasad, N. R., and Prasad, R. Object classification based context management for identity management in internet of things. *International Journal of Computer Applications 63*, 12 (2013).

[78] Manyika, J., Chui, M., Bisson, P., Stamford, Woetzel, J., Dobbs, R., Bughin, J., and Aharon, D. Unlocking the potential of the internet of things. Tech. rep., Mckinsey&Company, June 2015. `http://www.mckinsey.com/business-functions/business-technology/our-insights/the-internet-of-things-the-value-of-digitizing-the-physical-world`.

[79] Merkle, R. C. A certified digital signature. In *Advances in Cryptology—CRYPTO'89 Proceedings* (1989), Springer, pp. 218–238.

[80] Miller, C., and Valasek, C. Remote exploitation of an unaltered passenger vehicle. *Black Hat USA* (2015).

[81] Muñoz, J. L., Forne, J., Esparza, O., and Soriano, M. Certificate revocation system implementation based on the merkle hash tree. *International Journal of Information Security 2*, 2 (2004), 110–124.

[82] Myers, M., Ankney, R., Malpani, A., Galperin, S., and Adams, C. X.509 internet public key infrastructure online certificate status protocol - ocsp. RFC 2560, RFC Editor, June 1999. `http://www.rfc-editor.org/rfc/rfc2560.txt`.

[83] Naor, M., and Nissim, K. Certificate revocation and certificate update. *Selected Areas in Communications, IEEE Journal on 18*, 4 (2000), 561–570.

[84] Narten, T., Nordmark, E., Simpson, W., and Soliman, H. Neighbor discovery for ip version 6 (ipv6). RFC 4861,

RFC Editor, September 2007. http://www.rfc-editor.org/rfc/rfc4861.txt.

[85] Nystrom, M., and Kaliski, B. Pkcs 10: Certification request syntax specification version 1.7. RFC 2986, RFC Editor, November 2000.

[86] Nystrom, M., and Kaliski, B. Pkcs 9: Selected object classes and attribute types version 2.0. RFC 2985, RFC Editor, November 2000.

[87] on Security, K. Fbi: Smart meter hacks likely to spread. http://krebsonsecurity.com/2012/04/fbi-smart-meter-hacks-likely-to-spread/, April 2012.

[88] PCWorld. Internet-connected hello barbie doll can be hacked. http://www.pcworld.com/article/3012220/security/internet-connected-hello-barbie-doll-can-be-hacked.html, December 2015.

[89] Perlman, R. An overview of pki trust models. *Network, IEEE 13*, 6 (Nov 1999), 38–43.

[90] Pritikin, M., Yee, P., and Harkins, D. Enrollment over secure transport. RFC 7030, RFC Editor, October 2013.

[91] Ramsdell, B., and Turner, S. Secure/multipurpose internet mail extensions (s/mime) version 3.2 message specification. RFC 5751, RFC Editor, January 2010. http://www.rfc-editor.org/rfc/rfc5751.txt.

[92] Raza, S., Shafagh, H., Hewage, K., Hummen, R., and Voigt, T. Lithe: Lightweight secure coap for the internet of things. *IEEE Sensors Journal 13*, 10 (2013), 3711–3720.

[93] Rescorla, E., and Modadugu, N. Datagram transport layer security. RFC 4347, RFC Editor, April 2006. http://www.rfc-editor.org/rfc/rfc4347.txt.

[94] Roberts, P. Phony ssl certificates issued for google, yahoo, skype, others, October 2011. https://threatpost.com/phony-ssl-certificates-issued-google-yahoo-skype-others-032311/75061/.

[95] Rudner, M. Cyber-threats to critical national infrastructure: an intelligence challenge. *International Journal of Intelligence and CounterIntelligence 26*, 3 (2013), 453–481.

[96] Ryan, M. D. Enhanced certificate transparency and end-to-end encrypted mail. In *NDSS* (2014).

[97] Schaad, J. Internet x.509 public key infrastructure certificate request message format (crmf). RFC 4211, RFC Editor, September 2005.

[98] Schaad, J., and Myers, M. Certificate management over cms (cmc). RFC 5272, RFC Editor, June 2008. `http://www.rfc-editor.org/rfc/rfc5272.txt`.

[99] Schaad, J., and Myers, M. Certificate management over cms (cmc): Transport protocols. RFC 5273, RFC Editor, June 2008.

[100] Schneier, B. *Secrets and lies: digital security in a networked world*. John Wiley & Sons, 2011.

[101] Schukat, M., and Cortijo, P. Public key infrastructures and digital certificates for the internet of things. In *Signals and Systems Conference (ISSC), 2015 26th Irish* (2015), IEEE, pp. 1–5.

[102] SecurityWeek.Com. Ssl key exposed in ruggedcom switches. `http://www.securityweek.com/ssl-key-exposed-ruggedcom-switches`, August 2012.

[103] Sermersheim, J. Lightweight directory access protocol (ldap): The protocol. RFC 4511, RFC Editor, June 2006. `http://www.rfc-editor.org/rfc/rfc4511.txt`.

[104] Shelby, Z., Hartke, K., and Bormann, C. The constrained application protocol (coap). RFC 7252, RFC Editor, June 2014. `http://www.rfc-editor.org/rfc/rfc7252.txt`.

[105] Thomson, S., Narten, T., and Jinmei, T. Ipv6 stateless address autoconfiguration. RFC 4862, RFC Editor, September 2007. `http://www.rfc-editor.org/rfc/rfc4862.txt`.

[106] Trcek, D. *Managing Information Systems Security and Privacy*. Springer, 2006.

[107] V3.co.uk. Ces 2015: Ftc warns of internet of things security risks. `http://www.v3.co.uk/v3-uk/news/2389013/ces-2015-ftc-warns-of-internet-of-things-security-risks`, January 2015.

[108] Vixie, P., Thomson, S., Rekhter, Y., and Bound, J. Dynamic updates in the domain name system (dns update). RFC 2136, RFC Editor, April 1997. `http://www.rfc-editor.org/rfc/rfc2136.txt` Accessed on 20.05.2016.

[109] WIRED. It's insanely easy to hack hospital equipment. `https://www.wired.com/2014/04/hospital-equipment-vulnerable/`, April 2014.

[110] WIRED. A cyberattack has caused confirmed physical damage for the second time ever. `https://www.wired.com/2015/01/german-steel-mill-hack-destruction/`, January 2015.

[111] WIRED. Researchers hacked a model s, but tesla's already released a patch. `https://www.wired.com/2015/08/researchers-hacked-model-s-teslas-already/`, August 2015.

[112] WIRED. Samsung's smart fridge just got pwned. how about the rest of your smart home? `www.makeuseof.com/tag/samsung-smart-fridge-pwned/`, August 2015.

[113] Yang, L., Yu, P., Bailing, W., Xuefeng, B., Xinling, Y., and Gen, L. The internet of things security architecture based ibe integration with the pki/ca.

[114] Zeilenga, K. Lightweight directory access protocol (ldap) schema definitions for x.509 certificates. RFC 4523, RFC Editor, June 2006.

[115] Zimmermann, P. R. *The official PGP user's guide*. MIT press, 1995.

Appendix A

First appendix

A.1 Types of PKI model

A.1.1 A single CA or monopoly model

This model considers only one CA in the world for all certificate requesting clients. All devices, applications or equipments have to be pre–configured with the public key of this CA. figure demonstrates the monopoly model as discussed above. All the certificates should be requested from the organization running the CA. Even though it is the simplest model it has numerous weak–points against adoption of this strategy :

(a) There is no such single organization which can be trusted globally by all countries, companies, etc,.

(b) A single CA model is highly inconvenient, cumbersome to manage and poses security concerns. It is expensive and sub–optimal for a distant and unrelated organization to obtain digital certificate.

(c) In general, periodic changes in the keys is considered to be a good security practice, for example, migrating to higher key size. In case a CA's private key is compromised within the single CA model, all the clients have to re–configured with the new public key of CA.

(d) Single CA is essentially a monopoly model and gives privilege to only one organization for handling digital certificate life–cycle. Within any monopoly model an organization can charge excessive fees for granting certificates.

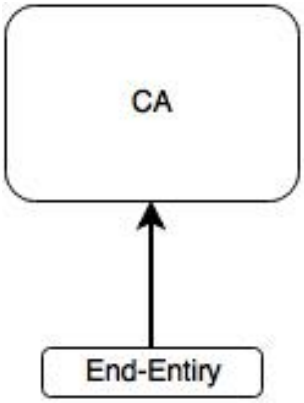

Figure A.1: Single CA monopoly model.

Figure A.1 shows the monopoly model as discussed above.

A.1.2 A single CA with RA model

This model essentially follows single CA model and all the end–points have to be configured with the public key of the single CA available. However, there are multiple RA's available which are trusted by the single CA. The CA stores the list of all public key's of RA's under it. RA is responsible for handling the certificate request from the end–points, authenticating or verifying mapping between the name and key of the end–points, sign the request and forward it to the CA. The CA creates the certificate for the requesting end–point. With this model, the user is never interacting directly with the CA. Revocation of key of any RA can be handled easily with this model, however, this model still has other drawbacks of monopoly model.

Figure A.2 demonstrates the single CA with RA model as discussed above.

A.1.3 Oligarchy CA model

In oligarchy model multiple CAs exist. Hence, the end–points have to be configured with a set of public keys. An end–point can choose and request public keys from a list of CAs. This model enforces competition at CA level and hence prevents excessive charging of fees for granting certificates. The security of this model depends on all the configured CA public keys to be secure. In case, any configured key is compromised an attacker can generate

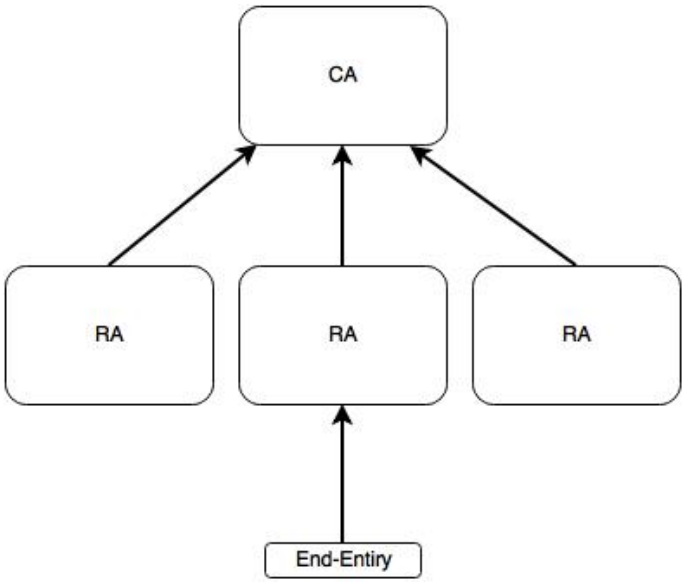

Figure A.2: Single CA with RA model.

forged certificates for end-points which will be trusted by all entities. So this model has more attack points as compared to monopoly model.

Figure A.3 demonstrates the oligarchy model as discussed above.

A.1.4 Configured plus Delegated CA model

This model is essentially implemented in all the web–browsers and similar to the models discussed above. In this model, the configured CA can sign

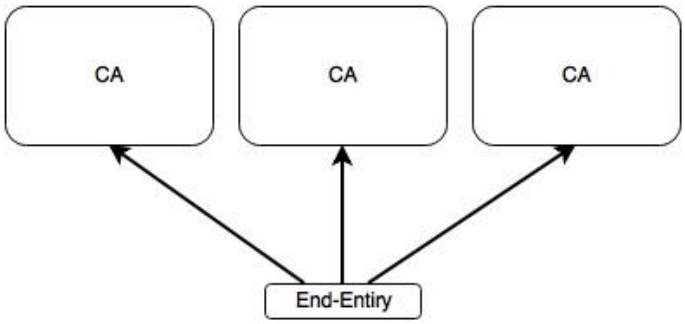

Figure A.3: Oligarchy model.

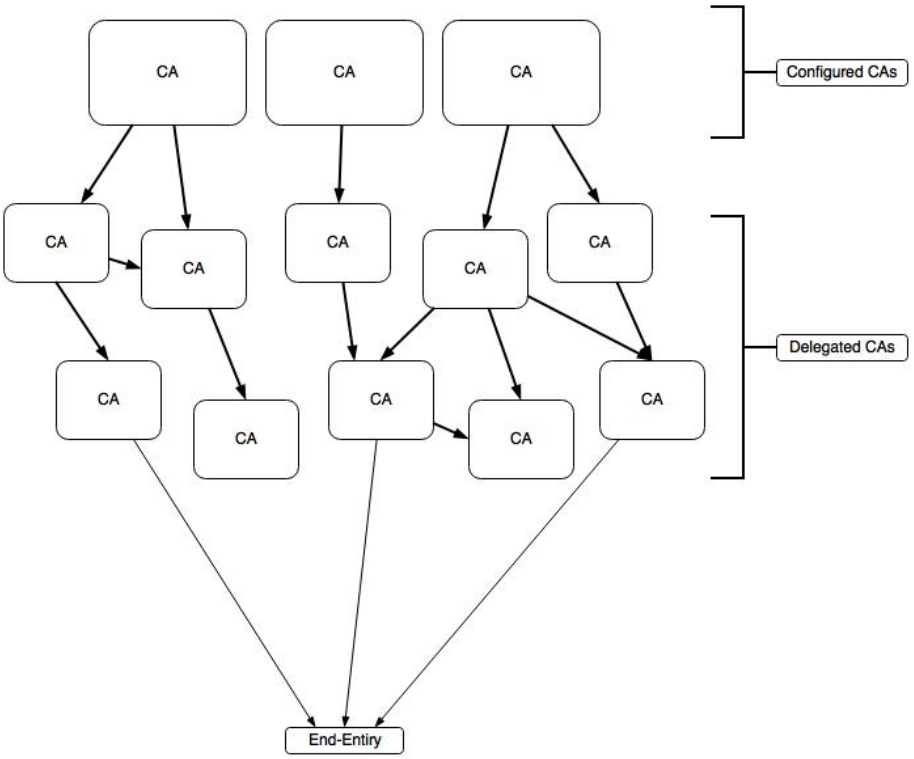

Figure A.4: Configured plus delegated CA model.

certificates authorising other CAs to grant digital certificates for end–points. The CAs whose keys are pre–configured with the end–points are called configured CAs and the CAs who are authorized by configured CAs are called delegated CAs. Certificates granted by either configured or delegated CAs are completely trusted by end–points. Certificates granted by delegated CAs include a certificate chain that link the CA to a configured CA. This model allows end–points to obtain certificates from multiple points either a configured or a delegated CA. This model is more secure as a user can obtain a certificate from a CA which is located in close proximity. However, compromise of any CA in the certificate chain pose similar threats to the security as discussed in above models. In case a delegated CA is compromised, it may go unnoticed for a significant period of time leaving ample opportunity for an attacker to pose as a delegated CA and sign certificated for any of the end–points. We discuss some such known attacks in Section 6.2.

Figure A.4 demonstrates the configured plus delegated CA model as discussed above.

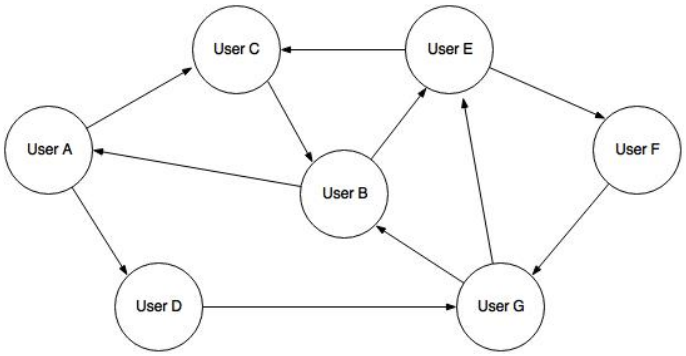

Figure A.5: Anarchy CA model.

A.1.5 Anarchy model

This model has been extensively used in public domain Pretty Good Privacy (PGP). The central idea is that the end–points start with a pre–configured list of public keys. Moreover, the end–points learn new public keys by downloading them from a central database or by receiving them through an email. With time, the end–points accumulate keys from other end–points and designate them as trusted. End–points are free to choose other trusted end–points. End–points also distribute with their key a collection of certifying signatures from other end–points. These eventually will lead to a decentralized fault-tolerant web of trust for all public keys [115].

Figure A.5 demonstrates the anarchy model as discussed above.

Other PKI models that have been proposed include top–down, flexible bottom–up, up–cross–down, etc., [89].

A.2 IoT economic impact

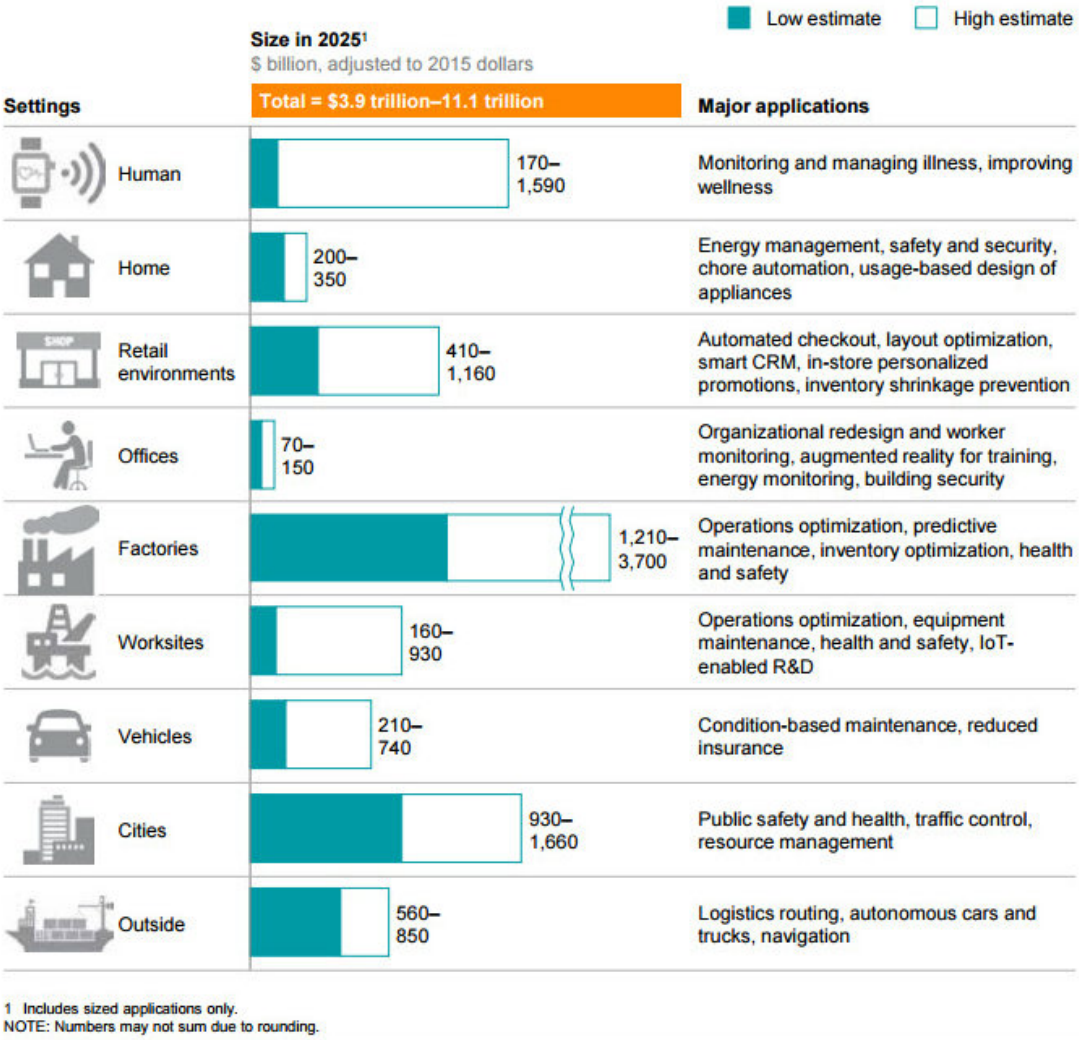

Figure A.6: Potential economic impact of IoT by 2025 estimated to be 3.9 to 11.1 trillion dollars [78].