Scuola di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea in Informatica

Tesi di Laurea

ANALISI E VALUTAZIONE DELLE TECNOLOGIE DI AUTENTICAZIONE E AUTORIZZAZIONE MULTILIVELLO PER SISTEMI IOT IN AMBIENTI FOG/CLOUD.

ANALYSIS AND EVALUATION OF AUTHENTICATION AND MULTILEVEL AUTHORIZATION TECHNOLOGIES FOR IOT SYSTEMS IN FOG/CLOUD ENVIRONMENTS.

ALESSANDRO MINI

Relatore: Tommaso Pecorella
Correlatore: Paolo Lollini

Anno Accademico 2019-2020

INDICE

ELENCO DELLE FIGURE

"The five most efficient cyber defenders are: Anticipation, Education, Detection, Reaction and Resilience. Do remember: "Cybersecurity is much more than an IT topic."
— Stephane Nappo

ABSTRACT

L' *Internet of Things* è un fenomeno tecnologico che riguarda l'estensione di ciò che identifichiamo come "internet" a oggetti e luoghi concreti.

Per via della forte trasformazione tecnologica di questi anni, sempre più oggetti assumono la veste di "Smart", <intelligenti>, in grado di comunicare dati e raccogliere informazioni autonomamente.

Si producono frigoriferi di casa che controllano automaticamente la temperatura, sveglie che si regolano da sole in base al traffico, reti elettriche intelligenti, tecnologia indossabile e strumenti di *e-health* come dispositivi di rilevazione di incidenti e assistenza per gli anziani in grado rilevare problemi autonomamente. Sempre più oggetti divengono parte di questa trasformazione tecnologica.

La tecnologia vista come un ecosistema isolato, complesso e per pochi decade poiché questa si fonde quasi biologicamente alle nostre vite. In quest'ottica rimane critica la domanda della sicurezza: l'eterogeneità dei dispositivi che accederà alla rete e l'enormità di dati che verranno quotidianamente inviati sarà sicura? Quale può essere un approccio efficiente a questo proposito?

L'obbiettivo principale di questa tesi triennale è studiare il problema dell'autenticazione ed autorizzazione multi-livello in ambienti integrati con tecnologie Cloud e FoG.

1

INTRODUZIONE

Internet of Things (IoT) è un termine utilizzato per dare un nome agli oggetti connessi ad internet che non siano i classici dispositivi come computer o smartphone. IoT è ad esempio un frigorifero che ordina automaticamente un prodotto quando "si accorge" che è finito. IoT è una casa che accende il riscaldamento o chiude le porte in autonomia.

Dispositivi di questo tipo sono risorse tanto importanti quanto "pericolose" per via del loro ambito di applicazione che in molte realtà diviene critico (basti pensare ad un impianto di *life monitoring*).

Il mio interesse per questo argomento nasce dal desiderio di conoscere approfonditamente la sicurezza informatica e la sicurezza delle reti. Esistono molte ricerche nell'ambito della *cybersecurity* riguardo ai problemi di reti di questo tipo. In questa tesi verrà trattato un problema specifico, quello dell'autenticazione e dell'autorizzazione.

1.1 RETI DI DISPOSITIVI

Nella visione dell'*Internet of Things*, gli oggetti creano una rete pervasiva ed interconnessa avvalendosi di molteplici tecnologie di comunicazione tra cui 4G, 5G, comunicazione M2M (*Machine to Machine*), Lora, 6G, standard come IEEE 802.15.4, 802.1.X, 802.11 (WiFi) e tecnologie che operano ad alto livello.

Nel mondo dell'IoT non sono presenti degli standard ufficiali [15], in questo ambito confluiscono buona parte delle tecnologie "moderne". Una rete di dispositivi ha dei requisiti necessari tra cui:

- Distribuzione equilibrata del carico di lavoro sulla rete.
- Capacità di gestire quantità elevate di dispositivi connessi contemporaneamente.
- La rete deve essere affidabile ed a latenza minima.

- Il traffico deve essere protetto, almeno la parte del traffico "riservata".
- Si devono implementare più standard di telecomunicazioni possibili (3GPP, TD-SCDMA, Wireless LAN) evitando l'effetto *ping pong* tra i diversi standard.

Le reti di dispositivi possono essere di varie tipologie a seconda della natura dei dispositivi che che vi fanno parte e di particolari richieste. Si hanno quindi reti domestiche (*es. per home automation*), reti per i trasporti, reti di servizi e reti nazionali.

1.1.1 *Struttura di una rete IoT*

In una generica rete di dispositivi si identificano tre tipologie di nodi [19]:

- **Nodi sensore:** Sono dei nodi della rete che corrispondono ai devices.
- **Nodi Cluster:** Sono dei nodi della rete che servono ad amministrare i nodi sensori.
- **Nodo Gateway:** Sono dei nodi che si occupano di comunicare con l'esterno e acquisire periodicamente informazioni in modo asincrono. (generalmente).

Un requisito importante è che venga preservata la riservatezza delle informazioni. Questo si può realizzare attraverso l'uso di metodi crittografici e l'adozione di protocolli sicuri, ben noti e testati. Sarà quindi illustrato un generico flusso di lavoro per l'ingresso di un dispositivo all'interno di una rete:

1. Un utente U vuole entrare nella rete, esegue una richiesta al *gateway*.
2. Il Gateway (o altri nodi delegati) valuta se U può entrare o meno in comunicazione con la rete, in caso di decisione positiva generalmente si crea una sessione crittografata tra l'utente ed il *gateway*.
3. U si è autenticato.
4. U viene autorizzato.

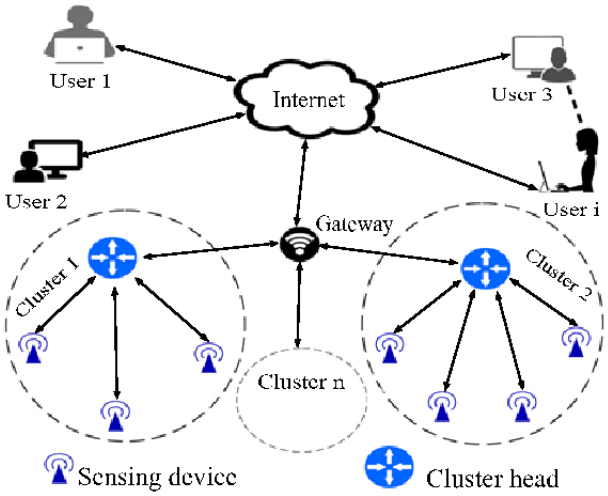


Figura 1.1: Esempio di rete IoT Generalizzato

Nelle reti IoT si ha contemporaneamente necessità di centralizzazione, ovvero di avere una componente centrale che diriga tutta la rete e di decentralizzazione, ovvero avere componenti locali che partecipano al lavoro e forniscono risorse al momento del bisogno.

1.2 CLOUD COMPUTING E IOT

Il *Cloud Computing* è una tecnologia che consente di sfruttare la rete internet per distribuire risorse software e hardware da remoto, si basa sue due entità: fornitore e cliente.

- **Fornitore:** Compagnia che fornisce il servizio, oltre a risorse come database, spazio di archiviazione e software su richiesta.
- **Cliente:** Azienda che acquista dal Fornitore le risorse e le utilizza.

In commercio si trovano servizi cloud di varia tipologia:

- **SAAS** (software as a service): il fornitore fornisce al cliente uno o più programmi installati nel server remoto. (Alcuni esempi sono Google Apps, Salesforce, Dropbox, MailChimp.)
- **DAAS** (data as a service): con questo servizio vengono messi a disposizione via web solamente i dati generalmente attraverso API HTTP. (Alcuni esempi sono Google Maps, Google Translate API or AccuWeather.)

- **HAAS** (hardware as a service): con questo servizio l'utente invia dati ad un server, dati che vengono e messi a disposizione e restituiti all'utente iniziale.
- **PAAS** (platform as a service): Invece che uno o più programmi singoli, viene eseguita in remoto una piattaforma software che può essere costituita da diversi servizi, programmi, librerie, ecc. (Alcuni esempi sono Heroku, Windows Azure.)
- **IAAS** (infrastructure as a service): oltre alle risorse virtuali in remoto, vengono messe a disposizione anche risorse hardware, quali server, capacità di rete, sistemi di memoria e archivio. (Alcuni esempi sono AWS EC2, Rackspace, Google Compute Engine).

Le soluzioni Cloud si adattano bene all'Internet of Things in quanto forniscono un server centrale in cui svolgere calcoli e gestire operazioni più "pesanti" che sarebbero impossibili da eseguire in dispositivi *low-power*.

I nodi gateway comunicano quindi con il Cloud ricevendo istruzioni da questo. Il Cloud può essere utile anche nella risoluzione del problema dell'autenticazione, un utente o un servizio che desidera autenticarsi può richiedere di essere autenticato al Cloud attraverso una tecnologia di autenticazione e servizi di *"Identity Providing"* (un servizio che crea, mantiene e gestisce le informazioni sull'identità degli utenti).

Un esempio di autenticazione di un dispositivo fatta attraverso il Cloud possiamo trovarlo nell'architettura *Cloud IoT Core* di Google [6]. In quest architettura si usa un token *JWT* ed un sistema di cerificati per verificare *in Cloud* che una sessione <*chiavePubblica,chiavePrivata*> appartenga ad un dispositivo legittimo. Anche le autorizzazioni vengono gestite in Cloud attraverso l'utilizzo dell' IAM (Cloud Identity and Access Management).

Il Cloud sebbene fornisca un notevole supporto all'ecosistema IoT presenta alcuni svantaggi, non vi è garanzia di uptime assoluto, i dati potrebbero essere tracciati e potrebbero avvenire attacchi mirati alle strutture di mantenimento dei dati.

1.3 FOG COMPUTING E IOT

La tecnologia FoG è una tecnologia che affianca il Cloud nelle reti di dispositivi IoT come livello intermedio tra i sensori ed il Cloud. Questa strategia punta a risolvere i problemi dovuti alla distanza, cercando di eseguire in locale parte dell'elaborazione di dati costruendo quindi una

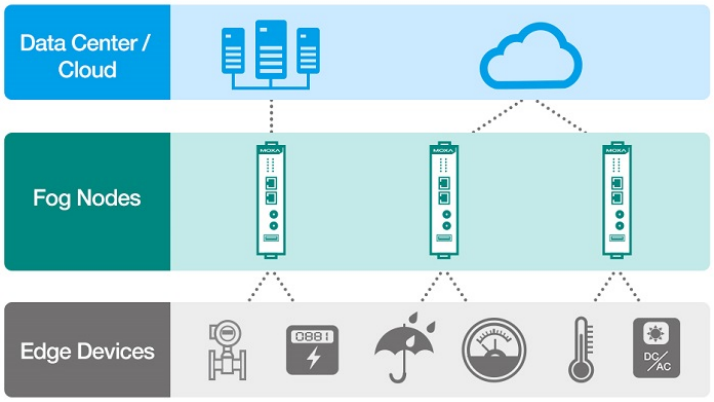


Figura 1.2: Descrizione dell'ambiente FoG e Cloud

rete distribuita che colleghi il Cloud e la rete di dispositivi. Con strutture a tre livelli di questo tipo possiamo ottenere un tempo di latenza molto basso. Il Cloud in quest'ottica diviene un dispositivo principalmente di *storing* (archiviazione) a cui l'architettura FoG invierà i dati più rilevanti, già elaborati e da conservare.

2

METODOLOGIE DI AUTENTICAZIONE ED AUTORIZZAZIONE

In questo capitolo verrà descritto nel dettaglio il problema dell'autenticazione e dell'autorizzazione. Verranno analizzati alcuni standard e tecnologie abilitanti che operano a livello di rete ed a livello applicazione. Queste verranno riprese ed implementate nelle soluzioni "concrete" che verranno analizzate in seguito nella trattazione.

2.1 PROBLEMA DELL'AUTENTICAZIONE ED AUTORIZZAZIONE

In primo luogo è importante definire e distinguere i termini "Autenticazione" ed "Autorizzazione".

1. **Autenticazione:** Con autenticazione si intende convalidare le credenziali di un utente che desidera entrare nella rete. L'autenticazione permette al sistema di confermare l'identità di un utente. Una procedura di autenticazione viene definita *multi-livello* quando vengono utilizzati più passi di autenticazione per convalidare un identità. Si definisce *multi-fattore* se si utilizzano più modi per confermare un identità o confermare la veridicità del processo. Un esempio di autenticazione *multi-fattore* si può riscontrare nell'utilizzo di un codice OTP dopo l'inserimento di username e password.

2. **Autorizzazione:** L'autorizzazione è uno step successivo all'autenticazione. In questa fase viene gestito l'accesso alle risorse da parte dell'utente. L'utente A si è autenticato nel sistema ed è stato riconosciuto come utente A, si determina cosa A può fare o meno dentro il sistema.
   Le informazioni vengono gestite secondo un criterio di accessibilità (esempio matrici ACM/ACL).

	File A	File B	File C	Printer 1
Alice	RW	RW	RW	OK
Bob	R	R	RW	OK
Carol	RW			
David			RW	OK
Faculty	RW		RW	OK

Figura 2.1: Esempio di matrice del controllo degli accessi, un metodo "classico" per la gestione delle autorizzazioni. Si può notare l'associazione tra gli utenti (autenticati) e più permessi tra cui l'accesso a dei files e ad un dispositivo condiviso.

2.1.1 *Il concetto di "Trust"*

È importante definire un concetto fondamentale da cui dipendono, in un modo o nell'altro, buona parte dei sistemi di sicurezza delle reti. Questo concetto è il concetto di *trust* (fiducia). In una rete si ha la necessità di avere delle entità che siano ritenute affidabili, non manipolabili e funzionanti nel modo corretto. Deve essere mantenuto un livello di trust poiché da questo dipende il funzionamento della rete, alcuni attacchi alle infrastrutture di rete si basano proprio su questo principio. Si identificano due modelli basilari di fiducia che ricorreranno nella trattazione.

1. **Fiducia per terze parti:** Due dispositivi A e B non si ritengono affidabili direttamente ma acquisiscono fiducia l'uno nell'altro attraverso un dispositivo "terzo" che entrambi ritengono affidabile. Questo modello è importante per reti su larga scala. Generalmente La fiducia viene acquisita attraverso l'utilizzo di primitive crittografiche, certificati ed autorità certificanti (CA).

2. **Fiducia diretta:** Nel modello a fiducia diretta due individui ripongono fiducia l'uno nell'altro senza l'utilizzo di terzi. Diviene necessario eseguire una mutua identificazione e lo stabilimento del livello di trust prima dello scambio di informazioni.

La gestione dell'autenticazione ed autorizzazione degli utenti può essere gestita in diversi modi. Si possono distinguere architetture *centralizzate* e *distribuite.*

Nelle architetture centralizzate si ha un elemento centrale *trusted* che verifica le richieste di accesso dei vari dispositivi, compie le decisioni, mantiene i dati e le *policy* da applicare.

Nelle architetture decentralizzate la verifica delle richieste di accesso viene eseguita nei nodi foglia, in questo caso ogni elemento della rete mantiene un suo livello di *trust.*

La maggior parte delle metodologie di autenticazione si basa sul controllo dell'*identità*. L'identità è un insieme di dati che descrivono univocamente un dispositivo. Tratteremo alcune delle principali tecnologie e standard che hanno importanza centrale per il problema dell'autenticazione, queste verranno infatti riprese e adattate nei vari protocolli che saranno studiati approfonditamente nei prossimi capitoli.

Prendendo come riferimento il modello OSI è possibile suddividere queste tecnologie/standard in base al livello in cui operano. Nello specifico quasi tutti i metodi di autenticazione vengono effettuati a livello di rete e/o a livello applicazione.

2.2 AUTENTICAZIONE A LIVELLO DI RETE

Nella gestione del problema dell'autenticazione a livello di rete si ha uno standard importante che è lo standard *IEEE 802.1.X.*

2.2.1 *802.1.X*

IEEE 802.1.X è uno standard per il controllo degli accessi alla rete "port based" che fornisce un metodo di autenticazione adattabile a reti di varia tipologia, da reti LAN a reti wireless. Questo standard utilizza EAP (Extensible Authentication Protocol) per il trasferimento ed incapsulamento delle informazioni relative all'autenticazione tra un nodo che desidera entrare nella rete (detto *Supplicant*) e un nodo Autenticatore. EAP è un protocollo per l'autenticazione che supporta sia metodi basati su password che su certificati digitali. Si identificano tre partecipanti al processo:

1. **Supplicant:** E' il dispositivo che vuole entrare nella rete.
2. **Autenticatore:** E il nodo che gestisce l'accesso, ad esempio un Access Point.
3. **Server di autenticazione:** E' un server che contiene un database di autenticazione che tiene traccia degli utenti e dei vari permessi.

L'autenticazione è composta dalle seguenti fasi:

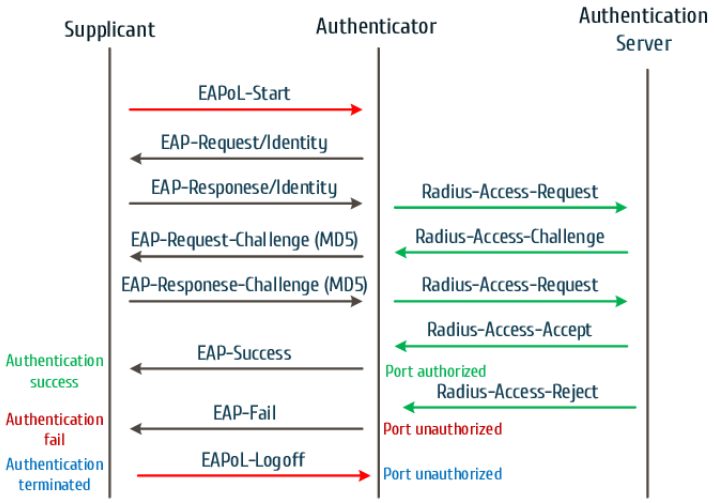


Figura 2.2: Schema di autenticazione previsto da IEEE 802.1.X

1. **Initialization:** Quando viene rilevato un nuovo *Supplicant*, la porta dell'Autenticatore è attiva e impostata sullo stato "non autorizzato".

2. **Initiation:** Per avviare la fase di autenticazione, l'Autenticatore trasmette ad intervalli regolari dei frame "EAP-Request Identity". Il *Supplicant* rimane in ascolto e in risposta invia un "EAP-Response Identity", contenente un User ID. L'Autenticatore in seguito lo inoltra al server di autenticazione.

3. **Negotiation**: Il server di autenticazione riceve le informazioni dall'Autenticatore, le elabora e risponde con una "EAP Request" che a sua volta verrà inviata al *Supplicant* per negoziare un metodo EAP da utilizzare nella comunicazione tra *Supplicant* ed Autenticatore.

4. **Authentication:** Se il server di autenticazione e il *Supplicant* riescono a concordare uno specifico metodo EAP, comincia uno scambio di messaggi suddiviso in più round finché il server di autenticazione risponderà con un messaggio "EAP-Success" o con un messaggio "EAP-Failure". Se l'autenticazione ha successo, l'Autenticatore imposta la porta sullo stato "authorized" permettendo il traffico; se l'autenticazione non ha successo, la porta rimane sullo stato "unauthorized".

802.1.X è importante poiché fornisce una soluzione al problema dell'autenticazione scalabile che verrà ripreso in molte delle tecnologie di autenticazione che operano a livello di rete, dallo standard 5g alle reti WiFi.

2.2.2 *Protocollo EAP*

EAP (Extensible Authentication Protocol) ha un importanza centrale nella trattazione poiché molte tecnologie di autenticazione a livello di rete lo adottano o ne implementano alcune parti. Questo protocollo è definito per i casi in cui non è possibile usufruire del protocollo IP (ad esempio perché l'autenticazione è precedente all'acquisizione degli indirizzi IP).

L'autenticazione EAP può essere one way o mutua ed avviene tra due entità, un client (EAP peer) ed un server (EAP server). Un aspetto fondamentale di questo protocollo è il supporto dei *metodi*.

Con "metodi" si intendono le varie tecnologie con cui si esegue l'identificazione del supplicant. Tra i metodi principali si ricordano:

- **EAP-MD5:** Metodo basato sulla funzione hash MD5 che fornisce un autenticazione one way fondata su userID e password.
- **EAP-TLS:** Metodo basato su TLS che supporta mutua autenticazione tramite certificati, è un metodo molto sicuro ma che richiede più round-trip.
- **EAP-AKA:** Metodo che fornisce un autenticazione mutua basata anch'essa su userID e password utilizzato principalmente in reti cellulari.

Il flusso di lavoro del protocollo EAP è di tipo *lock step* e basato su 4 tipi di messaggi: EAP-request, EAP-response, EAP-success, EAP-failure.

Ad ogni richiesta del server deve corrispondere una risposta del client. Il flusso di lavoro si sviluppa nelle seguenti fasi:

1. Il server invia al client una richiesta di tipo T, solitamente la prima richiesta che il server chiede al client è di tipo "Request Identity".
2. il client invia al server la risposta alla precedente richiesta concorde al tipo T.
3. Client e server continuano a scambiarsi messaggi di tipo EAP Request/EAP Response.
4. Dopo un certo numero di *rounds* il server restituisce EAP-Success o EAP-Failure in caso di successo o fallimento del processo di autenticazione.

EAP è un sistema di autenticazione largamente utilizzato, tra i vantaggi dell'uso di questo sistema si ricordano:

- Supporto a molteplici tecnologie di autenticazione, si possono decidere quali tecnologie implementare sulla base delle esigenze e dal livello di *security* richiesto dalle circostanze.
- Possibilità di separare l'autenticatore con il server di autenticazione che gestisce gli utenti e le policy.

EAP è versatile e può soddisfare esigenze diverse.

Varianti e implementazioni sono ricorrenti nelle tecnologie di autenticazione a livello di rete in varie realtà, ad esempio come analizzato in seguito le tecnologie più recenti per le reti cellulari lo implementano.

2.3 AUTENTICAZIONE A LIVELLO APPLICAZIONE

In questa sezione verranno analizzate le tecnologie di base e gli standard per la trattazione del problema dell'autenticazione ed autorizzazione a livello Applicazione.

2.3.1 *REST*

REST, (REpresentational State Transfer) è un termine coniato da Roy Fielding nella sua tesi di dottorato che definisce un architettura per lo scambio di informazioni in sistemi distribuiti. L'architettura REST prevede entità di tipo client e server che si scambiano risorse.

L'obiettivo è quello di realizzare un infrastruttura di gestione dei dati distribuita che sfrutti alcune semplici funzioni di accesso e interazione come quelle di HTTP: PUT, POST, GET, DELETE. Un sistema che adotta un architettura di questo tipo prende il nome di RESTFul. In queste architetture possono essere utilizzati vari protocolli in tra cui HTTP e CoAP (per dispositivi IoT).

REST è importante poiché fornisce una visione architetturale di tipo *Request/Response* basata su risorse che viene ripresa in molte implementazioni reali.

REST permette di ottenere una separazione tra client e server. È una soluzione facilmente scalabile ed essendo uno standard architetturale non dipende da linguaggi e tecnologie specifiche.

Alcuni sistemi di autenticazione di livello applicazione che verranno analizzati saranno basati su architetture RESTful. Generalmente in questi sistemi si ha un *client* (servizio) che deve essere autenticato ed autorizzato ed un *server* che gestisce la richiesta, questo tipo di richieste si risolve

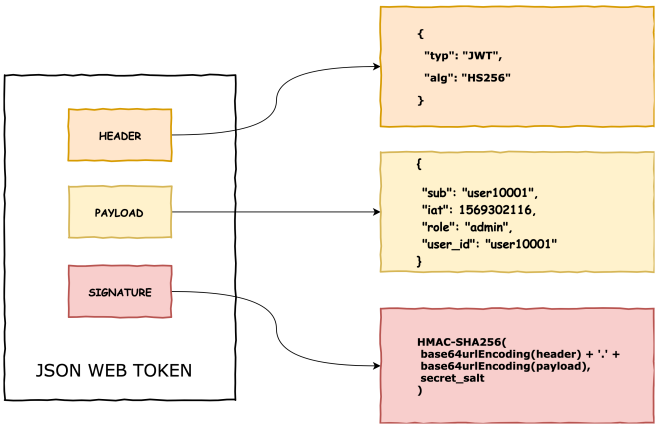


Figura 2.3: Struttura di un Token JWT suddiviso nei tre componenti.

attraverso la generazione e l'utilizzo di token. Un esempio sono i *Token JWT.*

2.3.2 *Token JWT*

JWT è uno standard per la trasmissione di informazioni (in modo sicuro) tra diverse parti attraverso l'utilizzo di file JSON. L'informazione contenuta all'interno del file prende il nome di *Token JWT* e viene firmata in modo da certificarne il contenuto. Un Token è composto da tre parti: un Header, un Payload ed una Signature. L'Header è composto da due campi: un campo `typ` che ne identifica il tipo ed un campo `alg` che riporta l'algoritmo usato per effettuare la firma, solitamente HMAC SHA256 o RSA. All'interno del payload si trovano le asserzioni relative all'utente che sta effettuando la richiesta. Nella generazione del token Il payload e l'Header vengono codificati in BASE64,concatenati e firmati nel seguente modo [21]:

```
firma = SA(secret + b64(header) + '.' + b64(payload))
Token = b64(Header) + "." + b64(payload) + b64(firma)
```

Dove:

- b64 : Funzione che codifica l'argomento in base64.
- SA : *"SignatureAlgorithm"* identifica l'algoritmo di firma da usare specificato nel campo "alg" dell'header (HMAC o RSA).
- + : Operazione di concatenazione tra stringhe.

Al momento della verifica gli interessati controllano il contenuto del Token e la corrispondenza della firma contenuta e quella da loro calcolata usando la loro copia del segreto. L'utilizzo di Token JWT è ricorrente nei sistemi di autenticazione ed autorizzazione o scambio di informazioni a livello applicazione. I vantaggi nell'utilizzo di questi Token ricade sulla loro semplicità, JSON è meno verboso di XML, i Token vengono firmati simmetricamente e molti linguaggi interagiscono nativamente con JSON attraverso primitive di serializzazione.

HMAC

HMAC (keyed-hash message authentication code)[14] è una tecnica per l'autenticazione di messaggi basata su funzione hash. Tramite HMAC è possibile garantire l'integrità e l'autenticità di un messaggio. Si utilizza una combinazione del messaggio originale e una chiave segreta per la generazione. Dato un messaggio m, una lunghezza di blocco B, una funzione crittografica H e un segreto (chiave segreta) K si divide m in blocchi di dimensione B, dopodiché si definisce $\mathrm{HMAC}(K, M)$:

$$\mathrm{HMAC}(K, m) = H\big((K' \oplus \mathrm{opad}) \parallel H((K' \oplus \mathrm{ipad}) \parallel m)\big) \tag{2.1}$$

Dove::

- $H(\cdot)$ è una funzione crittografica che opera in modo iterativo su blocchi di B bytes,
- m è il messaggio da crittografare,
- K è la chiave segreta,
- K' è una chiave segreta composta da B-bytes derivata da K,
- ipad è il byte `0x36` ripetuto B volte,
- opad è il byte `0x5C` ripetuto B volte.

2.3.3 *CoAP*

Un dispositivo viene definito "Constrained Device" (dispositivo limitato) se ha a disposizione una bassa potenza di calcolo, di memoria o di risorse [4]. Si definiscono 3 classi di dispositivi "limitati":

- **Classe 0:** Sono i dispositivi più "limitati" degli altri, generalmente sono i nodi più semplici come i sensori che generalmente hanno capacità di comunicazione molto limitate.
- **Classe 1:** Sono dispositivi in grado di comunicare con altri nodi e con nodi server ma generalmente non riescono ad utilizzare protocolli "full stack" come HTTP o TLS.
- **Classe 2:** Sono i dispositivi in grado di supportare la maggior parte dei protocolli usati dal nodo server.

I dispositivi della classe 0 non sono in grado di interagire direttamente con la rete, per questo motivo hanno bisogno di un intermediario come un gateway o un proxy che agisca da interfaccia.

I dispositivi di classe 1 generalmente hanno comunicatività fino a livello IP dello stack ISO/OSI. Per dispositivi di questo tipo sono stati studiati protocolli specializzati ed ottimizzati come CoAP.

CoAP (Constrained Application Protocol) è un protocollo web per il trasferimento di informazioni in reti di dispositivi constrained come le reti di dispositivi IoT [23]. L'obiettivo del protocollo è far interagire i dispositivi limitati con i servizi HTTP. Questo viene fatto re-implementando un sottoinsieme delle funzionalità offerte dall'architettura REST e HTTP ottimizzandole per comunicazioni M2M. La comunicazione tra nodi attraverso CoAP avviene in modo simile al modello client/server dell'architettura REST, nel caso di interazioni M2M viene suggerita un implementazione in cui ogni nodo agisce sia da client che da server [23].

Una richiesta CoAP è equivalente ad una HTTP e viene inviata da un client a un server, il server risponde al client con un codice ed eventualmente una rappresentazione della risorsa richiesta.

A differenza di HTTP, il protocollo CoAP è asincrono ed è di tipo *connectionless*. Nei casi in cui si richieda la proprietà di affidabilità della connessione CoAP permette l'utilizzo di un flag CON (a cui corrisponderà un messaggio ACK). In Figura 2.4 sono state messe a confronto le due richieste del valore "temperatura" effettuate dal client, nell'immagine di sinistra vi è riportata una comunicazione affidabile (presente il flag CON) mentre a destra si può vedere una comunicazione connectionless UDP-Like (presente il flag NON).

In entrambi i casi è possibile notare che le richieste vengono fatte attraverso comandi HTTP (reimplementati in CoAP) come GET.

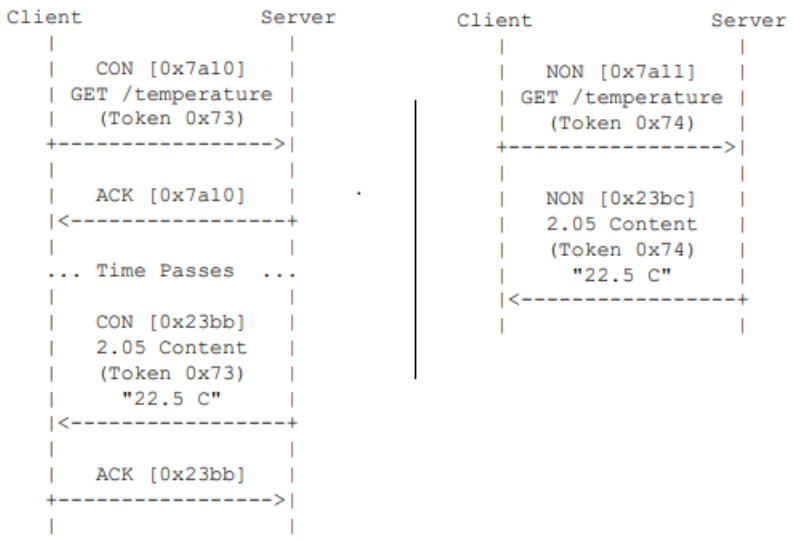


Figura 2.4: Flusso delle richieste in CoAP

2.3.4 *MQTT*

MQTT (Message Queuing Telemetry Transport) è un protocollo di livello applicazione leggero per lo scambio di messaggi di tipo *push protocol* basato sul modello pubblicazione/sottoscrizione che si adatta bene a reti di dispositivi "limitati".

Il modello a pubblicazione/sottoscrizione prevede due insiemi di dispositivi: mittenti *(publishers)* e destinatari *(subscribers)*. Questi non dialogano direttamente ma attraverso un dispositivo che prende il nome di *broker*. I client non conoscono i possibili destinatari, inviano pertanto i propri messaggi al broker.

I possibili destinatari si rivolgono al broker restando in attesa di messaggi e filtrandoli a seconda del tipo di informazioni che desiderano in quel momento. Il *broker* invia i messaggi ai destinatari che si sono sottoscritti ad un certo tipo di informazioni. In genere, il meccanismo di sottoscrizione consente ai subscriber di precisare nel modo più specifico possibile a quali messaggi sono interessati. Per esempio, un subscriber potrebbe "abbonarsi" solo alla ricezione di messaggi da determinati publisher, oppure aventi certe caratteristiche. Il protocollo MQTT adotta un meccanismo di pubblicazione e sottoscrizione di questo tipo attraverso un apposito message broker. Si garantiscono inoltre 3 livelli di QoS basati sulla "garanzia" che un certo messaggio arrivi al destinatario.

Si decide il livello di QoS da utilizzare sulla base della criticità della

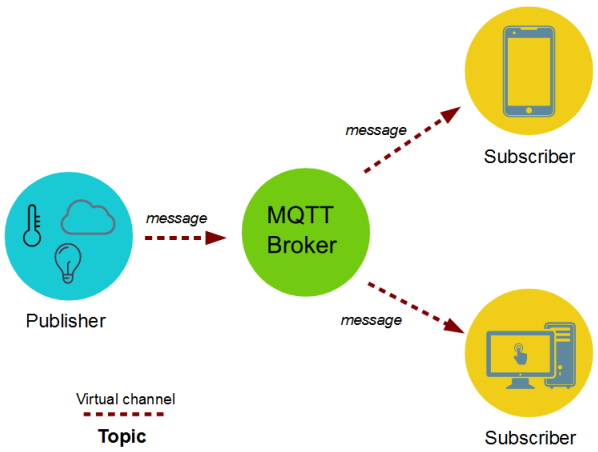


Figura 2.5: architettura MQTT

rete e dei dispositivi in gioco, un QoS alto solitamente genera problemi di *overhead*, il livello più usato è il QoS 1 [18]. Il cuore dell architettura MQTT è il broker, verranno elencati alcune delle soluzioni implementate da diverse compagnie:

- Mosquitto.
- RSBM.
- Hive MQ.
- Verne MA.
- RabbitMQ.
- Mosca.
- EMQX.

MQTT è importante per il mondo IoT poiché è un protocollo leggero che utilizza pacchetti a dimensione ridotta ed è real-time.

3

ANALISI DELLE METODOLOGIE A LIVELLO DI RETE

In questo capitolo verranno analizzate alcune tecniche di autenticazione attualmente esistenti che operano a livello di rete e che riprendono gli standard analizzati in precedenza. Analizzeremo le soluzioni "concrete" al problema dell'autenticazione proposti da Bluetooth Mesh, 5G, LoRaWAN, ZigBee e DIAMETER.

3.1 BLUETOOTH MESH

Bluetooth Mesh è uno standard di rete *mesh* basato su BLE (Bluetooth Low Energy) che consente la comunicazione molti-a-molti. BLE Mesh è un tipo di rete con struttura a livelli che presenta una caratteristica importante dal punto di vista dell'autenticazione, i dispositivi infatti sono autenticati come servizio ed i messaggi vengono inviati in *broadcast*.

3.1.1 *Struttura reti Bluetooth mesh*

I dispositivi prendono il nome di *nodi* e vengono inseriti all'interno della rete attraverso un processo di *provisioning* durante il quale vengono anche autenticati. All'interno della rete si identificano tre macro-categorie di nodi:

1. **Nodi Provisioner:** Sono dispositivi interni alla rete che gestiscono il *provisioning* degli altri dispositivi "esterni alla rete".
2. **Nodi Unprovisioned:** Sono i dispositivi che sono esterni alla rete.
3. **Nodi Provisioned:** Sono i dispositivi che hanno già ottenuto il provisioning.

Tra i nodi *provisioned* cioè quelli interni alla rete se ne identificano tre tipologie particolari [22]:

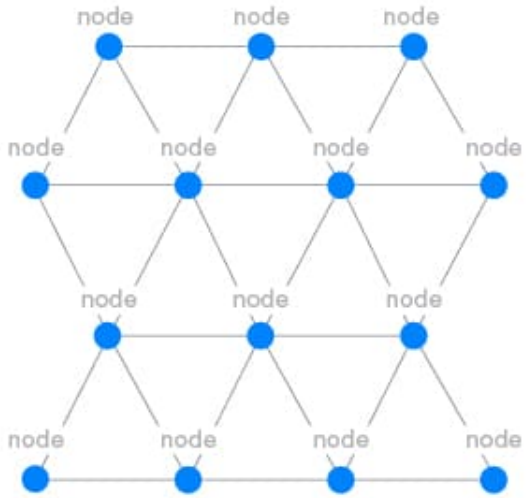


Figura 3.1: Topologia rete bluetooth mesh

1. **Nodo dispositivo:** Dispositivo in grado di comunicare attraverso BLE.
2. **Nodi Low Power:** Nodi che non fanno parte della rete BLE Mesh ma sfruttano solo il BTLE.
3. **Nodi Proxy:** Nodi che servono da interfaccia per altri nodi che non supportano comunicazione Bluetooth.

3.1.2 *Autenticazione in reti Bluetooth Mesh*

Come è stato riportato nella sezione precedente in Bluetooth Mesh l'autenticazione è parte del processo di provisioning, nello specifico è il quarto step. Per completezza riportiamo brevemente i primi tre step del processo di *provisioning*:

1. **Beaconing:** il dispositivo non inserito nella rete comunica al provisioner che è pronto ad entrare.
2. **Invitation:** il provisioner invita il dispositivo ad inviare le sue informazioni e modalità di provisioning.
3. **Scambio di chiavi:** In questa fase il provisioner sceglie uno tra quattro metodi di autenticazione supportati, il dispositivo non *provisioned* e il *provisioner* utilizzano primitive di crittografia a curva ellittica

per derivare una chiave simmetrica *ECDHSecret* che utilizzeranno per avviare una sessione di comunicazione sicura.

Segue quindi il processo di autenticazione, Bluetooth Mesh supporta quattro schemi basati su sistema *Out Of Band (OOB)*. Questi metodi sono: Output OOB, Input OOB, Static OOB, No OOB.

Output OOB

In questo metodo di autenticazione il dispositivo *unprovisioned* genera un numero casuale e lo mostra visivamente all'utente che provvederà a comunicare il numero generato al *provisioner*. Il *provisioner* esegui quindi la verifica ed il processo di autenticazione si conclude.

Input OOB

Questo metodo di autenticazione funziona come Output OOB in cui i ruoli sono scambiati, è infatti il *provisioner* a generare il numero casuale ed il dispositivo *unprovisioned* esegue il check.

Static OOB e No OOB

Static OOB si basa sul verificare l'uguaglianza delle chiavi che si scambiano due dispositivi. Un nodo *provisioner* ed un nodo *unprovisioned* si scambiano (la stessa) chiave k e verificano che questa coincida con una chiave comune (statica) detta *Static OOB Key.* In No-OOB non abbiamo un meccanismo di autenticazione.

Per i metodi di autenticazione basati su OOB si ha uno scambio e convalida di un numero n generato casualmente. Più precisamente se consideriamo un utente *unprovisioned* D che vuole autenticarsi nella rete ed un *provisioner* P. Dopo che avviene lo scambio di n si ha una una fase di verifica in 2 passaggi. Si generano delle chiavi di verifica z, z_1 e poi si procede alla verifica effettiva, la generazione delle chiavi avviene nei seguenti passi: [12]

1. P genera una chiave z usando la funzione AES-CMAC con in input la concatenazione dei valori scambiati con D nelle fasi di provisioning unito al numero random.

```
z = AES-CMAC_CK(n || AuthValue)
```

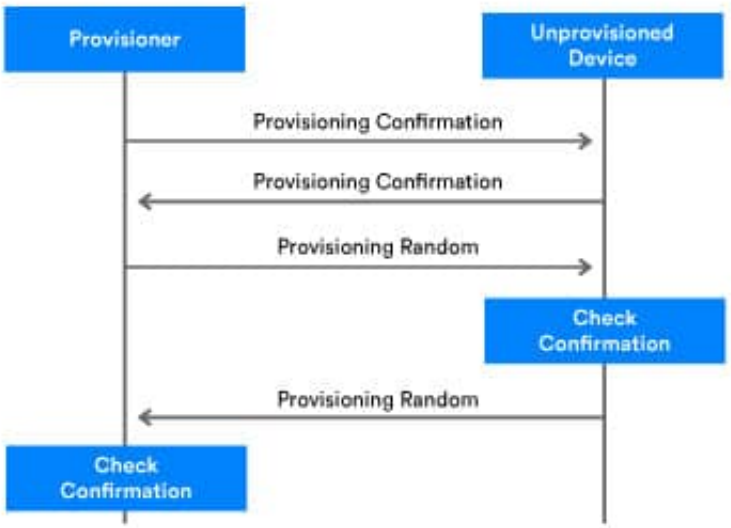


Figura 3.2: Verifica dei valori di conferma in Bluetooth Mesh

in cui

```
CK = k1(ECDHSecret, ConfirmationSalt, 'prck')
```

il campo ConfirmationSalt è dato dalla concatenazione dei campi ProvisioningInvitePDUValue, ProvisioningCapabilitiesPDUValue, ProvisioningStartPDUValue, PublicKeyProvisioner, PublicKeyDevice. La funzione k1 sopracitata è una funzione specificata da bluetooth mesh.

2. P genera una chiave z_1 usando AES-CMAC con i valori scambiati con D nelle fasi di provisioning precedenti (incluso il numero random generato prima).

```
z1 = AES-CMAC_CK(n || AuthValue)
```

3. P e D si scambiano z e z_1.

I due dispositivi devono quindi verificare z e z_1, se z_1 non corrisponde al campo ConfirmationDevice allora si interrompe il processo, altrimenti D invia il suo campo RandomDevice a P. Se RandomDevice soddisfa una nuova conferma crittografica allora l'autenticazione ha avuto successo.

Decommissioning

Con *Decommissioning* si intende il processo di "smantellamento" di un dispositivo, ovvero l'insieme di operazioni da eseguire quando un certo dispositivo deve essere buttato. È un problema importante poichè un dispositivo da "smantellare" può mantenere traccia delle chiavi e dei parametri di accesso alla rete. Se il dispositivo viene recuperato potrebbe contenere informazioni utili ad eventuali attacker. In BLE Mesh si propone una soluzione simile a quella adottata da ZigBee basata sulla rotazione delle chiavi. Si impone un aggiornamento delle chiavi di rete dopo aver inserito in *blacklist* il dispositivo da escludere che pertanto non riceverà l' *update*.

3.2 5G

Il termine *5G* è l'abbreviazione di reti mobili di quinta generazione, rispetto alle generazioni precedenti introduce delle nuove tecnologie tra cui *smart cells*, tecnologie WLAN, nuova struttura del RAN, SDNs, Massive MIMO/3DMIMO e ottimizzazioni per la gestione dei *big data* che lo rendono molto più veloce rispetto alle generazioni precedenti. Supporta un numero maggiore di dispositivi connessi, ha una latenza più bassa ed un affidabilità maggiore.

Queste caratteristiche collimano bene con le necessità di una rete di dispositivi densa di comunicazioni. il 5G porta grandi novità anche per le comunicazioni M2M. Prima di introdurre il problema dell'autenticazione nelle reti 5G e analizzare la soluzione proposta, sarà descritta brevemente l'architettura della rete introducendo il concetto chiave di *Slicing*.

3.2.1 *Struttura reti 5G*

Nelle reti LTE è presente una divisione tra componenti "di trasporto" (canale RAN) e componenti di amministrazione della rete (Core Network). 5G introduce il concetto di *network slicing*: si effettua una divisione della rete in "sezioni" e si assegna ogni "sezione" della rete a classi di dispositivi diversi che hanno esigenze di traffico e di comunicazione diverse.

Nel 5G vengono introdotte le NVF (Network Virtualized Functions), con questo termine intendiamo le tecnologie per virtualizzare classi di funzioni dei nodi di rete su una o più macchine virtuali. In una rete 5G queste funzioni sono virtualizzate nel cloud.

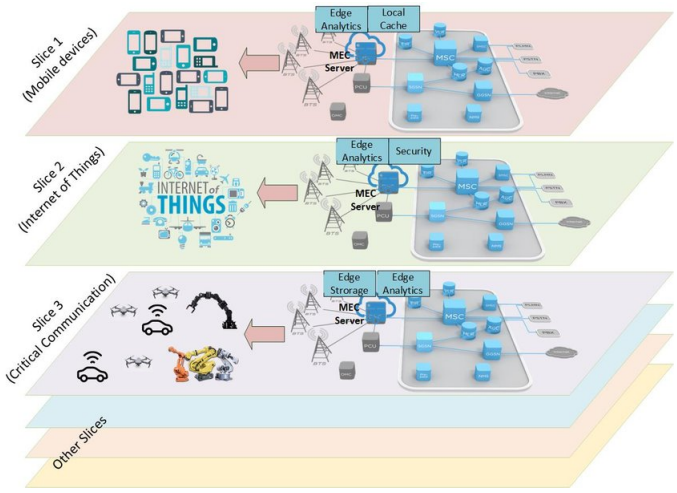


Figura 3.3: Slicing nelle reti 5G, si possono notare più *slices*, uno per i dispositivi mobili, uno per l'IoT, uno per comunicazioni critiche ed altri.

3.2.2 *Autenticazione in reti 5G*

Nell'ambiente 5G abbiamo un processo di autenticazione basato sull'identità del dispositivo (IMSI/SUPI). Si utilizza una complessa architettura che sfrutta funzioni di rete virtualizzate (NVF) per portare a compimento il processo di autenticazione. Entrano in gioco le seguenti parti:

- **UE (user equipment)**: È il dispositivo che intende autenticarsi, comunicando attraverso un canale radio con la rete 5G. Questo dispositivo può essere visualizzato come l'utente finale ed è composto da:
  1. ME: terminale fisico (esempio il cellulare).
  2. USIM: Circuito che attraverso il codice IMSI identifica il dispositivo e l'operatore.

     `MSI=(MCC,MNC,MSIN)`

     Dove
     - MCC (Mobile Country Code) : Identifica il codice del paese, ad esempio all'Italia è associato il codice "222".
     - MNC (Mobile Network Code) : Identifica l'operatore in modo univico.

  - MSIN (Mobile Subscription Identification Number) : Numero a 10 cifre che permette di identificare un dispositivo all'interno della rete.

- **Serving Network:** E' la parte della rete che contiene i nodi di comunicazione e gestione del canale radio, qui troviamo una prima funzione virtualizzata molto importante per il processo di autenticazione: la funzione SEAF (SEcurity Anchor Function). Questa funzione agisce da intermediario per l'autenticazione tra il dispositivo e i server della Home Newtork.

- **Home Network:** È la parte più interna della rete, composta da nodi di controllo e molte funzioni di rete virtualizzate, sono riportate le principali presenti nelle procedure di autenticazione dei paragrafi successivi.

  1. AUSF: funzione della home network che esegue l'autenticazione con l'UE che esegue i calcoli relativi all'autenticazione (es. esegue il key pairing quando si usa 5G-AKA/EAP-AKA).
  2. UDM: entità che si occupa della gestione dei dati, contiene alcune funzioni come ARPF.
  3. SIDF: nodo che ricava l'identità a lungo termine del device che entra nella rete, altra funzione è la SUPI che determina l'identità permanente del device.
  4. AMF: Access and Mobility Management Function è il compoente che riceve tutte le connessioni e le sessioni aperte con l'UE.
  5. ARPF: Funzione di UDM che decide la metodologia di autenticazione sulla base dell'identità del dispositivo.

Nelle reti 5G abbiamo a disposizione un *framework* che supporta tre metodologie di autenticazione: *5G-AKA, EAP-AKA', EAP-TLS*. Queste metodologie vengono scelte sulla base dell'operatore e sulle configurazioni, per esempio le reti Europee dovrebbero supportare sia EAP-AKA'[17] che 5G-AKA.

Protocollo di autenticazione 5G-AKA

5G-Authentication and Key Agreement (Protocol) è l'implementazione su reti 5G del protocollo AKA, un protocollo di sicurezza usato per la mutua autenticazione dei dispositivi mobili a partire dalle reti 3G, proposto

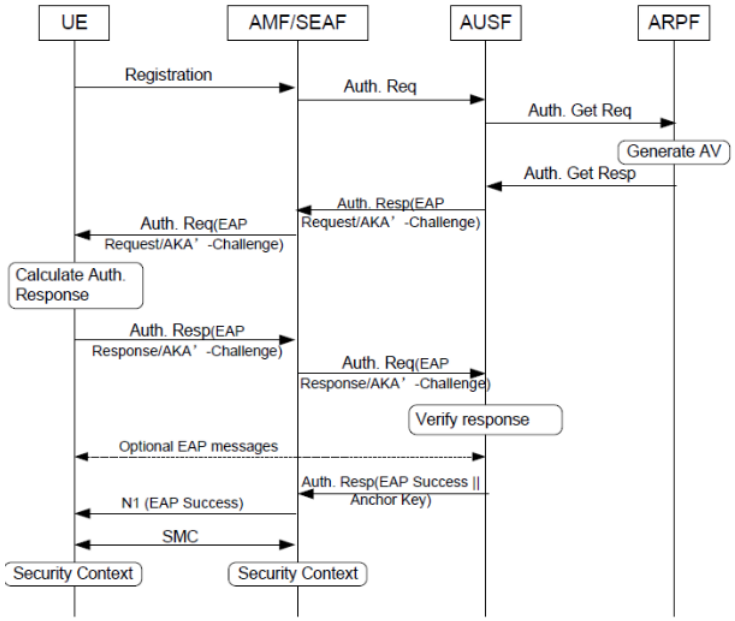


Figura 3.4: Schema di autenticazione 5G-AKA

come successore del precedente protocollo CAVE (Cellular Authentication and Voice Encryption). Possiamo descrivere questo protocollo di autenticazione nelle seguenti fasi:

1. UE richiede a SEAF di entrare nella rete attraverso una richiesta che può essere o un ID temporaneo o un identificatore statico crittografato (a differenza di LTE in cui la richiesta contiene l'identità in chiaro già nella pre-autenticazione).

2. UDM Decide il metodo di autenticazione attraverso la funzione ARPF ed invia alla funzione AUSF il vettore di autenticazione AV che è composto da:

   ```
   V=(AUTH,XRES,KAUSF,SUPI/SUCI)
   ```

   L'AUSF costruisce come nel caso LTE un token HXRES di presunta risposta e immagazina il valore K_{AUSF}, invia quindi AV all' UE.

3. L' UE valuta il campo AUTH di AV e si acquisisce il *trust* della rete.

4. Da adesso l'autenticazione procede dall'utente UE che invia alla funzione SEAF un token di risposta RES. SEAF valuta l'autenticità

di RES e lo inoltra ad AUSF che in quanto unità computazionale esegue la valutazione e prende la decisione finale sull'autenticazione di UE.

5. Se il Token RES non è valido viene rifiutato, altrimenti se è accettato la funzione AUSF computa la chiave K_{SEAF} e la invia a SEAF.

6. SEAF riceve K_{SEAF} e la utilizza per computare K_{AMF} e la invia alla funzione AMF.

7. AMF adesso valuta K_{AMF} generata da SEAF, se è autentica ed integra genera una chiave k_{gNB} che viene inviata al nodo gNB che a sua volta costuirà due chiavi (pubblica e privata) con cui crittografare la sessione con l'UE.

Protocollo di autenticazione EAP-AKA'

In questa sezione analizzeremo un altro metodo di autenticazione in reti 5G anche questo basato su AKA ma combinato con EAP, un protocollo usato nello standard 802.1.X [17]. Le differenze principali con 5G-AKA sono:

- SEAF è trasparente e si limita a fare il "forwarding" dei messaggi tra UE ed AUSF
- La derivazione della chiave K_{AUSF} è eseguita da AUSF stesso attraverso le chiavi ricevute da UDM/ARPF.

Il flusso di lavoro del protocollo può essere riassunto nelle seguenti fasi:

1. AMF/SEAF inizia l'autenticazione, UDM tramite la funzione AUSF decide di utilizzare EAP-AKA.

2. AUSF richiede l'authentication vector *trasformato* a ARPF, la cui struttura è

```
AV' =( XRES,AUTN,CK,IK,RAND);
```

   Dove:

   a) XRES = risposta attesa.
   b) AUTN = token di autenticazione.
   c) RAND = numero generato pseudocasualmente.

Figura 3.5: Entità coinvolte in autentazione eap-aka'

d) CK,IK = chiavi di sessione.

3. ARPF genera il vettore AV'=(XRES,AUTN,CK,IK,RAND); e lo invia all'AUSF.

Il processo di autenticazione da questo momento procede come in 5g-AKA con l'unica differenza che AMF/SEAF non partecipa più attivamente all'autenticazione ma è l'AUSF che compara XRES con RES e decide se ammettere o meno il dispositivo nella rete. La risposta finale viene inviata all'AMF/SEAF che costruiscono quindi le chiavi di sessione in caso di autenticazione avvenuta. La conclusione del processo di autenticazione viene notificata attraverso un messaggio EAP-SUCCESS come specificato nello standard EAP. La differenza principale tra EAP-AKA e EAP-AKA' [2] è l'introduzione di una nuova funzione di derivazione delle chiavi, in generale abbiamo le seguenti chiavi:

1. K_{AUSF} è master key usata dall'UE e dalla rete per derivare le altre chiavi.

2. K_{AMF} è la chiave generate per derivare a sua volta altre chiavi come K_{nasenc} e K_{nasint}, chiavi di protezione per messaggi su protocollo Non-access stratum (protocollo tra UE e Core network) per lo stabilimento delle sessioni.

3. K_{gnB} è la chiave generata per i nodi GNB per la protezione dell'user plane insieme a K_{upint},K_{rrcenc} e K_{rrcint}.

In 5G si utilizzano algoritmi diversi per mantenere la confidenzialità e l'integrità delle informazioni.

- Per la confidenzialità si utilzzano:
  - NEA0: Non si utilizza alcuna crittografia.
  - 128-NEA1: Algoritmo di cifratura SNOW3G.
  - 128-NEA2: Implementazione di AES-128 CTR.
  - 128-NEA3: Cifrario a flusso ZUC.
- Per il mantenimento dell'integrità si utilizzano:

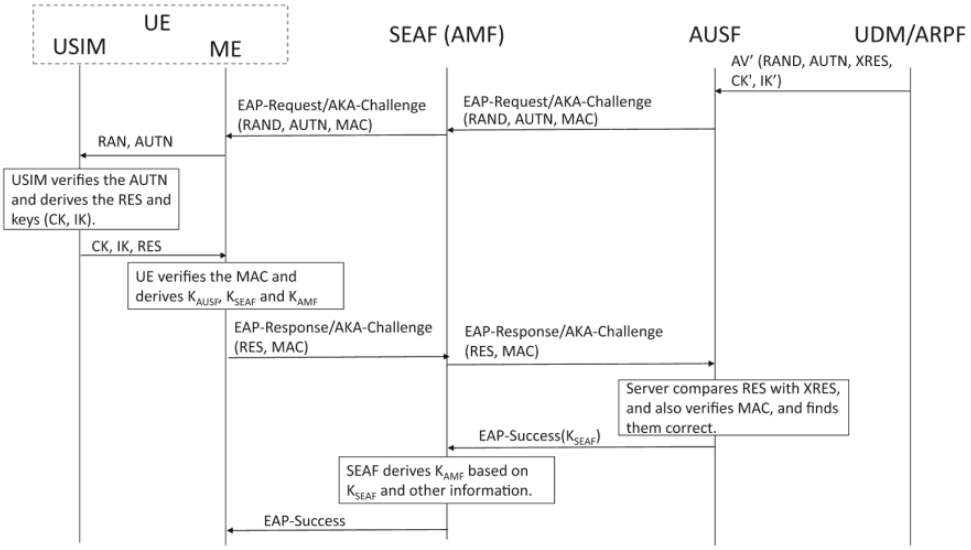


Figura 3.6: Schema di autenticazione EAP-AKA'

- NIA0: Non si utilizza alcuna primitiva per mantenere l'integrità.
- 128-NIA1: Procedura basata su SNOW3G.
- 128-NIA2: Si utilizza AES-128 CMAC.
- 128-NIA3: Procedura basata su cifrario ZUC.

L'utilizzo di 128-NIA3 è opzionale, così come l'utilizzo di primitive di protezione dell'integrità tra l'UE e i nodi della rete.

Decommissioning

In 5G non si riscontra una particolare gestione del problema del *decommissioning* in quanto si non si hanno informazioni sensibili permanentemente registrate nell'UE e l'autenticazione si basa soltanto sull'identità del singolo dispositivo. Questo implica che la "rimozione" di un dispositivo viene fatta semplicemente a livello di matrice di controllo degli accessi. Quando un dispositivo tenta di accedere alla rete con un certo IMSI si verifica la presenza di questo codice nell' HLR, se non lo si trova l'accesso non viene garantito.

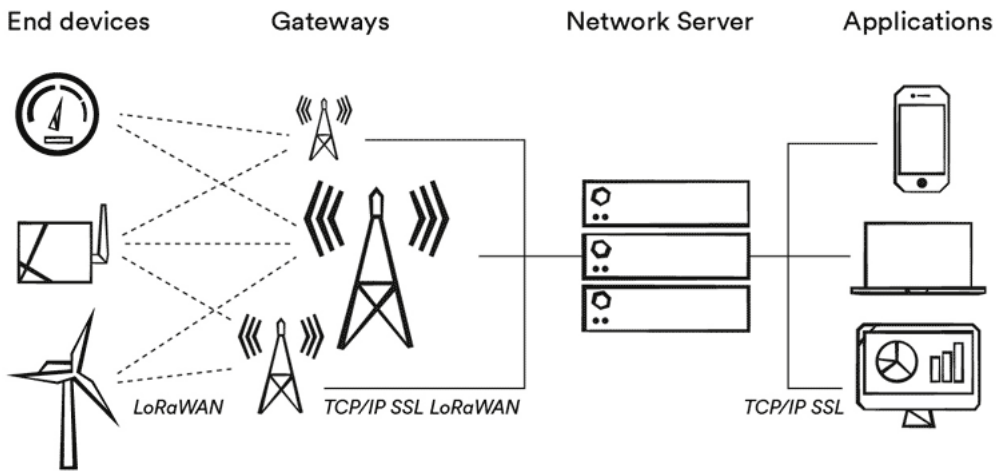


Figura 3.7: Struttura rete LoRaWAN

3.3 LORAWAN

LoRa è una tecnologia di trasmissione dati a lungo raggio che usa le frequenze radio 433 MHz, 868 MHz (Europa) e 915 MHz (Nord America); le reti che si formano prendono il nome di reti LoRaWAN. LoRa si propone come un alternativa allo standard wifi, bluetooth e lan per la comunicazione con data-rate bassi. Questa tecnologia trova spazio nel mondo IoT in quanto permette di coprire distanze di trasmissione molto lunghe (dell'ordine dei 50km), permette di ridurre al minimo i costi in termini di energia, e supporta comunicazioni (sessioni) criptate.

3.3.1 *Struttura reti LoRaWAN*

LoRaWAN definisce il protocollo di comunicazione e l'architettura del sistema per la rete, mentre il livello fisico LoRa permette il collegamento di comunicazione a lungo raggio, la topologia delle reti è tipicamente *star-of-stars*. I Dispositivi della rete funzionano in modo asincrono e trasmettono quando hanno dati disponibili per l'invio. I dati trasmessi da un dispositivo vengono ricevuti da più *gateway* e sono inoltrati a un server centralizzato. In una rete LoRa sono previsti quattro partecipanti:

- **EP**: End points ovvero i dispositivi IoT connessi.
- **LoRa Gateways**: Sono nodi che ricvevono le comunicazioni dagli endpoints e le inviano al Network Server NS.

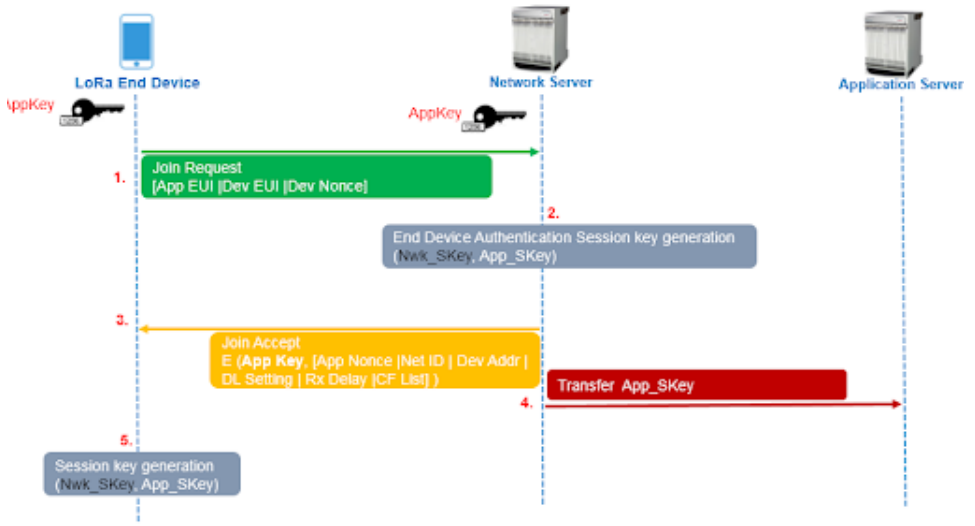


Figura 3.8: Autenticazione OTA in reti LorA

- **Remote Computer**: End user che riceve informazioni dai devices.
- **LoRa NS:** "LoRaWAN Network Server" è il server della rete LoRaWAN che amministra la rete, elimina pacchetti duplicati, esegue scheduling e agisce da moderatore della rete.

Le informazioni trasmesse da un dispositivo sulla rete sono ricevute da tutti i nodi LoRa nelle vicinanze, questo garantisce una maggior resilienza della rete.

3.3.2 *Autenticazione in LoRaWAN*

Nelle reti LoRaWAN si hanno due metodi di autenticazione:

1. **OTA (Over the air Authentication)**: Si utilizza quando il dispositivo EP è già operativo o è stato riconfigurato.
2. **ABP (Activation by personalization)**: Si utilizza in corrispondenza di EP che presentano la possibilità di configurare le chiavi di cifratura direttamente sul dispositivo.

Autenticazione OTA

Possiamo descrivere il funzionamento dell'autenticazione OTA in reti LoRaWAN come segue

1. l'EP comunica al network server una *Join Request* (richiesta)

   ```
   JR=(appEui,devEui,devNonce)
   ```

   Dove appEui, devEui definiscono l'identità del dispositivo mentre il campo devNonce funziona da nonce, ovvero numero intero $n \geqslant 0$ ad utilizzo unico che serve a garantire la "freschezza" della comunicazione e l'impossibilità di replay attack/messaggi duplicati. Il messaggio in questione viene trasmesso in chiaro sulla rete.

2. Il network server riceve la richiesta, verifica il nonce, se supera il controllo si genera un messaggio che prende il nome di *Join Accept* composto dai campi:

   ```
   JA=AppKey( AppNonce, NetID,
   DevAddr, DLSettings, RxDelay)
   ```

   e lo si invia nuovamente al dispositivo EP, stavolta cifrato con la chiave appKey, la chiave appKey è una chiave a 16 byte univoca presente in ogni dispositivo. In questa fase si generano anche una coppia di chiavi di sessione <NwkSKey ,AppSKey> che verranno usate in seguito per generare la sessione di comunicazione cifrata.

3. L'end Device dopo aver ricevuto il *Join Accept* lo decripta usando appKey e usa i campi di JA per costruire le due chiavi di sessione.

Durante la fase di autenticazione la chiave AppSKey generata al punto 2 viene trasmessa anche all'application server, un server che fornisce supporto alla rete, nella struttra Cloud-Fog può essere un nodo FoG. LoRa tuttavia non rilascia indicazioni sull'utilizzo di questa chiave nell'application server.

Autenticazione "by Personalization"

In questo caso non è presente alcuna join procedure , il dispositivo EP ha già le chiavi per generare le sessioni al proprio interno. In queste reti i dispositivi vengono identificati singolarmente attraverso la loro identità, il campo DevEUI che è un un identificatore IEEE EUI-64.

Decommissioning

Il problema del *Decommissioning* in reti di questo tipo si basa su una procedura che prende il nome di "Exit Procedure". Tuttavia lo standard ufficiale non specifica direttive sui comandi da utilizzare, l'unico vincolo è che questa procedura deve partire dal livello Applicazione.

3.4 ZIGBEE

ZigBee è uno standard di comunicazione che riprende lo standard *IEEE 802.15.4* per le reti WPAN, è formato da 4 layer: fisico, mac, rete (NWK), application (APL). Le reti ZigBee possono assumere topologie diverse (star, tree, cluster), possiamo costruire reti distribuite o centralizzate.

3.4.1 *Struttura della rete*

In una rete ZigBee sono presenti tre tipologie di dispositivi:

- **ZigBee Coordinator (ZC)**: Nodo che gestisce la rete, gestisce i Trust Center della rete e può agire esso stesso da trust center. Si occupa di mantenere la lista dei dispositivi associati ed autenticati e di gestire operazioni di join, rejoin e dispositivi orfani.
- **ZigBee Router (ZR)**:nodi che Agiscono come router, hanno lo stesso hardware di ZC.
- **ZigBee End Device (ZED)**: Sono nodi *endpoint*.

3.4.2 *Autenticazione in reti ZigBee*

L'autenticazione nelle reti ZigBee avviene dopo l'associazione (conforme allo standard IEE 801.15.4). I dispositivi entrano nella rete dopodiché si avvia il processo di autenticazione ed autorizzazione. Sono presenti due tipologie di autenticazione, una per nodi di tipo "residenziale" ed una per nodi di tipo "commerciale" [10]. Considerando un utente U che desidera autenticarsi nella rete descriviamo le due tecnologie di autenticazione:

- **Residential mode:**
  1. U verifica se possiede o meno la chiave di rete N_k.

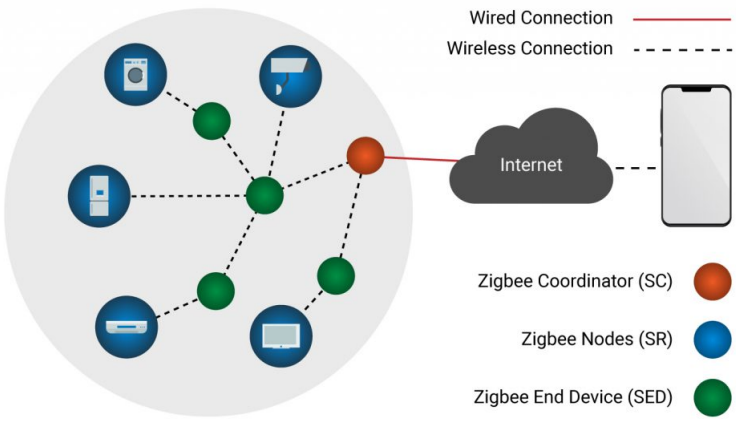


Figura 3.9: Architettura rete ZigBee

2. Se U non possiede N_k la ottiene da ZC attraverso un canale *non protetto*, se la possiede riceve una chiave composta da 0.
3. Ottenuta N_k il dispositivo U si sincronizza con ZC ed entra nella rete.

- **Commercial mode:**
  1. U verifica se possiede o meno la chiave di rete N_k.
  2. Se U non possiede una chiave di rete N_k il coordinatore ZC invia N_k ad U attraverso un canale *protetto*, se U non possiede la master key M_k il coordinatore ZC può inviarla ad U attraverso un canale non protetto.
  3. Nel momento in cui U ha M_k si avvia il protocollo SKKE per eseguire lo scambio di chiavi tra U e ZC. Questo protocollo funziona tra due dispositivi un Iniziatore ed un Target [10]. il protocollo SKKE consente lo scambio di chiavi con il seguente flusso:
     a) L'iniziatore stabilisce una chiave di collegamento L_k basata sulla master key M_k e avvia una comunicazione con il target.
     b) Il target deriva dal messaggio la chiave L_k.
     c) Se entrambi posseggono la stessa L_k si usa L_k come shared key per la comunicazione.

ZigBee introduce un meccanismo di *ingresso* nella rete che permette ad un dispositvo di entrare nella rete sfruttando dispositivi fisicamente vicini. Questo protocollo si basa su *Touchlink Commissioning*. Dato un dispositivo U che vuole entrare nella rete ed un nodo che sia ZC o ZR: [3]

1. ZC/ZR (Iniziatore) concatena i campi TrID e RsID e deriva una chiave di trasporto K_{Trans} quindi si genera una chiave di rete N_k usando K_{Trans}, invia quindi una nuova chiave β ottenuta dalle precedenti al dispositivo U senza ulteriori passi crittografici.
2. U riceve il messaggio, verifica TrID e invia una risposta di conferma.

Segue quindi il processo di autenticazione.

Identificazione e decommissioning

In queste reti i dispositivi vengono identificati attraverso un indirizzo MAC a 64 bit, un EUI a 48 bit e uno short address (il cui uso è ristretto alla join procedure) di 16 bit. Per quanto riguarda invece il decommissioning è gestito soltanto nelle reti ZigBee centralizzate, si prevedono due metodologie:

1. **Remove by command:** si invia un comando al device che rimuove tutti i dati in esso contenuti.
2. **Remove by key update:** si esclude dal gruppo di trust il dispositivo e si aggiornano le chiavi degli altri.

Viene suggerito di utilizzare la metodologia "Remove by key update".

Differenze tra 802.15.4 e ZigBee

802.15.4 è uno standard per reti ad area personale ed efficienti dal punto di vista energetico. ZigBee riprende le direttive di 802.15.4 per il livello fisico e mac e definisce nuovi livelli superiori non previsti nello standard. Questi livello vengono utilizzati per implementare servizi di vario genere all'interno della rete come il servizio di *profiling* dei dispositivi connessi.

3.5 DIAMETER

DIAMETER è un protocollo di scambio dati orientato a sistemi di autenticazione di tipo peer to peer che permette la comunicazione tra un *Authenticator* ed un *Authentication server.* DIAMETER è basato su TCP/IP

e supporta IPSec [5]. I client possono supportare comunicazioni TCP o SCTP (porta 3868) mentre gli agent devono supportarle entrambe. E' possibile anche un implementazione TLS/DTLS (porta 5868) [5]. DIAMETER funziona a livello di trasporto per quanto riguarda lo stabilimento della connessione e l'invio di messaggi mentre le sessioni sono definite a livello applicativo [5]. I messaggi scambiati da DIAMETER prendono il nome di AVPs.

3.5.1 *Autenticazione in DIAMETER*

DIAMETER identitifca i dispositivi attraverso la struttura DiameterIdentity, una formato che raccoglie più campi ed identifica un dispositivo [7]. Per quanto riguarda l'autenticazione questa è di tipo distribuito in quanto a differenza di RADIUS in DIAMETER si possono avere più server di autenticazione che lavorano in modo distribuito rimanendo in architettura 802.1.X. Considerando un dispositivo U che vuole autenticarsi e un dispositivo D interno alla rete abbiamo che: [24]

1. U invia a D un messaggio CER con il campo

   ```
   Auth-Application-Id AVP = 1
   ```

2. D invia ad U un messaggio CEA con il campo

   ```
   Result-Code = DIAMETER-SUCCESS
   ```

   Se l'ingresso nella rete ha avuto successo. Si passa poi alla fase dell'autenticazione, che può avvenire in due modi:

   a) Autenticazione One-Time: U invia quindi una AA-Request (richiesta di autenticazione) e D risponde con una AA-Answer (conferma di autenticazione).

   b) Autenticazione Multi-Round: Si eseguono più round di autenticazioni one-time in cui i due dispositivi di scambiano più informazioni.

L'autenticazione avviene attraverso lo scambio di messaggi AAR/AAA.

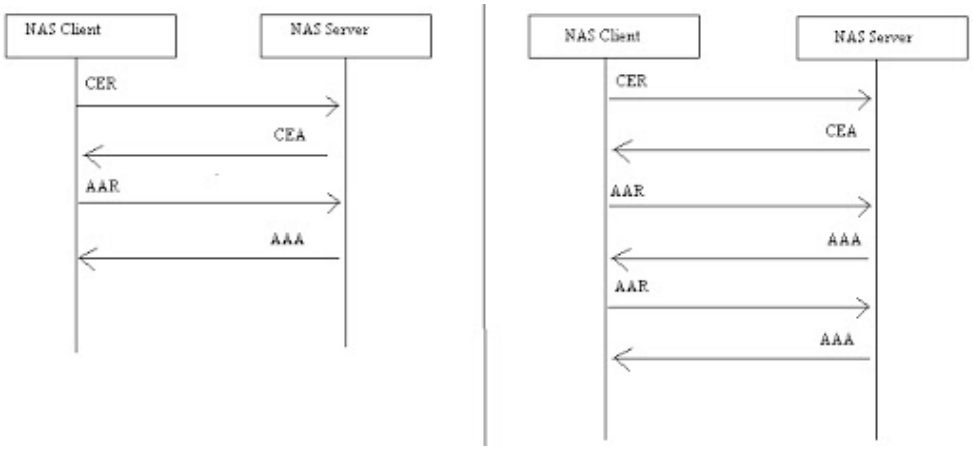


Figura 3.10: Autenticazione one-time e multi-round in Diameter

Autenticazione DIAMETER-EAP

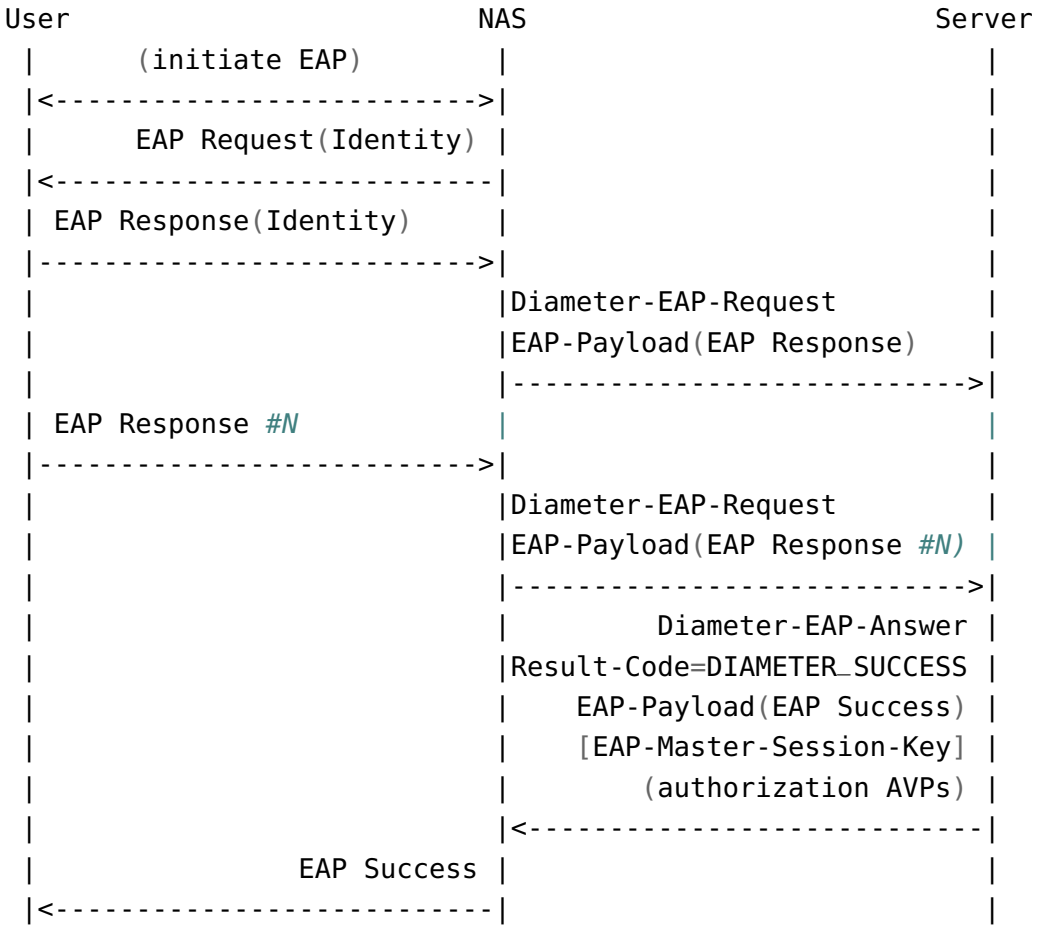
DIAMETER supporta anche l'autenticazione attraverso EAP (come da standard 802.1.X) attraverso un interfaccia basata su NASREQ (Diameter Network Access Server Requirements) [9]. Il suo flusso di lavoro può essere riassunto nei seguenti punti:

1. si avvia EAP attraverso un link come PPP o IEEE 802.11i (WPA2 - Wi-Fi).
2. U invia una richiesta di EAP.
3. D risponde con una conferma Diameter-EAP-Answer, un token che contiene il pacchetto EAP, si avvia quindi il tipo di autenticazione multi round:
   a) il client invia la sua identità al server attraverso un pacchetto segnato come "EAP-RESPONSE".
   b) il client invia il suo Payload EAP.

   Possono essere richiesti più round, al termine della conversazione di ha il pacchetto conclusivo DIAMETER-SUCCESS.

Dopo l'autenticazione segue l'autorizzazione, il client invia una risposta Diameter-EAP-Answer con un AVP (codice identificativo del pacchetto) di autorizzazione. Il server gestisce le autorizzazioni, può fornire subito i permessi richiesti o dividerli in più sotto-autorizzazioni, può anche fornire al client delle chiavi di sessione per avviare un canale cifrato [9].

Segue quindi lo schema di autenticazione DIAMETER-EAP:



4

ANALISI DELLE METODOLOGIE A LIVELLO APPLICAZIONE

In questa sezione verranno analizzate alcune metodologie di autenticazione che operano nell'application layer utilizzando le tecnologie e gli standard menzionati nel capitolo 2. Verranno presentati DCAF, Oauth e Autho.

4.1 DCAF

DCAF (Delegated CoAP Authentication and Authorization Framework) è uno standard di autorizzazione e autenticazione basato su protocollo CoAP che opera a livello applicazione. Questo protocollo ha lo scopo di realizzare un meccanismo di sicurezza dei messaggi e autenticazione/autorizzazione per reti di dispositivi "limitati" che utilizzano CoAP.

In queste circostanze i meccanismi di autenticazione sono difficili da realizzare a causa delle risorse limitate, DCAF descrive un'architettura in cui si introducono dei nuovi nodi nella rete. Questi nodi non "limitati" assumono il ruolo di gestori ed eseguono le operazioni più pesanti per gli altri nodi come la gestione delle chiavi e l'autorizzazione per l' accesso alle risorse.

DCAF gestisce le autorizzazioni di accesso alle risorse attraverso dei token di accesso. Un dispositivo che desidera accedere ad un elemento deve ottenere il permesso dal nodo che gestisce le autorizzazioni [11]. In questo processo le informazioni di autorizzazione e le policy non vengono scambiate ma vengono ricavate in modo implicito. Dal punto di vista della sicurezza dei messaggi si utilizza DTLS, una variante del protocollo TLS ideata/progettata per lavorare con UDP. DCAF è basato su richieste CoAP, si prevede un architettura composta da:

- **Server:** Un nodo che ospita e rappresenta una risorsa.

- **Client:** Un nodo che tenta di accedere a una risorsa sul nodo server.
- **Server Authorization Manager (SAM):** Un'entità che gestisce autenticazione e autorizzazione per un nodo di tipo server.
- **Client Authorization Manager (CAM):** Un'entità gestisce i dati di autenticazione e autorizzazione per un nodo di tipo client.
- **Authentication Manager:** Nodo che esegue il processo di autenticazione per il client.
- **Client Overseeing Principal (COP):** nodo coordinatore che è responsabile dei Client e controlla le autorizzazioni di accesso ad una certa risorsa.
- **Resource Overseeing Principal (ROP):** nodo coordinatore che è responsabile delle risorse e controlla le autorizzazioni di accesso.

Ogni nodo server S ha un nodo SAM che lo autentica e lo autorizza. Un client C che vuole accedere ad una risorsa r_i su un nodo server deve richiedere un token di accesso al SAM che serve S o, se si tratta di un dispositivo "limitato", al nodo CAM. CAM decide se S ha il permesso ad accedere alla risorsa r_i secondo le politiche stabilite da COP, in caso affermativo COP trasmette la richiesta a SAM. Se SAM decide che C è autorizzato ad accedere alla risorsa in base a i criteri impostati da ROP, genera una chiave precondivisa k_p per la comunicazione tra C e S e la utilizza per costruire un token di accesso.

SAM aggiunge le autorizzazioni di accesso al ticket in modo che CAM e S possano interpretarle. CAM verifica se le autorizzazioni nel ticket di accesso sono conformi all'autorizzazione COP per il nodo C e in caso di esito del confronto positivo COP invia il token generato a C. Il processo si conclude nel momento in cui C presenta il token (valido) ad S. Si instaura quindi un canale di comunicazione sicuro tra C ed S. Il sistema DCAF gestisce (opzionalmente) anche delle risorse non autorizzate a cui è sempre possibile accedere [11].

DCAF implementa un framework di autorizzazione e autenticazione su protocollo CoAP ottimizzato per dispositivi "limitati", tuttavia l'autenticazione ed autorizzazione di un servizio attraverso questo sistema è molto complicata e richiede l'inserimento di un numero di partecipanti non indifferente.

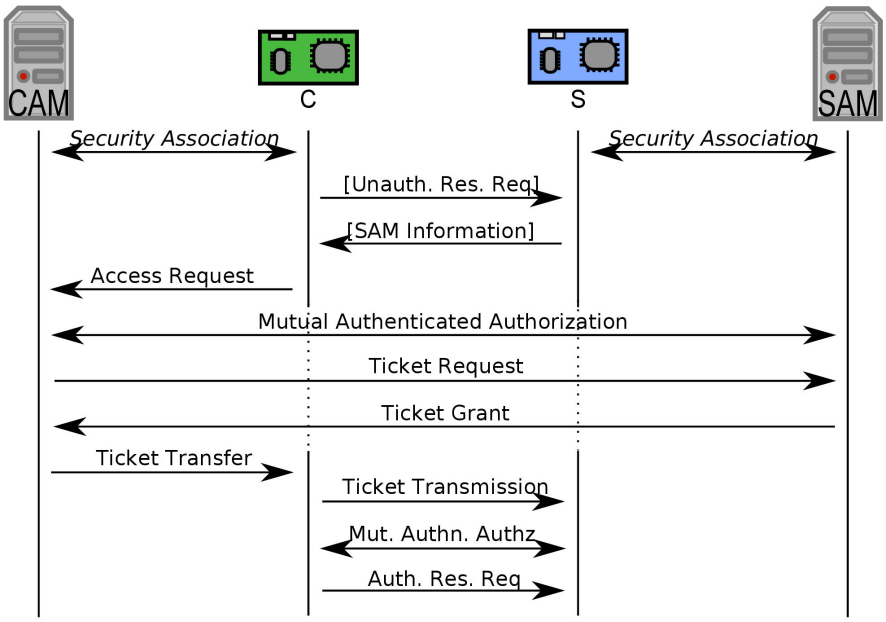


Figura 4.1: Esecuzione di DCAF

4.2 OAUTH 2.0

Oauth 2.0 è uno standard di autorizzazione che permette ad un utente o ad un applicazione di ottenere l'accesso ad un certo servizio HTTP. In Oauth non si utilizza l'accesso attraverso username e password ma utilizzando dei *token* la cui validità viene accertata da dei server di autorizzazione [13].

Un utente ad esempio può richiedere ad un sensore di temperatura di inviargli i propri rilevamenti chiedendo il permesso direttamente ad un server di autorizzazione, il quale in caso di accesso consentito rilascia un token di accesso specifico per quell'utente. Oauth definisce 4 partecipanti:

- **Resource owner:** entità che possiede la risorsa.
- **Resource server:** server che possiede fisicamente la risorsa, è in grado di accettare o rifiutare token di accesso che gli vengono presentati.
- **Client:** utente/entità che desidera accedere ad una certa risorsa.
- **Server di autorizzazione:** Server che emette i token di accesso dopo aver autenticato il resource owner e aver ottenuto l'autorizzazione.

Il server di autorizzazione può essere contemporaneamente anche un resource server e può emettere token di accesso altri. Oauth è un protocollo di *autorizzazione* (basato su deleghe). Per questo motivo non si ha una vera e propria metodologia di autenticazione, sebbene sia possibile costruire un protocollo di autenticazione usando Oauth.

Generalmente si può considerare la ricezione di un token valido come prova *(proof of ownership)* che l'utente si sia effettivamente autenticato, tuttavia la natura stessa dei token, ad esempio il fatto che siano riutilizzabili porta comunque problemi. L'azienda OpenID ha proposto una miglioria di Oauth2 che aggiunge una gestione delle identità ed autenticazione ad Oauth 2.0 che prende il nome di OpenID Connect.

4.2.1 *OpenID Connect*

OpenID Connect è definito come un "identity layer" che viene aggiunto alla struttura di Oauth. Permette ai Client di verificare l'identità di un resource server attraverso un processo di autenticazione che viene eseguito dal server di autorizzazione di Oauth. Il vantaggio principale di Utilizzare OpenID Connect è che permette di utilizzare soltanto Oauth per la gestione del problema dell'autorizzazione ed autenticazione. Un esempio di servizio che implementa Oauth 2.0 e OpenID Connect è il Microsoft Identity Platform.

Descriviamo quindi l'integrazione tra Oauth 2.0 e OpenID Connect nel caso in cui un utente (Client) desideri accedere ad una certa risorsa su un Resource Server.

1. Il client invia una richiesta di autenticazione al server di autorizzazione attraverso OpenID Connect.

2. Il server di autorizzazione autentica il client e ricerca le sue rispettive autorizzazioni. Genera quindi un token di accesso per il client.

3. Il client invia una richiesta al Resource Server.

4. il client ottiene la risorsa richiesta.

L'utilizzo di Oauth + OpenID Connect permette di avere un controllo dell'identità più sicuro, decentralizzato e facile da configurare. Si utilizza tuttavia un servizio di terze parti per la gestione di dati sensibili. Il sistema presentato ha un architettura più semplice di quella proposta da DCAF, si implementa un solo protocollo che contiene già le primitive

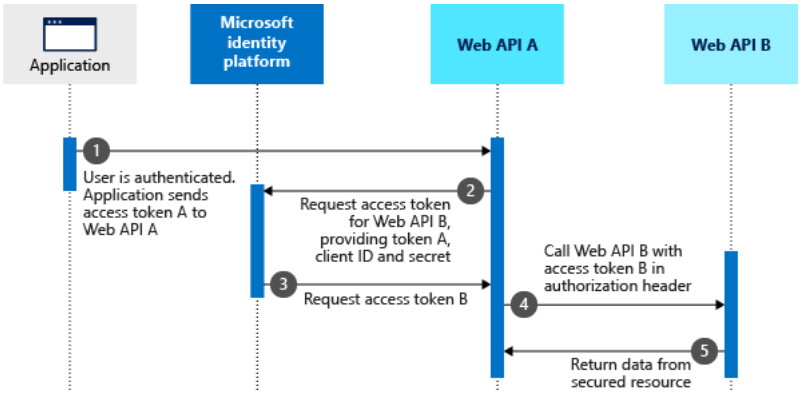


Figura 4.2: Integrazione OpenID Connect ed Oauth 2.0 in Microsoft identity framework

di sicurezza necessarie. Molte compagnie tra cui Microsoft e Google implementano questo sistema.

Il "Sistema Pubblico di Identità Digitale" italiano implementa OpenID Connect su Oauth 2.0 per la gestione di autenticazione ed autorizzazione e gestione degli accessi alle API. [1] È stato scelto di utilizzare l' *International Government Assurance Profile* o IGov 1.0 in cui si precede l'utilizzo di *Authorization code flow*, questo flusso è la soluzione ideale per sessioni lunghe o aggiornabili attraverso l'uso del refresh token. L'Authorization code flow ottiene l'authorization code dall'authorization endpoint dell'OpenID Provider e tutti i token sono erogati dal token endpoint.

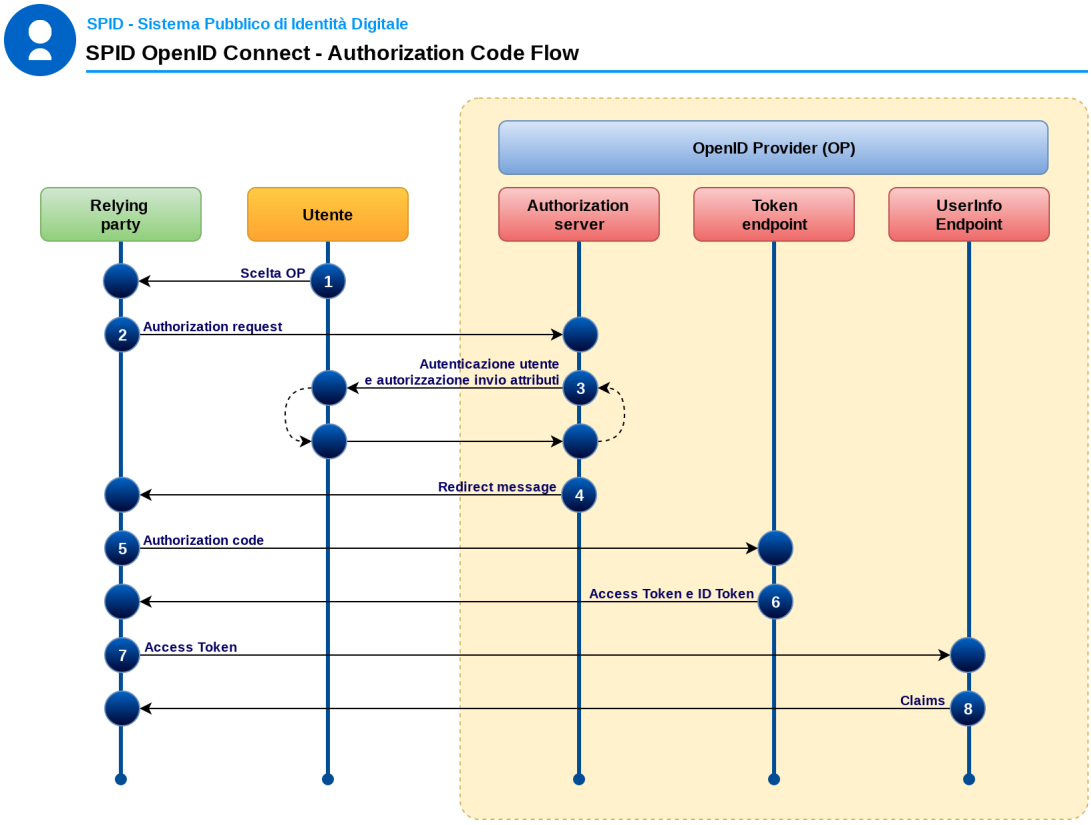


Figura 4.3: Flusso "Authorization code flow" implementato in SPID.

4.3 AUTHO

Autho è un prodotto dell'omonima azienda per la gestione dell' autenticazione ed autorizzazione per dispositivi, utenti e servizi altamente estensibile, basato su *tokens* di tipo JWT ed implementato a livello applicativo. Autho fornisce agli sviluppatori una *dashboard* completa per l'amministrazione dei dispositivi e delle metodologie di autenticazione. Nella *dashboard* si possono configurare dei *tenants* ovvero dei profili che raccolgono configurazioni ed APIs diverse. Verranno analizzate le metodologie specifiche per autenticare ed autorizzare servizi Machine to Machine dispositivi "limitati".

4.3.1 *Autenticazione ed autorizzazione di un servizio M2M*

Con "Servizio M2M" intendiamo un applicazione autenticata ed autorizzata senza l'intervento dell'utente ma attraverso un API presente su un server. Quest'ultima viene generata sfruttando la *dashboard* di Autho in cui si specificano configurazioni come la durata dei token, l'algoritmo di firma utilizzato (RS256 o HS256), quali permessi assegnare e quali applicazioni autorizzare.

L'API viene quindi attivata ed integrata nel server che gestirà Autho. Un applicazione o servizio S che vuole autenticarsi dovrà eseguire i seguenti passi:

1. Il servizio invia al server di autenticazione autho una richiesta POST contenente l'identificativo univoco ClientID e il segreto ClientSecret (una chiave condivisa scelta durante la configurazione di Autho).

```
{
      audience: "API",
      grant_type: "client_credentials",
      client_id: "",
      client_secret: ""
}
```

2. Il server Autho verifica i dati inviati.

3. Il server Autho risponde con un HTTP 200 contenente un token di accesso:

```
{
     "access_token":"eyJz93a...k4laUWw",
```

```
        "token_type":"Bearer",
        "expires_in":86400
}
```

4. il servizio è in grado di richiedere ed ottenere risorse utilizzando l'access token per invocare l'api, ad esempio può richiedere dei valori da un *timesheet*:

```
{
        'user_id': '007',
        'date': '2017-05-10T17:40:20.095Z',
        'project': 'StoreZero',
        'hours': 5
}
```

   Si conclude quindi il processo di autenticazione ed autorizzazione di un servizio, a partire dallo step numero quattro il servizio è in grado di interagire con il server richiedendo risorse.

4.3.2 *Autenticazione ed autorizzazione di un dispositivo*

Con Autho è possibile autenticare ed autorizzare servizi che operano su dispositivi, anche constrained. Per fare questo si utilizza un *device* intermedio capace di comunicare sulla rete e con l'utente, come un computer o uno smartphone. Si utilizza il *"Device Authentication Flow"* specificato in Oauth 2.0. Dato un dispositivo limitato che richiede una certa autorizzazione, il dispositivo intermedio e l'utente si hanno le seguenti fasi:

1. Il dispositivo limitato richiede l'autorizzazione al server di autho comunicando il suo ClientID.

2. Il server di autho risponde inviando vari campi tra cui device-code, user-code, verification-uri e interval.

3. Il dispositivo (intermedio) comunica all'utente il metodo da utilizzare per l'autorizzazione. Solitamente si prevede che l'user navighi nella pagina web verification-uri ed inserisca il numero user-code, un numero generato dal server di Autho e che viene mostrato fisicamente sul dispositivo in un processo simile ad OOB.

4. Il dispositivo entra quindi in una fase di *polling* con periodo di interrogazione `interval`. Uscirà dalla fase di polling nel momento in cui l'utente concluderà l'autorizzazione sul dispositivo intermedio.

Figura 4.4: Esempio di un dispositivo (una Smart TV) che riporta il codice di autorizzazione "ZGMG-DFSL". l'utente dovrà inserirlo nella pagina web *verification_uri*. Questa è una simulazione presente nella pagina web di Autho che mostra l'interazione tra il dispositivo intermedio e l'utente.

Il lavoro si sposta adesso sul dispositivo "intermedio", come accennato nel punto numero tre l'utente visita *verification_uri* ed inserisce l'*user_code*, dopodichè:

1. Se il numero inserito è corretto la pagina *verification_uri* richiederà all'utente di autenticarsi e chiederà l'autorizzazione per il dispositivo.

2. Se l'utente si autentica correttamente ed autorizza il dispositivo il processo si conclude.

3. Il dispositivo è in grado di richiedere risorse e comunicare con l'API.

Logout di un utente

Il Logout di un utente da una sessione di Autho può essere eseguito su tre livelli:

1. **Livello Sessione-Applicazione:** Si imposta la sessione dell'utente (Esempio un cookie) come non più valida.

2. **Livello Sessione-Autho:** In Autho si mantengono dei Cookies per ricordare le sessioni degli utenti, ad esempio nell'autenticazione ed autorizzazione di tipo SSO. In questo caso viene modificata la validità del cookie.

3. **Livello Identity Provider:** A questo livello si può forzare il logout dell'utente dalla piattaforma di *Identity Providing*.

Autho rappresenta un framework ben consolidato che supporta autenticazioni multifattore, SSO, Monitoring, SDK cross platform con molti esempi pronti per l'uso. È estremamente flessibile e presenta ottimizzazioni per dispositivi IoT.

Integrazione MQTT ed Autho

E' possibile integrare Autho con MQTT, nella documentazione ufficiale è riportata questa possibilità di integrazione con broker "mosca" (MQTT broker node.js) [8].

1. Il broker Mosca rimane in ascolto su una certa porta.

2. Si ridefinisce all'interno di mosca.js le funzioni di autenticazione pre-costruite: *AuthenticateWithCredentials, authenticateWithJWT, authorizePublish e authorizeSubscribe*. Nell'implementazione proposta quando un nodo client esegue un comando *CONNECT*, sul broker viene richiamata una funzione "authenticate" che autentica il client verificando la coppia $(\mathtt{username}, \mathtt{password})$ inserita nel comando *CONNECT*.

3. Autho verifica e in caso positivo rilascia il token JWT che identifica ed autorizza il client.

Questa implementazione presenta come svantaggio un maggior *overhead* e le credenziali vengono trasmesse in rete, cosa che sarebbe meglio sostituire implementando un sistema di autenticazione per certificati come TLS. Questo esempio mostra tuttavia l'estensibilità di questo protocollo.

5

CONFRONTO ED INTEGRAZIONE

5.1 CONFRONTO DELLE TECNOLOGIE ANALIZZATE

In questa sezione verrà eseguito un confronto tra le metodologie di autenticazione ed autorizzazione analizzate nei capitoli precedenti.

Nome protocollo	Layer	Fattore	Architettura	Basato su token	Procedura
EAP-AKA′	Network	Identità	Distribuita	Si(AV′)	Mutua
5G-AKA	Network	Identità	Distribuita	Si(AV)	Mutua
LoraWAN-OOTA	Network	Identità	Centralizzata	No	One Way
LoraWAN-ABP	Network	Fisico	Centralizzata	No	One Way
ZigBee	Network	Fisico	Centralizzata	No	One way
Bluetooth Mesh	Network	Centralizzata	Distribuita	No	Three Way
DIAMETER-EAP	Network	Identità	Centralizzata	No	Mutua

Tabella 5.1: Confonto delle tecnologie di autenticazione ed autorizzazione a livello di rete.

Nome protocollo	Protocollo	Fattore	Architettura	Basato su token	Procedura
Oauth	HTTP	Identità	Centralizzata	Si	3-way
Autho	HTTP/MQTT	Identità	Centralizzata	Si	3-way
DCAF	CoAP	Identità	Centralizzata	Si (Tickets)	Mutua autenticazione

Tabella 5.2: Confonto delle tecnologie di autenticazione ed autorizzazione a livello applicazione.

5.2 CONSIDERAZIONI

Scelto un certo sistema di autenticazione ed autorizzazione che opera a livello di rete si desidera che soddisfi le seguenti caratteristiche:

1. Deve essere uno standard aperto e non proprietario.
2. Deve garantire l'autenticazione del singolo dispositvo, sono da evitare in questo ambito tutti quei sistemi che prevedono autenticazioni di gruppo (come BLE Mesh).
3. Deve possedere una strategia di decommissioning forte.
4. Si deve limitare al minimo la trasmissione delle identità nella rete, soprattutto senza l'utilizzo di una tecnologia crittografica forte.
5. Si devono evitare quei sistemi che mantengono una quantità eccessiva di informazioni sensibili sulla rete nei dispositivi.
6. La confidenzialità deve essere mantenuta con sistemi crittografici forti.

Alla luce di queste osservazioni sono da scartare le tecnologie di autenticazione ed autorizzazione fornite da Bluetooth Mesh e da LorAWAN.

Delle rimanenti si ha 5G che soddisfa tutti i requisiti richiesti e garantisce quello che allo stato attuale può essere l'ottimo per la gestione dell'autenticazione ed autorizzazione a livello di rete in ambienti IoT.

ZigBee, almeno in versione "commercial", può essere una buona proposta per ambienti più "piccoli", tuttavia presenta il problema del "possesso delle chiavi". Il processo di autenticazione è basato soltanto sul possesso della chiave corretta da parte del dispositivo che deve essere autenticato, questo può portare ad attacchi noti come "NK Sniffing" (Acquisizione della chiave di rete) [16].

Attacchi di questo tipo possono permettere ad eventuali malinentzionati di accedere alla rete.

ZigBee può comunque essere considerato un buon *threadoff* tra economicità e affidabilità per applicazioni di media grandezza.

Per le tecnologie di autenticazione a livello applicazione ci aspettiamo che soddisfino le seguenti proprietà:

1. Essere standard aperti e non proprietari.
2. Essere ben studiate, flessibili e facilmente integrabili.
3. Metodi basati su token e certificati.

Tra le tecnologie analizzate rimangono DCAF e Oauth, tra queste DCAF è estremamente ottimizzato per IoT ma rimane un sistema poco studiato e di complessa applicazione.

L'utilizzo di Oauth (con eventuale layer di autenticazione OpenID Connect) diviene una tecnologia molto buona in quanto soddisfa tutte le caratteristiche richieste.

Se si considera un ambiente composto da IoT FoG e Cloud, l'utilizzo di un solido sistema di autenticazione ed autorizzazione a livello di rete per i dispositivi ed a livello applicazione per i servizi non basta a garantire il corretto funzionamento e la *security/reliability* del sistema, è necessario strutturare un opportuna *integrazione*.

Allo stato attuale della ricerca non è (ancora) stata proposta un integrazione totale che sfrutti al massimo il *machine-to-machine* ed in grado di operare autonomamente senza l'intervento umano.

5.3 INTEGRAZIONE IOT-FOG-CLOUD

Si propone un'architettura che integri dispositivi IoT, FoG e Cloud con l'obbiettivo di analizzare il problema dell'autenticazione ed autorizzazione ai vari livelli.

Con la terminologia "Sistemi SFC" (Sistemi Sensors-Fog-Cloud) definiremo le architetture di rete composte da (molti) nodi dispositivo topologicamente vicini tra loro, (alcuni) nodi FoG prossimi ai dispositivi e (pochi) nodi Cloud geograficamente lontani dal resto della rete.

L'analisi di questa integrazione sarà suddivisa per livelli.

Livello dei dispositivi

Supponendo di lavorare in un ambito industriale molto vasto si può prevedere l'utilizzo di 5G come tecnologia di comunicazione device-device e device-fog.

Dal punto di vista dell' autenticazione ed autorizzazione 5G presenta un *framework* robusto ed altamente estensibile, tra i metodi di autenticazione analizzati si può preferire l'utilizzo di EAP-AKA' in quanto basato su EAP. Una valida alternativa potrebbe essere EAP-TLS basato su EAP con certificati TLS, questo metodo tuttavia implicherebbe lo storing e la gestione dei certificati sia lato client che lato server e questo potrebbe costituire un problema su dispositivi low-power.

L'utilizzo di 5G per comunicazioni device-device e device-fog si adatta bene in alcuni ambiti come le smart grid, smart cities, industrie su larga scala.

In altri ambiti come la piccola industria, l'home automation o reti ospedaliere rimane difficile costruire una rete locale interna che sia basata sul 5G. Si richiederebbe infatti l'implementazione di una core network, software di controllo, SIM/eSIMs per ogni dispositivo, hardware in grado di virtualizzare nodi e funzioni di rete ed altre tecnologie non facilmente accessibili da un qualunque reparto IT.

Recentemente sono state proposte alcune soluzioni per riprodurre una rete 5G "privata" e scalabile, utilizzando tecnologie per facilitare al massimo il setup. Un esempio è l'azienda italiana di nome Athonet.

In molte realtà di dimensioni più contenute si può prevedere di utilizzare una rete ZigBee per il device to device e device to fog. Questa tecnologia infatti è certamente più economica dell'implementazione di una rete privata 5G, nella gestione del problema dell'autenticazione ed autorizzazione si andrebbe ad utilizzare la versione "commercial".

Livello di management

Rientrano in questo livello i nodi FoG, i dispositivi comunicano tra loro (device-device) e con i nodi FoG (device-fog) attraverso 5G o ZigBee secondo l'implementazione scelta, per realizzare la comunicazione FoG-FoG e Fog-Cloud si propone l'utilizzo della tecnologia MQTT.

Questa scelta ricade sul fatto che è un protocollo leggero, affidabile e facilmente scalabile in funzione del numero di dispositivi connessi.

L'uso di MQTT prevede un broker centrale, ogni nodo FoG e Cloud si sottoscrive soltanto agli argomenti di suo interesse sul broker. Il cloud sarà "interessato" ad argomenti di storing e consuming dei dati, i nodi

FoG a loro volta si sottoscriveranno a topics riguardo l'elaborazione dei dati ed operazione di gestione come il *load balancing*, inizializzazione della rete ed elaborazione dei dati.

Esistono vari broker MQTT, se viene scelto 5G per il device-device e device-fog è possibile scegliere il broker EMQX, una tecnologia di *brokering* opensource compatibile ed ottimizzata per reti 5G.

Se si utilizza ZigBee si prevede l'utilizzo dell'interfaccia zigbee2mqtt con broker Mosquitto come proposto nelle documentazioni.

A livello FoG e Cloud l'autenticazione dei servizi sarà eseguita nell' application layer su un nodo delegato che svolgerà il ruolo di Authentication (Authorization) server; il nodo che ricoprirà questo ruolo potrebbe essere il broker MQTT stesso.

Si può prevedere l'utilizzo di Oauth 2.0 con layer di identità/autenticazione OpenID Connect.

Il livello FoG può essere composto da sotto-livelli specializzati sulla base della funzione che ricoprono. Si può prevedere un livello FoG per la raccolta delle risorse e pre-elaborazione ed un livello per le operazioni di mantenimento della rete e comunicazione con il cloud.

Livello di storing e consuming

In questo livello si considera principalmente il Cloud che svolgerà operazioni di storing dei dati e consuming di questi. Alcuni dati verranno resi disponibili all'esterno attraverso servizi di tipo RESTFul.

Il Cloud comunica direttamente con il server di autenticazione ed autorizzazione (broker MQTT), autenticando i propri servizi e rimanendo in attesa di topics sullo storing e l'analisi delle informazioni.

Le applicazioni ed i servizi del cloud saranno i principali *consumer dei dati* e vengono autenticati sul broker MQTT.

Si può notare che per risolvere il problema dell'autenticazione ed autorizzazione vengono proposte principalmente soluzioni di tipo centralizzato.

Queste presentano problemi riguardo la scalabilità, la sicurezza (si ha infatti un solo punto centrale vulnerabile) e l'affidabilità (il nodo centrale può essere soggetto maggiormente a malfunzionamenti poiché è estremamente attivo).

5.4 IL PROBLEMA DELLA PROPRIETÀ DEI DATI

Un ulteriore problema che rimane difficile da gestire in soluzioni centralizzate è quello della *"proprietà"* dei dati generati e trasmessi, è importante analizzare questo problema in quanto potrebbe essere risolto con soluzioni nuove di autenticazione/autorizzazione e gestione della rete distribuite.

Queste soluzioni rappresentano un nuovo orizzonte nella ricerca e sono basate su *blockchains*.

Verrà quindi introdotto brevemente l'argomento delle *blockchain* e verrà analizzata l'integrazione con l'ambiente IoT FoG e Cloud con particolare attenzione ai problemi dell'autorizzazione/autenticazione e possesso dei dati.

5.4.1 *Blockchains e smart contracts*

Sulle *blockchains* è presente una discreta confusione, spesso vengono fraintese con particolari strumenti finanziari o con tecnologie di intelligenza artificiale.

Una blockchain è un particolare tipo di struttura dati distribuita che viene costruita ricorsivamente mediante concatenazione di blocchi contenenti valori (sotto forma di *transazioni*) e legati attraverso funzioni hash crittografiche.

La caratteristica principale delle blockchain è l'utilizzo del *consenso* (ottenuto attraverso algoritmi di *proving*, generalmente di tipo Proof of Work oppure Proof of Stake eseguiti da *miners*) per mantenere una ed una sola catena immutabile tra più entità e verificare i dati inseriti.

Ciò che è scritto nella catena non può essere ne modificato ne cancellato, se un elemento viene aggiornato rimane anche la sua versione precedente (che rimanderà alla versione aggiornata).

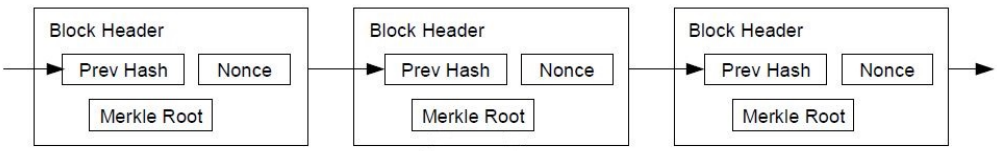


Figura 5.1: Struttura di una blockchain,si può notare la concatenazione dei blocchi. il capo "PrevHash" contiene l'hash della serializzazione del blocco precedente.

Sulla blockchain sono presenti blocchi contenenti informazioni.

In alcune di queste strutture, come quella di Ethereum oltre alle informazioni possono essere immagazzinati ed eseguiti gli *Smart Contracts*.

Gli smart contracts <contratti intelligenti> sono definiti come *"contractual type arrangement"* ovvero un insieme di clausole contrattuali espresse attraverso linguaggio informatico che opera su un software o un protocollo.

I contratti intelligenti possiedono la caratteristica di eseguirsi automaticamente sulla base di determinate condizioni predeterminate dalle parti, questi costituiscono delle "regole digitali" per lo scambio e negoziazione dei dati tra parti.

Si ricorda brevemente la definizione di *dato* come[20]

> Una descrizione elementare, spesso codificata, di un'informazione, un'entità, di un fenomeno, di una transazione, di un avvenimento o di altro.

Ad un dato si possono associare un:

1. **Possessore:** L'entità che *possiede* fisicamente il dato, nel caso di una rete questo è un nodo.
2. **Proprietario:** L'entità che *ha prodotto* il dato. Nel caso di un dispositivo di e-health si può identificare con proprietario la persona fisica che ha emesso certe misurazioni.

E' importante sottolineare questa sottile differenza poiché in una rete densa come un ambiente SFC si riesce a mantenere traccia del possesso dei dati (che divengono risorse) ma vi è un forte rischio di perdere traccia dell' *ownership* di questi.

Nella ricerca sono state proposte integrazioni tra blockchains e smart contracts con l' IoT come EdgeFoG e FoGBus.

Verrà analizzata una generica integrazione tra questi, con una particolare attenzione al problema dell'autenticazione/autorizzazione e alla proprietà dei dati in questi nuovi sistemi che prendono il nome di BoT (Blockchain of things).

5.4.2 *Integrazione con ambiente IoT-FoG-Cloud*

Si utilizza una blockchain per costruire un meccansmo di autenticazione ed autorizzazione per i servizi.

Si definisce un architettura composta da:

1. **Livello Dispositivi :** I dispositivi vengono autenticati attraverso una tecnologia di autenticazione a livello di rete. Un dispositivo raccoglie un certo dato e lo invia per l'elaborazione ai nodi FoG.

2. **Livello FoG :** Il FoG interagisce con i dispositivi e con la blockchain, si predispone l'utilizzo di una blockchain in cui possono essere eseguiti gli smart contracts come la b.c. di Ethereum.

3. **Livello Cloud :** Il cloud mantiene il ledger condiviso ed esegue operazioni di manutenzione sulla catena. Il cloud eroga servizi per il consuming dei dati.

Nella blockchain condivisa tra tutti i nodi FoG ed il nodo Cloud viene eseguito un *singolo* smart contract che "mappa" tutti i nodi FoG con tutti i relativi dispositivi IoT e una lista di consumers autorizzati ad accedere ai dati erogati.

Nel momento in cui un utente U vuole accedere ad un certo dato D invia una "richiesta" al cloud che consulta la blockchain ed esegue lo smart contract. Lo smart contract verifica l'identità di U e la sua associazione con il dispositivo che eroga il dato D. Se l'esito della verifica è positivo U accede al dato.

L'utilizzo di questa architettura unita ad un opportuna gestione dei dati erogati da parte dei nodi FoG che "scriveranno" il possessore del dati dati incapsulati nei vari blocchi permette di non perdere traccia dell'ownership del dato.

5.4.3 *Considerazioni*

Attraverso l'utilizzo di una blockchain in un ambiente IoT-FoG-Cloud si possono risolvere i seguenti problemi:

1. **Ownership dei dati :** Ogni blocco della catena mantiene traccia dei proprietari dei vari dati sotto forma di transazioni, questi dati non possono essere estratti dalla catena da fonti non autorizzate.

2. **Accesso da terze parti :** Nella BoT i dati sono distribuiti pertanto non c'è un possessore centrale, questi sono crittografati in modo da rendere impossibile recuperarli interamente attraverso attacchi o processi di *tampering.*

3. **Accesso non autorizzato :** La struttura distribuita e la crittografia forte garantiscono un elevata resistenza contro tentativi di accesso non autorizzato.

4. **Manipolazione dei Dati :** Per definizione una blockchain si utilizza per costruire un database distribuito che garantisca l'integrità e L'immutabilità del contenuto, questo implica che i dati non possono essere modificati involontariamente. Qualora venissero modificati sarebbero comunque disponibili le versioni precedenti.

Queste soluzioni allo stato attuale rimangono proposte in ambito di ricerca, non ci sono implementazioni reali che ne dimostrino l'effettiva affidabilità e funzionamento, potrebbero tuttavia divenire un buon metodo per la gestione dell'autenticazione dei servizi e la gestione dei dati in un sistema distribuito a livello applicazione.

Uno sviluppo interessante per applicazioni di questo tipo potrebbe essere l'unione di più blockchains corrispondenti a processi industriali diversi attraverso opportuni smart contracts per unire "logicamente" con regole digitali i processi produttivi di più aziende.

Tra gli svantaggi di un simile sistema si può prevedere:

1. Maggior carico sulla rete.
2. Necessità di hardware performante per le operazione crittografiche richieste a partire dal livello Fog.
3. Necessità di tecnologie di storing avanzate sia in termini di dimensione che di mantenimento

6

CONCLUSIONI

In questa trattazione è stato analizzato un importante problema relativo alla sicurezza delle reti di dispositivi IoT, quello dell'autenticazione ed autorizzazione.

E' stato scomposto il problema in due sotto-problemi, si sono analizzate le tecnologie che operano a livello di rete e a livello applicazione, sono stati evidenziati problemi e *best practices*, è stata analizzata una possibile integrazione tra IoT,FoG e Cloud e queste tecnologie.

E' stato trattato il problema del "trust" di una rete, del possesso e proprietà dei dati generati e circolanti, a questo proposito si è parlato di *blockchains*, *smart contracts* ed alternative "nuove" e distribuite.

In definitiva allo stato attuale della ricerca non sono ancora stati proposti metodi di autenticazione ed autorizzazione a livello di rete ed a livello applicazione che si integrino perfettamente con l'obbiettivo di sfruttare al massimo le richieste di una rete di dispositivi.

La ricerca in questo ambito sta procedendo in direzione della decentralizzazione dei sistemi IoT, verso lo sfruttamento ottimale del m2m e nuove tecnologie di gestione di dati condivisi come la *blockchain.*

6.1 RINGRAZIAMENTI

Il raggiungimento di questo traguardo per me non è solo il sostenimento di una prova, ma il compiersi di un percorso di crescita personale ed interiore durato molti anni.

Guardandomi indietro mi chiedo spesso come sia possibile, pensando alla posizione da cui sono partito, trovarmi qui adesso a concludere le ultime righe della mia tesi.

La realtà è che questo percorso è stato ricco di ostacoli, spunti per crescere e migliorarsi.

Sono molte le persone che vorrei ringraziare per avermi aiutato ad essere qui ora.

Inizio dalla mia famiglia, in particolare i miei genitori, mia sorella ed i miei nonni, che mi sono sempre stati vicino ed hanno creduto in me.

Un ringraziamento particolare va a Ginevra che in questi anni con la sua comprensione e pazienza ha saputo assistermi e consigliarmi nei momenti più difficili.

Un grande abbraccio va ai miei zii Paolo e Marta e ai miei amici e colleghi, che mi hanno regalato momenti indimenticabili.

Un pensiero affettuoso va a mia sorella Francesca la quale da sempre mi ha fatto sentire la sua vicinanza.

Non posso fare a meno di ricordare Mara con la quale ho condiviso tanti momenti della mia infanzia.

Ringrazio tutte le persone che mi hanno sostenuto e mi hanno permesso di intraprendere questo bellissimo percorso.

Alessandro.

BIBLIOGRAFIA

[1] AGID. «Linee Guida OpenID Connect in SPID: 4.1. Applicazioni per dispositivi mobili». In: URL: https://docs.italia.it/AgID/documenti-in-consultazione/lg-openidconnect-spid-docs/it/bozza/flusso/applicazioni-per-dispositivi-mobili.html.

[2] J. Arkko e H. Haverinen. «Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA)». In: 4187. RFC Editor, gen. 2006.

[3] Frederik Armknecht, Zinaida Benenson, Philipp Morgner e Christian Muller. «On the Security of the ZigBee Light Link Touchlink Commissioning Procedure». In: (2016), pp. 1–12.

[4] Carsten Bormann, Mehmet Ersue e Ari Keränen. «Terminology for Constrained-Node Networks». In: Request for Comments 7228. RFC Editor, mag. 2014. DOI: 10.17487/RFC7228. URL: https://rfc-editor.org/rfc/rfc7228.txt.

[5] P. Calhoun, J Loughney, E. Guttman, G. Zorn e J. Arkko. «Diameter Base Protocol». In: 3588. http://www.rfc-editor.org/rfc/rfc3588.txt. RFC Editor, set. 2003. URL: http://www.rfc-editor.org/rfc/rfc3588.txt.

[6] «Cloud IoT Core Documentation». In: Google. URL: https://cloud.google.com/iot/docs/concepts/device-security.

[7] «Diameter Protocol Explained: Diameter Address Format». In:

[8] Autho Documentation. «Authenticating & Authorizing Devices using MQTT with Autho». In: URL: https://auth0.com/docs/integrations/authenticate-devices-using-mqtt.

[9] P. Eronen, T. Hiller e G. Zorn. «Diameter Extensible Authentication Protocol EAP Application». In: 4072. RFC Editor, ago. 2005.

[10] Shahin Farahani. ZigBee Wireless Networks and Transceivers. USA: Newnes, 2008. ISBN: 0750683937.

[11] Stefanie Gerdes, Olaf Bergmann e Carsten Bormann. «Delegated CoAP Authentication and Authorization Framework (DCAF)». In: draft-gerdes-ace-dcaf-authorize-04. Work in Progress. Internet Engineering Task Force, ott. 2015. URL: https://datatracker.ietf.org/doc/html/draft-gerdes-ace-dcaf-authorize-04.

[12] Mesh Working Group. Mesh Profile. Vol. 1.0.1. Gen. 2019, pp. 247–263.

[13] Dick Hardt. The OAuth 2.0 Authorization Framework. RFC 6749. Ott. 2012. DOI: 10.17487/RFC6749. URL: https://rfc-editor.org/rfc/rfc6749.txt.

[14] Hugo Krawczyk, Mihir Bellare e Ran Canetti. «HMAC: Keyed-Hashing for Message Authentication». In: Request for Comments 2104. RFC Editor, feb. 1997. DOI: 10.17487/RFC2104. URL: https://rfc-editor.org/rfc/rfc2104.txt.

[15] David Linthicum. «The state of IoT standards 2016: How to choose». In: (gen. 2016).

[16] Olayemi Olawumi, Keijo Haataja, M. Asikainen, Niko Vidgren e Pekka Toivanen. «Three Practical Attacks Against ZigBee Security: Attack Scenario Definitions, Practical Experiments, Countermeasures, and Lessons Learned.» In: dic. 2014. DOI: 10.1109/HIS.2014.7086198.

[17] S. Rommer, P. Hedman, M. Olsson, L. Frid, S. Sultana e C. Mulligan. 5G Core Networks: Powering Digitalization. Elsevier Science, 2019. ISBN: 9780081030097.

[18] Dipa Soni e Ashwin Makwana. «A SURVEY ON MQTT: A PROTOCOL OF INTERNET OF THINGS(IOT)». In: apr. 2017.

[19] Mohammad Wazid, Kumar Das Ashok, Vanga Odelu, Neeraj Kumar, Mauro Conti e Minho Jo. «Design of Secure User Authenticated Key Management Protocol for Generic IoT Networks». In: IEEE Internet of Things Journal PP (dic. 2017), pp. 1–1. DOI: 10.1109/JIOT.2017.2780232.

[20] Wikipedia. «Dato — Wikipedia, The Free Encyclopedia». In: [Online; accessed 30-August-2020]. 2020.

[21] Wikipedia. «JSON Web Token — Wikipedia, The Free Encyclopedia». In: 2020.

[22] Martin Woolley e Sarah Schmidt. Bluetooth mesh networking. bluetooth.com, 2017. URL: https://www.bluetooth.com/wp-content/uploads/2019/03/Mesh-Technology-Overview.pdf.

[23] Shelby Zach, Klaus Hartke e Carsten Bormann. «The Constrained Application Protocol (CoAP)». In: Request for Comments 7252. RFC Editor, giu. 2014. DOI: 10.17487/RFC7252. URL: https://rfc-editor.org/rfc/rfc7252.txt.

[24] G. Zorn. Diameter Network Access Server Application. RFC 7155. RFC Editor, apr. 2014.