



DIPARTIMENTO DI
INGEGNERIA INFORMATICA,
MODELLISTICA, ELETTRONICA
E SISTEMISTICA
DIMES

Corso di Laurea Magistrale in Ingegneria Informatica

Progetto di Network Security

*Sicurezza del protocollo 802.15.4
analisi delle vulnerabilità e patching*

Docente
Prof Floriano De Rango

Studenti
Antonio Rosanò (204967)
Lorenzo Morelli (207038)
Antonio Piluso (204865)

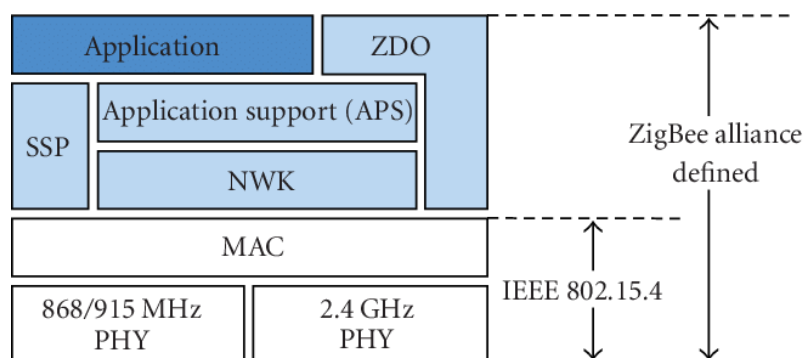
A.A. 2020-2021

INDICE

1	Introduzione.....	3
2	IEEE 802.15.4.....	4
3	Sicurezza IEEE 802.15.4.....	8
4	Vulnerabilità.....	13
4.1	Replay Protection Attack	13
4.2	No Integrity on Acknowledgment Packets.....	14
5	Modifiche proposte	16
5.1	Soluzioni disponibili	16
5.1.1	Timestamp come Frame counter.....	16
5.1.2	Autenticazione ACK.....	16
5.1.3	Gestione proattiva del coordinatore	17
5.1.4	Modalità CCM*	17
5.2	Soluzioni adottate	17
5.2.1	Autenticazione dell'ACK.....	18
5.2.2	Deprecazione livello 4 AES-CTR.....	20
6	Performance	23
6.1	Primo scenario	26
6.2	Secondo scenario	27
7	Conclusioni.....	29
8	Riferimenti	30

1 INTRODUZIONE

Il campo della network security consiste nell'insieme delle politiche, dei processi e delle pratiche adottate per prevenire, rilevare e monitorare l'accesso non autorizzato, l'uso improprio, la modifica o la negazione di una rete di computer e delle risorse accessibili dalla rete. Rendere sicure, sotto diverse asserzioni di sicurezza, le reti è una delle sfide più impegnative in questo decennio, questo perché la tecnologia avanza sempre in maniera spedita e l'esigenza, che da sempre caratterizza lo sviluppo tecnologico, di trovare qualcosa che sia sempre più soddisfacente per l'utente finale porta con sé complessità maggiori e quindi più strade di intrusione, manomissione e corruzione che un hacker può sfruttare per trarre vantaggio da questi contesti.



Il progetto è incentrato sullo studio, l'analisi e l'implementazione dei meccanismi di sicurezza dettati dallo standard IEEE 802.15.4. L'obiettivo del progetto è quello di mettere in pratica le tecniche di cifratura, autenticazione e integrità viste al corso di Network security mantenendo un occhio critico sulle scelte e proporre delle modifiche finalizzate a rendere il protocollo protetto da vulnerabilità note. Per questo motivo il progetto è stato suddiviso in diversi step logici:

- Studio del protocollo
- Implementazione meccanismi sicurezza
- Studio e analisi delle vulnerabilità
- Studio ed implementazione delle patch di sicurezza
- Analisi delle performance

Tali argomenti saranno trattati in maniera approfondita nei capitoli a seguire facendo emergere la logica critica che ha contraddistinto tutto lo sviluppo del progetto. Nella stesura dell'elaborato sono stati di fondamentale importanza diversi documenti/articoli di ricerca che saranno citati nel documento stesso (riferimenti a fine documento).

Il codice su cui sono state fatte le simulazioni è stato reso pubblico e consultabile da questo indirizzo:

https://github.com/ToniRos95/SEC802.15.4_Omnetpp

2 IEEE 802.15.4

IEEE 802.15.4 è uno standard che definisce il funzionamento delle low-rate wireless personal area network (LR-WPAN). Al suo interno specifica il livello fisico e MAC per tali reti, ed è mantenuto dalla IEEE 802.15 working group, che ha definito tale standard nel 2003. Esso è la base per lo sviluppo di altri standard come Zigbee, ISA100.11a, WirelessHART, MiWi, 6LoWPAN, ciascuno dei quali estende ulteriormente i livelli superiori che non sono definiti in IEEE 802.15.4.

Lo standard **IEEE 802.15.4** [1] cerca di realizzare dei livelli di rete che vadano a rispecchiare le caratteristiche delle reti WPAN (Wireless Personal Area Network), che si concentrano sulla comunicazione a basso costo e a bassa velocità tra i vari dispositivi. Tutto ciò può andare in contrasto con altri approcci, come il Wi-Fi, che offrono più larghezza di banda ma richiedono più potenza. La caratteristica principale è quindi l'utilizzo di una comunicazione di dispositivi vicini con poca o nessuna infrastruttura sottostante, con l'intenzione di sfruttarla per ridurre ulteriormente il consumo energetico. In generale si ha un raggio di comunicazione di 10 metri, con una velocità di trasferimento di 250 kbit/s. Ovviamente è possibile raggiungere dei compromessi per favorire dispositivi embedded che richiedono un minor dispendio di energia per attuare la comunicazione. Pertanto, per ottenere ciò, è possibile utilizzare velocità ancora più basse. Come già accennato, l'obiettivo principale di IEEE 802.15.4, per quanto riguarda i WPAN, è il raggiungimento non solo di bassi consumi, ma anche di bassi costi di produzione attraverso l'uso di ricetrasmittitori relativamente semplici, consentendo al contempo flessibilità e adattabilità dell'applicazione.

Il **livello fisico (PHY)** fornisce il servizio di trasmissione dei dati e un'interfaccia al physical layer management entity, il quale offre accesso a ogni funzione di gestione del livello fisico e mantiene un database di informazioni sulle reti WPAN. Pertanto, il PHY gestisce il ricetrasmittitore radio fisico ed esegue la selezione del canale insieme alle funzioni di gestione dell'energia e del segnale. Funziona su una delle tre possibili bande di frequenza senza licenza:

- 868,0–868,6 MHz: Europa, consente un canale di comunicazione (2003, 2006, 2011)
- 902-928 MHz: Nord America, fino a dieci canali (2003), esteso a trenta (2006)
- 2400-2483,5 MHz: utilizzo in tutto il mondo, fino a sedici canali (2003, 2006)

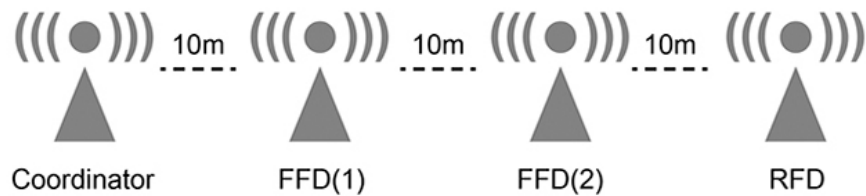
La versione dello standard del 2003 specifica due livelli fisici: uno che lavora nelle bande 868/915 MHz con velocità di trasferimento di 20 e 40 kbit / s, e uno nella banda 2450 MHz con una velocità di 250 kbit / s. La revisione del 2006 migliora le velocità di trasmissione dati massime delle bande 868/915 MHz, portandole a supportare anche 100 e 250 kbit / s. Inoltre, prosegue definendo quattro strati fisici a seconda del metodo di modulazione utilizzato.

Il **Medium Access Control (MAC)** consente la trasmissione di frame MAC tramite l'utilizzo del canale fisico. Oltre al servizio dati, offre un'interfaccia per la gestione dell'accesso al canale fisico e il beaconing di rete. Controlla anche la convalida dei frame, garantisce intervalli di tempo e gestisce le associazioni dei nodi. Infine, offre punti di aggancio per servizi sicuri.

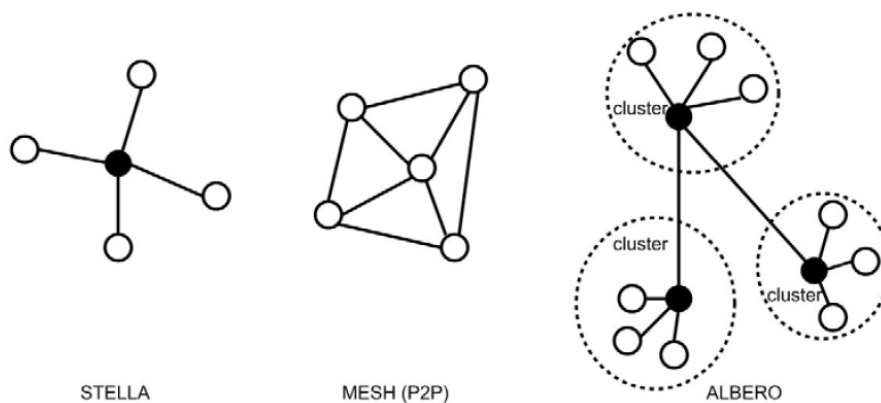
Altre specifiche, come ZigBee, SNAP e 6LoWPAN/Thread, si basano su questo standard. I sistemi operativi RIOT, OpenWSN, TinyOS, Unison RTOS, DSPnano RTOS, nanoQplus, Contiki e Zephyr utilizzano anche alcuni elementi hardware e software IEEE 802.15.4.

Lo standard definisce due tipi di nodi all'interno della rete. Il primo è il **full function device (FFD)**, che può fungere sia da coordinatore di una rete che da nodo comune. Al suo interno implementa un modello generale di comunicazione che gli consente di parlare con qualsiasi altro dispositivo. D'altra

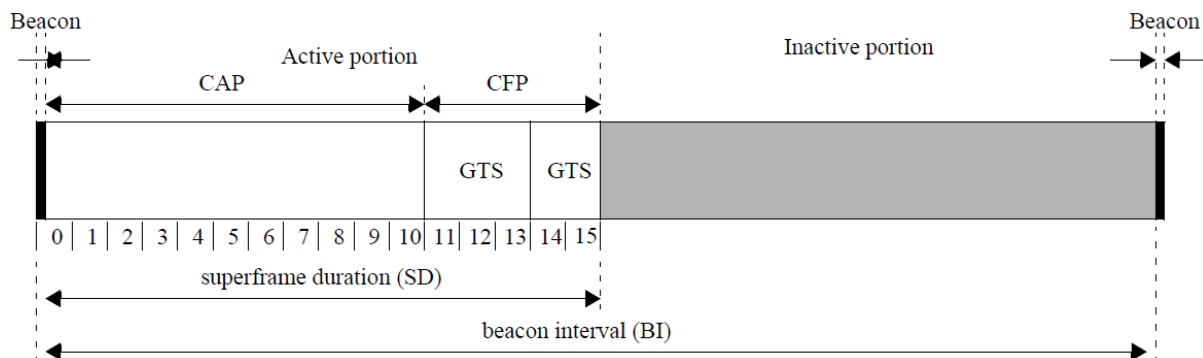
parte, ci sono i **reduced function device (RFD)**. Essi sono pensati per essere dispositivi estremamente semplici con risorse e requisiti di comunicazione modesti. Per questo motivo, possono comunicare solo con FFD e non possono mai agire come coordinatori.



Le reti possono essere costruite come reti **peer-to-peer** o a **stella**. Tuttavia, per un corretto funzionamento, ogni rete necessita di almeno un FFD che svolga il ruolo di **coordinatore**. Ogni dispositivo ha un identificatore univoco a 64 bit (MAC address esteso) e, se vengono soddisfatte alcune condizioni, è possibile utilizzare identificatori brevi a 16 bit (PANID) all'interno di un ambiente limitato. In ogni dominio PAN quindi, le comunicazioni useranno probabilmente identificatori brevi. La topologia peer-to-peer è stata invece pensata per scenari più complessi (come, ad esempio, le reti wireless di sensori) in cui la comunicazione multi-hop rappresenta un requisito imprescindibile. È importante sottolineare che lo standard supporta la formazione di topologie peer-to-peer, ma che la gestione dell'instradamento dei pacchetti in tali reti sia comunque demandata ai protocolli di routing, usualmente implementati ai livelli più alti dello stack protocollare. Nella formazione di topologie peer-to-peer, continua ad esistere il PAN Coordinator ma è consentito anche agli altri FFD di operare come coordinatori secondari, fornendo funzionalità di accesso alla rete e di sincronizzazione agli altri dispositivi interconnessi. La struttura può essere estesa come una rete **mesh** generica i cui nodi sono reti di alberi di cluster con un coordinatore locale per ogni cluster, oltre al coordinatore globale.



Una rete LR-WPAN, secondo lo standard IEEE 802.15.4, può opzionalmente operare in modalità **beacon-enabled**. In questo caso, l'asse temporale viene suddiviso in una sequenza di **super-frame**, ciascuno dei quali è delimitato da appositi pacchetti di segnalazione (definiti beacon).



I beacon sono trasmessi dai nodi coordinator e sono responsabili della sincronizzazione di tutti i dispositivi della rete. In tale modalità di funzionamento, il super-frame viene ripartito in time-slot elementari e contiene obbligatoriamente un Contention Access Period (CAP), nel quale l'accesso multiplo al canale è gestito tramite una variante a basso consumo energetico dell'algoritmo **CSMA/CA**. Sempre in tale modalità operativa, il super-frame può opzionalmente prevedere un Contention Free Period (CFP), in cui talune stazioni possono ottenere l'accesso al mezzo privo di collisioni in appositi time-slot garantiti (Guaranteed Time Slot, GTS), ed un Inactive Period, nel quale le interfacce radio possono essere messe in uno stato a basso consumo energetico per risparmiare le batterie dei dispositivi. Quando i beacon non sono abilitati, i nodi accedono al canale semplicemente utilizzando l'algoritmo CSMA/CA ed inoltre non è prevista alcuna ripartizione in time-slot e super-frame dell'asse temporale. Per quanto riguarda, invece, le modalità secondo cui i nodi della WPAN si scambiano reciprocamente i messaggi, si devono considerare tre distinte possibili interazioni:

- comunicazione dispositivo → coordinator: Un generico nodo della WPAN può in ogni momento inviare i propri dati al coordinatore utilizzando l'algoritmo di accesso multiplo CSMA/CA.
- comunicazione coordinator → dispositivo: quando un dispositivo vuole ricevere i dati dal proprio coordinatore, esso invia una richiesta al coordinatore e rimane in attesa dei dati. Nella modalità beacon enabled il coordinatore esplicitamente dichiara nel messaggio di beacon quali sono i nodi figli per cui dispone di dati pendenti.
- comunicazione dispositivo → dispositivo: è utilizzata in topologie peer-to-peer e richiede che i nodi siano tra loro sincronizzati.

Lo standard IEEE 802.15.4, come detto in precedenza, è stato concepito a supporto di reti wireless a corto raggio (WPAN), basso rate e ridotti consumi energetici. Questi requisiti non possono essere soddisfatti dall'algoritmo CSMA/CA implementato nello standard IEEE 802.11, poiché quest'ultimo non è stato ottimizzato per gli stessi obiettivi. Infatti, nelle reti WiFi, le stazioni interessate ad accedere al canale devono eseguire l'operazione di Carrier Sensing durante l'intervallo di backoff (oltre che nel DIFS) prima di poter trasmettere. Questo comporta un elevato dispendio di energia, visto che le operazioni di sensing causano elevati consumi, paragonabili a quelli che si ottengono in trasmissione. Pertanto, nello standard IEEE 802.15.4, non è prevista alcuna operazione di sensing del canale durante l'intervallo backoff, lasciando che, al suo scadere, i nodi interessati a trasmettere eseguano il carrier sensing per un breve intervallo di tempo pari a due time-slot (procedura di Clear Channel Assessment, CCA).

Viene di seguito riportato il generico **frame MAC** utilizzato nel protocollo:

Octets: 2	1	0/2	0/2/8	0/2	0/2/8	0/5/6/10/14	variable	2
Frame Control	Sequence Number	Destination PAN Identifier	Destination Address	Source PAN Identifier	Source Address	Auxiliary Security Header	Frame Payload	FCS
		Addressing fields						
MHR							MAC Payload	MFR

Le varie tipologie di pacchetti verranno successivamente analizzati nel capitolo relativo alla sicurezza.

3 SICUREZZA IEEE 802.15.4

La specifica 802.15.4 descrive protocolli wireless di accesso ai media per dispositivi di rete personali che vengono utilizzati in diversi contesti; per quanto riguarda gli ambienti indoor (domestici e non), esso consente di monitorare e/o gestire da remoto ad esempio:

- l'accensione e lo spegnimento delle luci;
- la regolazione del colore e dell'intensità dell'illuminazione;
- elettrodomestici intelligenti, televisori, digitale terrestre e altri dispositivi elettronici;
- l'allarme e i sistemi antifurto wireless;
- l'accensione e lo spegnimento dell'aria condizionata e del riscaldamento;
- l'apertura e la chiusura delle tapparelle;
- gli accessi in azienda o in ufficio tramite badge o sistemi biometrici;
- i servizi di teleassistenza;
- i parametri vitali dei pazienti.

Per quanto riguarda gli ambienti outdoor (domestici e non), esso consente di monitorare e/o gestire da remoto ad esempio:

- sistemi e centraline di irrigazione;
- l'apertura di cancelli e garage;
- i parametri ambientali come umidità, temperatura e inquinamento;
- il pagamento dei pedaggi.

Queste applicazioni spesso usano dispositivi incorporati controllati da un microcontrollore a 8 o 16 bit destinati a funzionare senza intervento umano per mesi. Il lungo periodo di funzionamento incustodito con hardware poco potente ha due implicazioni per la progettazione di tali sistemi: il software che gira su questi dispositivi deve essere semplice e corretto, e i dispositivi devono fare un uso efficiente della loro energia limitata. Il chip di comunicazione wireless che fa parte di questi dispositivi deve rispettare questi requisiti, poiché esso ed il microcontrollore rappresentano le due maggiori fonti di consumo energetico. La specifica 802.15.4 è destinata a supportare una varietà di applicazioni, molte delle quali sono sensibili alla **sicurezza**. Per esempio, consideriamo il caso di una rete di sensori che misura l'occupazione di un edificio come sistema di allarme: c'è un'ovvia preoccupazione per la **privacy** nel tracciare le persone nell'edificio. Inoltre, se la rete non è protetta, un avversario potrebbe modificare e iniettare messaggi per causare un allarme o, cosa più preoccupante, per sopprimere segnali di allarme legittimi. Molte applicazioni richiedono riservatezza e la maggior parte ha bisogno di protezione dell'integrità.

Il sottolivello MAC è responsabile di fornire servizi di sicurezza su specifici frame in entrata e in uscita quando richiesto dai livelli superiori. Questo standard supporta i seguenti servizi di sicurezza:

- Riservatezza
- Autenticità
- Protezione da replay attack

Un dispositivo può **opzionalmente** implementare la sicurezza. Se non la implementa, non deve fornire un meccanismo per il sottolivello MAC per eseguire qualsiasi trasformazione crittografica sui frame in entrata e in uscita né richiedere alcun attributo PIB associato alla sicurezza. Un dispositivo che implementa la sicurezza fornisce un meccanismo per il sottolivello MAC per fornire trasformazioni

crittografiche sui frame in entrata e in uscita utilizzando le informazioni negli attributi PIB associati alla sicurezza solo se l'attributo `macSecurityEnabled` è impostato su `TRUE`.

Gli algoritmi di creazione e di ricostruzione di un frame sicuro sono descritti dal seguente pseudo-codice [2]:

Algorithm 1 Outgoing frame security procedure

```

Require: Status = SUCCESS
if macSecurityEnabled = false then
    status ← UNSUPPORTED_SECURITY
    EXIT
end if
CalculateAuthLen(SecurityLevel)
CalculateAuxLen(KeyIdMode, SecurityLevel)
data_expansion = AuxLen + AuthLen
if FrameHLen + data_expansion + FCS
aMaxPHYPacketSize then
    status ← FRAME_TOO_LONG
    EXIT
end if
if macFrameCounter = 0xffffffff then
    status ← COUNTER_ERROR
    EXIT
end if
if status = SUCCESS then
    InsertAuxSecHeader(Frame, AuxSecHeader)
    CCMStarTransf(Frame, AuxSecHeader)
    macFrameCounter ++
    SendSecFrame(SecFrame)
end if

```

Algorithm 2 Incoming frame security procedure

```

if macSecurityEnabled = true and SecurityLevel = 0 then
    status ← UNSUPPORTED_SECURITY
    EXIT
end if
if macFrameCounter = 0xffffffff then
    status ← COUNTER_ERROR
    EXIT
end if
if (CCMStarInverseTransf(SecFrame)) then
    macFrameCounter ++
    status = SUCCESS
    AcceptFrame
else
    RejectFrame
end if

```

Rendere sicuro un frame implica l'utilizzo di tecniche di crittografia e autenticazione in modalità **CCM** (*counter with cipher block chaining message authentication code*). Ricostruire un frame sicuro implica l'uso della trasformazione di decrittazione CCM e di controllo dell'autenticazione. La lunghezza *M* del campo Authentication per la trasformazione CCM forward e la trasformazione CCM inverse è determinata usando il campo **Security Level** del campo **Security Control** all'interno dell'**ASH** (*Auxiliary Security Header*).

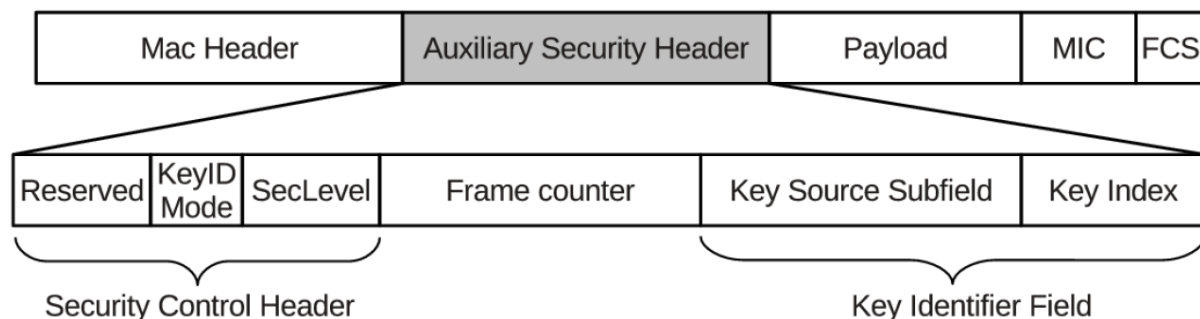
Security level	<i>a</i> data	<i>m</i> data
0	None	None
1	MHR Open Payload field Unsecured Private Payload field	None
2	MHR Open Payload field Unsecured Private Payload field	None
3	MHR Open Payload field Unsecured Private Payload field	None
4	None	Unsecured Private Payload field
5	MHR Open Payload field	Unsecured Private Payload field
6	MHR Open Payload field	Unsecured Private Payload field
7	MHR Open Payload field	Unsecured Private Payload field

Nel processo di trasformazione CCM, i campi di dati che vengono utilizzati rappresentano stringhe di ottetti. Il campo Private Payload del frame originale non protetto è sostituito dalla concatenazione a destra di quel campo e dal campo '**C**' se non è prevista la riservatezza dei dati, altrimenti è solo

sostituito dal campo 'C'. Il contenuto dei dati 'C' per ciascuno dei livelli di sicurezza è definito nella seguente tabella:

Security level	c data
0	None
1	MIC-32
2	MIC-64
3	MIC-128
4	Encrypted Private Payload field
5	Encrypted Private Payload field MIC-32
6	Encrypted Private Payload field MIC-64
7	Encrypted Private Payload field MIC-128

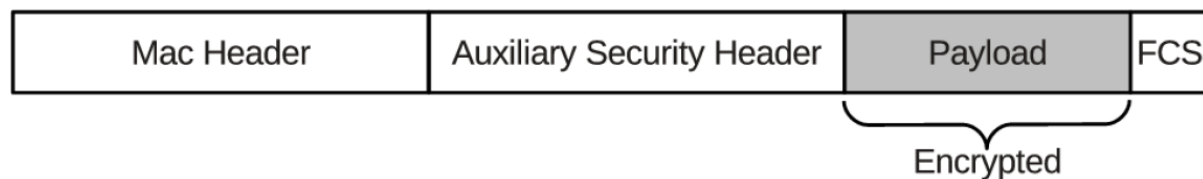
Tutte le informazioni riguardanti la sicurezza del pacchetto sono contenute nel campo **Auxiliary Security Header**. Questo ha una lunghezza variabile e contiene informazioni necessarie per l'elaborazione della sicurezza, compresi un campo Security Control, un campo Frame Counter (che realizza la protezione da replay attack) e un campo Key Identifier. Il campo Auxiliary Security Header è presente solo se il campo Security Enabled è impostato a uno. Il campo Auxiliary Security Header è formattato come segue:



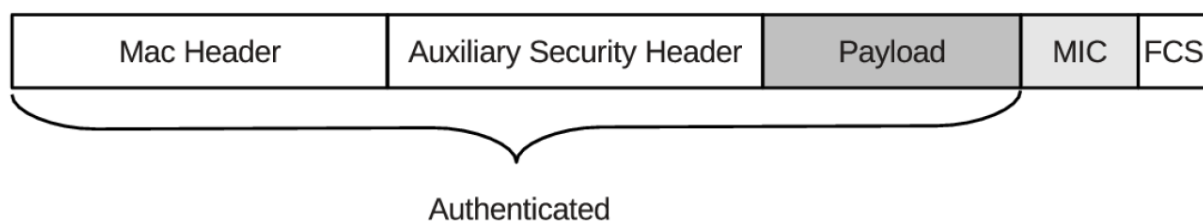
Il Security Level descrive l'effettiva protezione del frame che viene fornita descrivendone i vari livelli di autenticità dei dati (per permettere la minimizzazione dell'overhead di sicurezza nei frame trasmessi dove richiesto) e per la riservatezza opzionale dei dati. La protezione crittografica offerta dai vari livelli di sicurezza è mostrata nella tabella successiva.

Security level	Security level field $b_2 b_1 b_0$	Security attributes	Data confidentiality	Data authenticity	Encrypted authentication tag length, M , (octets)
0	000	None	OFF	NO	0
1	001	MIC-32	OFF	YES	4
2	010	MIC-64	OFF	YES	8
3	011	MIC-128	OFF	YES	16
4	100	ENC	ON	NO	0
5	101	ENC-MIC-32	ON	YES	4
6	110	ENC-MIC-64	ON	YES	8
7	111	ENC-MIC-128	ON	YES	16

Il livello di sicurezza **CTR** protegge un frame criptando il suo payload in counter mode. Come regola generale, il livello di sicurezza CTR richiede un'operazione di cifratura a blocchi per ogni blocco da cifrare [3].

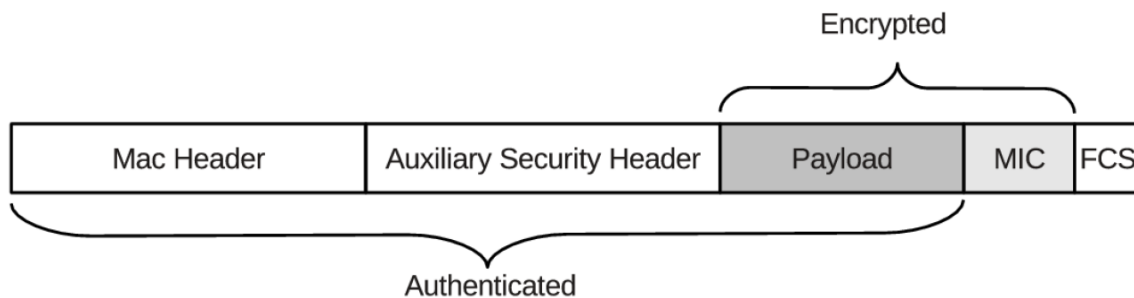


Il livello di sicurezza **CBC-MAC** protegge un frame autenticando l'intestazione del frame, l'auxiliary Security Header ASH, ed il payload. Il CBC-MAC inizialmente calcola un Message Integrity Code (MIC) a 128 bit usando il cifrario a blocchi AES in modalità cipher-block-chaining. Successivamente, il MIC viene troncato e aggiunto al frame. Il MIC può essere troncato a 4, 8 o 16 byte, portando così a tre varianti di CBC-MAC di sicurezza crescente, cioè rispettivamente *CBC-MAC-4*, *CBC-MAC-8*, e *CBC-MAC-16*. Come regola generale, il livello di sicurezza CBC-MAC richiede un'operazione di cifratura a blocchi per ogni blocco da autenticare.



Infine, il livello di sicurezza **CCM** assicura un frame usando il cifrario a blocchi AES nel counter with CBC-MAC mode. Il livello di sicurezza CCM inizialmente autentica l'intestazione del frame, l'ASH e il payload come nel livello di sicurezza CBC-MAC. Come il livello di sicurezza CBC-MAC, il MIC può essere troncato a 4, 8, o 16 byte producendo così tre varianti del CCM di sicurezza crescente, cioè rispettivamente *CCM-4*, *CCM-8*, e *CCM-16*. Infine, il livello di sicurezza CCM cripta il MIC risultante e il payload in counter mode. Come regola generale, il livello di sicurezza CCM richiede un'operazione di

cifratura a blocchi per ogni blocco di campi cifrati o autenticati (cioè intestazione del frame, ASH e MIC) e due operazioni di cifratura per il payload, che è sia autenticato che cifrato.

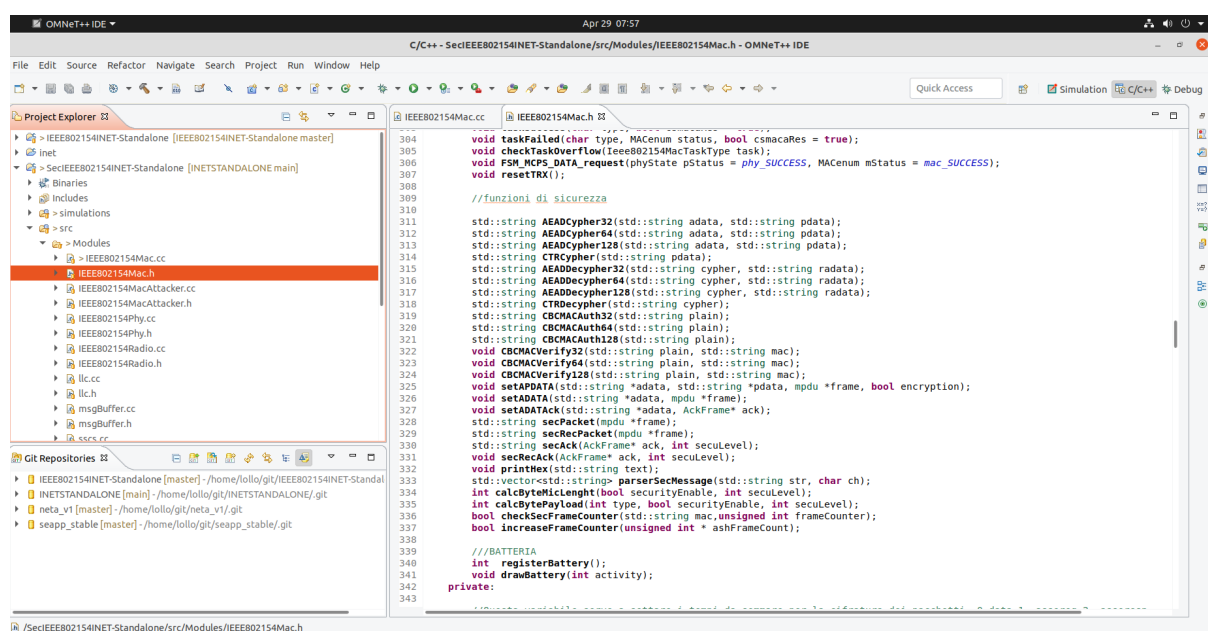


Il campo payload è differente e varia in base alla tipologia del pacchetto; tali differenze vengono spiegate accuratamente nelle specifiche del protocollo [1].

Come discusso nella parte introduttiva della relazione, l'obiettivo del progetto è quello di implementare tutte le specifiche di sicurezza del protocollo che sono state precedentemente descritte. È stato necessario l'utilizzo di diversi tools per realizzare in maniera simulata l'architettura necessaria al raggiungimento degli obiettivi progettuali.

Di seguito vengono menzionati tali tools:

- **Simulatore: Omnet++**
 - <https://omnetpp.org/>
- **Framework: IEEE802154INET-Standalone**
 - <https://github.com/michaelkirsche/IEEE802154INET-Standalone>
- **Libreria crittografica: Crypto++**
 - <https://cryptopp.com/>

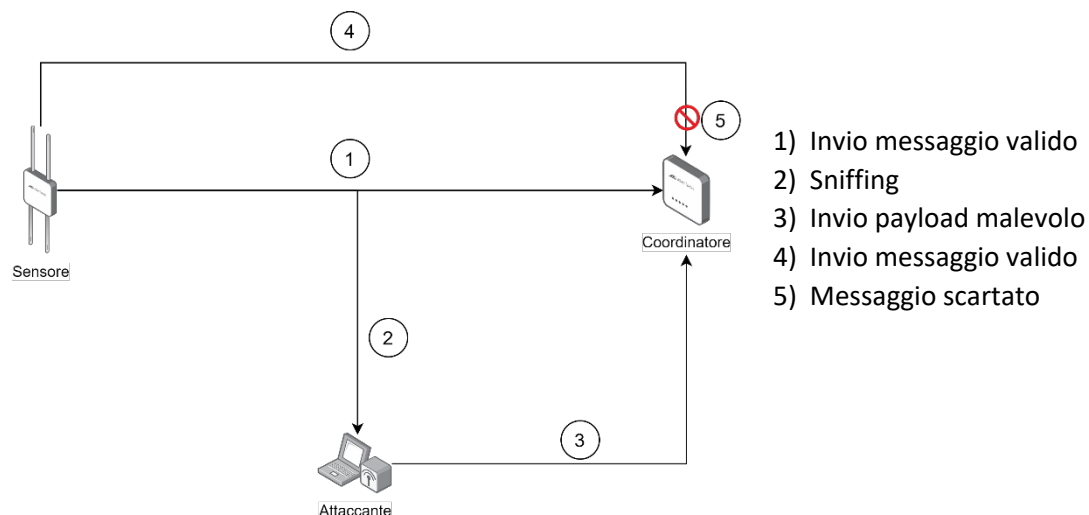


4 VULNERABILITÀ

La suite AES-CTR usa la counter mode senza un MAC. Lo standard non impone ai progettisti di supportare la suite in modalità CTR; solo AES-CCM-64 è obbligatorio. Tuttavia, dagli studi effettuati, si è notato che AES-CTR risulta essere pericoloso e non dovrebbe mai essere abilitato o implementato. La crittografia non autenticata introduce un rischio significativo di vulnerabilità a livello di protocollo. I ricercatori hanno trovato diverse vulnerabilità nei protocolli che usano la crittografia protetta solo da un CRC senza un codice di autenticazione del messaggio crittograficamente forte. Tutti gli attacchi sono incentrati sul fatto che nel corso della modifica del testo cifrato, l'avversario può costruire modifiche appropriate al CRC in modo che il ricevitore accetti il pacchetto. I ricercatori hanno scoperto vulnerabilità di crittografia non autenticata in **IPSec** [4], **802.11** [5] e **SSH versione 1** [6] che compromettono non solo l'integrità ma anche la riservatezza. Qualsiasi applicazione che utilizza la suite di sicurezza AES-CTR diventa vulnerabile ad attacchi simili. Come si è notato, gli sviluppatori hanno la tendenza ad assumere che *"dato che la decifratura con la chiave sbagliata produrrà spazzatura, un ulteriore controllo di integrità non è necessario"*, ma questa assunzione non solo è imprecisa, ma porta con sé notevoli rischi. La radice del problema è che, in molti sistemi, la capacità di falsificare messaggi non autentici spesso permette ad un attaccante di ingannare un endpoint per rivelare dei segreti. La gravità di questo problema dipende dai dettagli del protocollo specifico dell'applicazione, quindi è impossibile fare qualsiasi considerazione che si applichi a tutte le implementazioni. Tuttavia, gli studi hanno dimostrato ripetutamente che l'uso della crittografia senza un MAC pone un rischio significativo di violazioni della sicurezza. Pertanto, questa modalità non deve mai essere utilizzata in quanto intrinsecamente vulnerabile. A seguito di queste considerazioni, sono stati individuati due attacchi potenzialmente critici, ovvero il **"Replay Protection Attack"** e **"No Integrity on Acknowledgment Packets"** [2].

4.1 REPLAY PROTECTION ATTACK

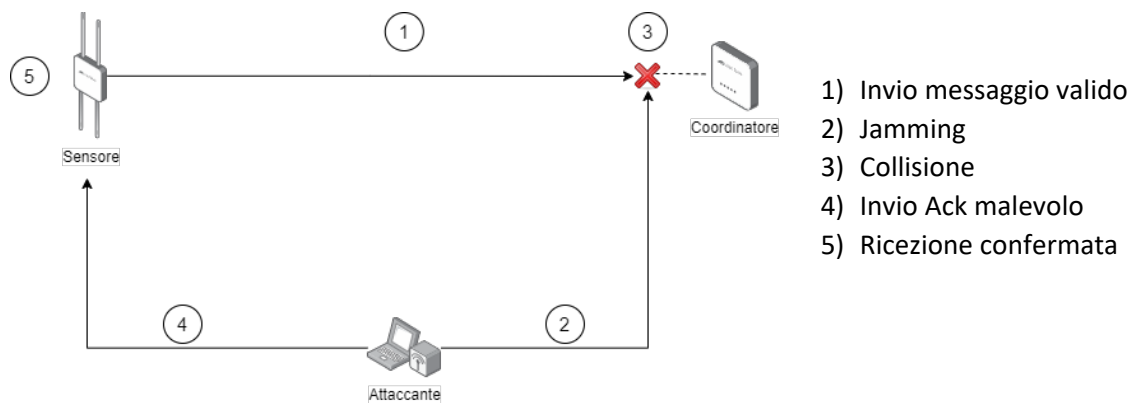
Il **Replay Protection Attack** è un attacco a singolo pacchetto che è applicabile quando una rete 802.15.4 usa la suite AES-CTR con la protezione di replay abilitata. Si suppone che un mittente **S** e un destinatario **D** comunichino con la suite AES-CTR usando la chiave **K**, e che il destinatario abbia abilitato la protezione da replay. Si ricorda che **D** mantiene un high water mark composto dalla chiave e dal frame counter, rifiutando i pacchetti il cui contatore è più piccolo dell'high water mark. Si considera cosa succede quando un avversario invia un pacchetto falsificato con indirizzo sorgente **S**, contatore di chiavi **0xFF**, Frame counter **0xFFFFFFFF**, e qualsiasi payload (non necessariamente un testo cifrato valido sotto la chiave **K**). Il destinatario decifrerà il pacchetto sotto la chiave **K**, ottenendo un testo spazzatura. Poiché non c'è controllo di accesso o autenticazione del messaggio, il destinatario accetterà il pacchetto anche se contiene spazzatura. Tuttavia, prima di passare il payload all'applicazione, il livello di controllo dell'accesso ai media aggiornerà l'high water mark a **0xFFFFFFFF**. La prossima volta che il vero mittente **S** cerca di inviare un pacchetto legittimo, il destinatario lo rifiuterà indipendentemente da ciò che **S** fa, perché l'high water mark ha raggiunto il suo valore massimo e qualsiasi pacchetto successivo sembrerà essere replay.



Attacchi simili possono anche essere usati per impedire la consegna dei prossimi N pacchetti da inviare su qualche link, dove il numero N può essere scelto dall'attaccante. Questo dimostra che un attaccante può interrompere permanentemente un link 802.15.4, se tale link usa AES-CTR con la protezione da replay abilitata. L'attacco è facile da montare, perché richiede solo l'invio di un singolo pacchetto contraffatto; l'attaccante non ha bisogno di alcun accesso o attrezzatura speciale.

4.2 NO INTEGRITY ON ACKNOWLEDGMENT PACKETS

La specifica 802.15.4 non include alcuna protezione di integrità o di riservatezza per i pacchetti di **acknowledgment**. Quando si invia un pacchetto, il mittente ha la possibilità di richiedere una conferma da parte del destinatario impostando un bit nel campo flag. Se il flag dell'acknowledgment è impostato, il destinatario restituisce un pacchetto di acknowledgment che contiene il numero di sequenza del pacchetto. L'access control layer del mittente ritrasmette il pacchetto un certo numero di volte se non riceve il pacchetto di acknowledgment in tempo. L'applicazione che invia viene avvisata quando l'acknowledgment arriva. Tuttavia, la mancanza di un MAC all'interno dell'acknowledgment permette ad un avversario di falsificare un Ack per qualsiasi pacchetto. Un avversario ha solo bisogno di creare un acknowledgment falsificato con il numero di sequenza appropriato dal pacchetto originale; questo non è difficile, poiché questo numero di sequenza è inviato in chiaro. Questa debolezza può essere combinata con un **jamming** mirato per impedire la consegna di pacchetti selezionati. Si supponga che un attaccante identifichi un pacchetto che non vuole arrivi al destinatario previsto. L'attaccante può trasmettere una breve raffica di interferenze mentre il pacchetto viene inviato, causando l'invalidità del CRC al destinatario e lo scarto del pacchetto da parte del destinatario. Poi, l'attaccante può falsificare un Ack che sembra valido, ingannando il mittente nel pensare che il pacchetto sia stato ricevuto. Questa vulnerabilità rende le conferme inaffidabili in presenza di avversari. Per esempio, supponiamo che un'applicazione usi gli acknowledgment come un meccanismo di invio affidabile. L'applicazione continua a tentare di inviare il pacchetto fino a quando rimane non ricevuto. Se l'applicazione di invio riceve un Ack, non può mai essere sicura che i dati siano effettivamente arrivati a destinazione. La conferma potrebbe essere legittima o potrebbe essere falsificata. Di conseguenza, non è possibile utilizzare questa tipologia di pacchetti per finalità di sicurezza o affidabilità, ma solo per le performance.



Nel capitolo successivo verranno proposte delle soluzioni che implicano una modifica dello standard ma, se questo non fosse possibile per il progettista, si consiglia almeno di seguire queste due raccomandazioni:

- *Non usare la suite di sicurezza AES-CTR*: questo perché, come si è notato, l'AES-CTR può portare con sé vulnerabilità note. L'incapacità di supportare correttamente il rilevamento di replay (nonostante le specifiche affermino il contrario) è solo un esempio. Non si raccomanda a nessuna applicazione di usare la suite di sicurezza AES-CTR.
- *Non fare affidamento sull'acknowledgment*: i progettisti di applicazioni non dovrebbero fare affidamento sugli acknowledgment così come sono, poiché non forniscono alcuna garanzia. Ricevere un Ack non significa che il destinatario abbia effettivamente ricevuto il pacchetto. Se un'applicazione avesse bisogno della funzionalità di acknowledgment, dovrebbe usarli a livello di applicazione inviando pacchetti di dati con controllo di integrità e non usarli come sono forniti dall'802.15.4. Gli acknowledgment a livello di applicazione duplicano molte funzionalità che l'access control layer fornisce ai media. Per esempio, un'applicazione avrebbe bisogno di usare il proprio meccanismo per ritentare un invio finché non arriva un Ack a livello di applicazione. C'è certamente una complessità aggiunta in questo workaround, ma a parte la protezione dell'integrità per gli acknowledgment, crediamo che sia l'unica alternativa sicura.

5 MODIFICHE PROPOSTE

In questa sezione, vengono proposti alcuni miglioramenti alla sicurezza per prevenire gli attacchi descritti nel capitolo 4. Per andare a correggere alcuni comportamenti vulnerabili nel protocollo sono state prese in considerazione diverse soluzioni che verranno discusse in questo capitolo. All'interno del simulatore ne sono state implementate solo alcune, ovvero quelle ritenute più adatte.

5.1 SOLUZIONI DISPONIBILI

Le soluzioni individuate sono quattro e sono caratterizzate dalla necessità di rendere più robusto il protocollo soffermandosi su diversi punti critici:

- Rendere più robusto il meccanismo di replay protection
- Legare il frame counter ad altri meccanismi più sicuri
- Autenticare laddove ce ne sia la necessità

Di seguito vengono riportate nel dettaglio le soluzioni individuate.

5.1.1 Timestamp come Frame counter

Sia il replay protection attack che il denial of service sono legati al meccanismo di frame counter implementato per garantire la protezione da attacchi di replay. Il frame counter viene utilizzato come contatore per evitare che, all'interno della rete, si possa andare a gestire due volte lo stesso pacchetto (replay-packet). Inoltre, il frame counter causa potenziali problemi quando i nodi sono in modalità sleep o l'alimentazione dei nodi è temporaneamente esaurita in quanto ne consegue un reset dei valori necessari al protocollo, tra cui il frame-counter, e questo causa un rigetto dei pacchetti validi da parte del coordinatore. Una soluzione a ciò può essere l'utilizzo del **timestamp**. La resilienza al replay attack si ottiene facendo in modo che il ricevitore controlli il timestamp recente ottenuto dal mittente e rifiuti il frame che ha il timestamp uguale o inferiore al timestamp ottenuto in precedenza. Inoltre, non c'è un contatore da avanzare. Lo svantaggio di questo approccio è che la lunghezza del campo risulta essere maggiore. Poiché all'interno della specifica IEEE 802.15.4 vengono definiti i beacon frame anche per supportare la sincronizzazione del tempo sui dispositivi, l'uso del timestamp può prevenire l'attacco di replay-protection come segue. Ogni volta che il mittente riceve un frame con un timestamp, esso lo confronta con il tempo corrente. Se il tempo corrente è molto più piccolo del timestamp (secondo una certa soglia), il mittente comprende di essere sotto attacco, e rifiuta il frame. Pertanto, il valore del timestamp registrato non può mai essere tale da poter lanciare un attacco di replay-protection o un attacco denial of service. Inoltre, quando un sensore si sveglia o ottiene l'alimentazione dopo un'interruzione di corrente, contatta il coordinatore, sincronizza il tempo con i frame beacon ricevuti, e alza tutti i timestamp fino al tempo corrente. In questo modo, sia l'attacco di replay-protezione che l'attacco denial of service possono essere evitati.

5.1.2 Autenticazione ACK

Per il frame ACK, si propone di aggiungere MIC alla fine del frame ACK, dove MIC è ottenuto dall'algoritmo di autenticazione AES-CBC-MAC. Il campo autenticato è l'intero frame ACK e la grandezza del MIC segue le stesse regole dettate dal protocollo per gli altri pacchetti.

5.1.3 Gestione proattiva del coordinatore

Per preservare la rete dal replay protection attack, ogni nodo dovrebbe tenere traccia del frame counter per ogni dispositivo con cui comunica. Oltre questo il coordinatore dovrebbe avere un approccio proattivo in cui chiede ai dispositivi registrati sulla rete (a slot di tempo prefissati) il proprio frameCounter. Tuttavia, questo schema potrebbe non essere molto robusto quando c'è un guasto come un'interruzione di corrente o un riavvio. Inoltre, è un po' scomodo per mantenere la coerenza.

5.1.4 Modalità CCM*

Altra soluzione è quella di prevedere la sola modalità **CCM***, in cui un contatore determinato dal frame counter del dispositivo sorgente viene utilizzato per fornire la “freschezza” del frame e per prevenire il replay-protection attack. Per ogni nodo a cui un dispositivo invia o riceve frame protetti, viene creata una voce **ACL** nel MAC PIB, contenente l'indirizzo implicito o esplicito dell'entità e i corrispondenti valori di sicurezza associati, compresa una chiave AES, un frame counter per i frame in uscita, e un frame counter esterno per quelli in entrata. La chiave simmetrica AES è di 16 byte per proteggere i frame in entrata e in uscita; il frame counter in uscita è usato da un dispositivo quando origina un frame; e il frame counter esterno per i frame in entrata è usato da un dispositivo per verificare la validità dei frame in entrata. Questo contatore viene incrementato ogni volta che viene trasmesso un frame sicuro, ma non è previsto che torni al valore precedente proprio per assicurare che il **CCM* nonce** sia unico e per garantire la validità o per rilevare i duplicati. La suite di sicurezza IEEE 802.15.4 include tre componenti, l'AES-CCM per la crittografia e l'autenticazione, l'AES-CBC-MAC per la sola autenticazione e l'AES-CTR per la sola crittografia. Questi tre componenti separati richiedono un'implementazione più corposa rispetto alla sola implementazione CCM* unificata che permetterebbe l'utilizzo di tutte le altre primitive crittografiche unificandole in una sola modalità di utilizzo; passare da una modalità all'altra compromette la sicurezza a meno che non si mantengano chiavi separate, ma richiede una memorizzazione aggiuntiva; e il CBC-MAC non fornisce “freschezza” dei frame oltre ad essere soggetto ad attacchi di replay. Pertanto, la sostituzione della suite di sicurezza, l'AES-CCM con l'**AES-CCM***, comprometterebbe la compatibilità con le precedenti versioni del protocollo in quanto dovrebbe essere riconsiderato l'approccio di negoziazione della sicurezza (uso e generazione delle chiavi crittografiche) così come la modalità senza sicurezza. Questa modifica comprometterebbe la retrocompatibilità del protocollo ma risolverebbe, in maniera compatta ed efficace, le diverse vulnerabilità descritte precedentemente.

5.2 SOLUZIONI ADOTTATE

Sono state analizzate diverse soluzioni per poter utilizzare il protocollo in maniera sicura e la scelta ritenuta più corretta è stata quella di ottenere un compromesso. Pertanto, sono state ritenute inadatte soluzioni come quella del timestamp o onlyCCM* rispettivamente per la troppa semplicità e per l'eccessivo compromesso (causato dalla perdita di retrocompatibilità) che renderebbe la scelta troppo drastica. Sostanzialmente si è deciso di scegliere una via sicura ma senza stravolgere in pieno il protocollo andando ad implementare due delle 4 soluzioni proposte con determinate accortezze. Di seguito vengono riportate le modifiche apportate:

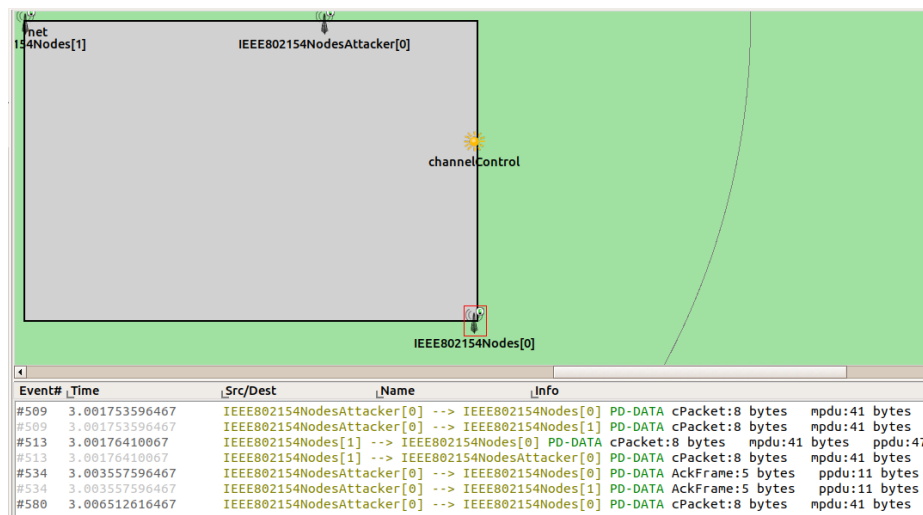
5.2.1 Autenticazione dell'ACK

Si è scelto di implementare questa soluzione in quanto relativamente semplice dal punto di vista implementativo. Andando a prevedere anche un MIC per l'ACK esso non può essere generato a piacimento da un attaccante. Per evidenziare il problema e la relativa soluzione è stato implementato un attacco (su protocollo non modificato) che evidenzia come sia relativamente facile causare una collisione e far credere al nodo mittente che il pacchetto sia arrivato a destinazione andando a generare un ACK fatto ad-hoc che rende non più affidabile la ricezione del pacchetto per confermare la ricezione del messaggio da parte del ricevente. Tale soluzione è stata integrata all'interno dei livelli di sicurezza già presenti mantenendo la coerenza delle scelte di sicurezza adottate dal protocollo. Per quanto riguarda i livelli in cui viene prevista anche la cifratura del payload, oltre all'autenticazione dell'intero messaggio, la soluzione prevede che il campo payload sia vuoto e quindi non vi è nessuna cifratura, in questo caso la robustezza del messaggio non è data dall'impossibilità di visualizzare il sequence number (in quanto viaggia in chiaro) ma è dovuta alla segretezza della chiave e quindi l'impossibilità, da parte di un attaccante, di generare un MIC valido per un pacchetto forgiato ad-hoc.

```
8163 std::string IEEE802154Mac::secAck(AckFrame* ack, int secuLevel)
8164 {
8165     std::string adata;
8166     std::string result;
8167     switch (secuLevel)
8168     {
8169     case 1:
8170         setADATack(&adata, ack);
8171         result = CBCMACAuth32(adata);
8172         break;
8173     case 2:
8174         setADATack(&adata, ack);
8175         result = CBCMACAuth64(adata);
8176         break;
8177     case 3:
8178         setADATack(&adata, ack);
8179         result = CBCMACAuth128(adata);
8180         break;
8181     case 4:
8182         break;
8183     case 5:
8184         setADATack(&adata, ack);
8185         result = CBCMACAuth32(adata);
8186         break;
8187     case 6:
8188         setADATack(&adata, ack);
8189         result = CBCMACAuth64(adata);
8190         break;
8191     case 7:
8192         setADATack(&adata, ack);
8193         result = CBCMACAuth128(adata);
8194         break;
8195     default:
8196         std::cout << "secuLevel : " << secuLevel << endl;
8197         throw cRuntimeError("secuLevel error");
8198     }
8199 }
8200

8206 void IEEE802154Mac::secRecAck(AckFrame* ack, int secuLevel)
8207 {
8208     std::string radata;
8209     std::string decipherT;
8210     std::stringstream temp;
8211     HexDecoder decoder(new FileSink(temp));
8212     switch (secuLevel)
8213     {
8214     case 1:
8215         setADATack(&radata, ack);
8216         decoder.Put((const byte *) ack->getMic(), 8);
8217         CBCMACVerify32(radata, temp.str());
8218         break;
8219     case 2:
8220         setADATack(&radata, ack);
8221         decoder.Put((const byte *) ack->getMic(), 16);
8222         CBCMACVerify64(radata, temp.str());
8223         break;
8224     case 3:
8225         setADATack(&radata, ack);
8226         decoder.Put((const byte *) ack->getMic(), 32);
8227         CBCMACVerify128(radata, temp.str());
8228         break;
8229     case 4:
8230         break;
8231     case 5:
8232         setADATack(&radata, ack);
8233         decoder.Put((const byte *) ack->getMic(), 8);
8234         CBCMACVerify32(radata, temp.str());
8235         break;
8236     case 6:
8237         setADATack(&radata, ack);
8238         decoder.Put((const byte *) ack->getMic(), 16);
8239         CBCMACVerify64(radata, temp.str());
8240         break;
8241     case 7:
8242         setADATack(&radata, ack);
8243         decoder.Put((const byte *) ack->getMic(), 32);
8244         break;
8245     }
```

Di seguito viene mostrato l'attacco performato sul simulatore omnet++:



Gli eventi **509** e **513** realizzano la collisione da parte dell'attaccante (Jamming) che riesce a corrompere l'invio del pacchetto valido generato dal nodo_1. Sfruttando questa collisione l'attaccante crea un Ack con il medesimo sequence number inoltrandolo alla rete e riuscendo ad ingannare il nodo_1 dell'avvenuta ricezione da parte del nodo coordinatore.

```

● PD-DATA (AirFrame)
├── controlInfo = NULL (cObject)
├── encapsulatedPacket = (ppdu) PD-DATA (cPacket)
│   ├── controlInfo = NULL (cObject)
│   ├── encapsulatedPacket = (mpdu) DATA.indication
│   │   ├── controlInfo = NULL (cObject)
│   │   ├── encapsulatedPacket = (cPacket) (cPacket)
│   │   │   ├── fcs = 0 [...] (unsigned short)
│   │   │   ├── ash (Ash)
│   │   │   ├── src = 0A:AA:00:00:00:00:01 (MACAddressExt)
│   │   │   ├── srcPANid = 0 [...] (unsigned short)
│   │   │   ├── dest = 0A:AA:00:00:00:00:00 (MACAddressExt)
│   │   │   ├── destPANid = 0 [...] (unsigned short)
│   │   │   ├── sqnr = 128 [...] (unsigned char)
│   │   │   ├── fcf = 9271 [...] (unsigned short)
│   │   │   ├── isGTS = false [...] (bool)
│   │   │   ├── isIndirect = false [...] (bool)
│   │   │   ├── payload = '187ABC9EE4FF85D19' [...] (string)
│   │   │   └── mic = " [...] (string)
│   │   ├── base
│   │   ├── message
│   │   ├── packet
│   │   └── sending
│   ├── Preamble[4] (unsigned char)
│   ├── SFD = 229 [...] (unsigned char)
│   ├── PHR = 41 [...] (unsigned char)
│   └── base

```

```

● PD-DATA (AirFrame)
├── controlInfo = NULL (cObject)
├── encapsulatedPacket = (ppdu) PD-DATA (cPacket)
│   ├── controlInfo = NULL (cObject)
│   ├── encapsulatedPacket = (AckFrame) Acknowledg
│   │   ├── controlInfo = NULL (cObject)
│   │   ├── encapsulatedPacket = NULL (cPacket)
│   │   │   ├── fcs = 0 [...] (unsigned short)
│   │   │   ├── sqnr = 128 [...] (unsigned char)
│   │   │   ├── fcf = 16384 [...] (unsigned short)
│   │   │   ├── Mic = " [...] (string)
│   │   ├── base
│   │   ├── message
│   │   ├── packet
│   │   └── sending
│   ├── Preamble[4] (unsigned char)
│   ├── SFD = 229 [...] (unsigned char)
│   ├── PHR = 5 [...] (unsigned char)
│   ├── base
│   ├── message
│   ├── packet
│   └── sending
│   ├── pSend = 1.000000 [...] (double)
│   ├── channelNumber = 25 [...] (int)
│   ├── duration = 0.000352 [...] (simtime_t)
│   ├── bitrate = 250000.000000 [...] (double)
│   └── snr = 0.000000 [...] (double)

```

```

● PD-DATA (AirFrame)
├── controlInfo = NULL (cObject)
├── encapsulatedPacket = (ppdu) PD-DATA (cPacket)
│   ├── controlInfo = NULL (cObject)
│   ├── encapsulatedPacket = (mpdu) DATA.indication
│   │   ├── controlInfo = NULL (cObject)
│   │   ├── encapsulatedPacket = (cPacket) (cPacket)
│   │   │   ├── fcs = 0 [...] (unsigned short)
│   │   │   ├── ash (Ash)
│   │   │   ├── src = 0A:AA:00:00:00:00:01 (MACAddressExt)
│   │   │   ├── srcPANid = 0 [...] (unsigned short)
│   │   │   ├── dest = 0A:AA:00:00:00:00:00 (MACAddressExt)
│   │   │   ├── destPANid = 0 [...] (unsigned short)
│   │   │   ├── sqnr = 129 [...] (unsigned char)
│   │   │   ├── fcf = 9271 [...] (unsigned short)
│   │   │   ├── isGTS = false [...] (bool)
│   │   │   ├── isIndirect = false [...] (bool)
│   │   │   ├── payload = '187ABC9EE4FF85D18' [...] (string)
│   │   │   └── mic = " [...] (string)
│   │   ├── base
│   │   ├── message
│   │   ├── packet
│   │   └── sending
│   ├── Preamble[4] (unsigned char)
│   ├── SFD = 229 [...] (unsigned char)
│   ├── PHR = 41 [...] (unsigned char)
│   └── base

```

Dal contenuto dei pacchetti si può notare come l'attaccante riesce a sfruttare questa vulnerabilità in quanto l'ultimo pacchetto che invia avrà un sequence number maggiore e non prova a ritrasmettere il pacchetto precedente in quanto l'attaccante riesce a convincere il nodo_1 dell'avvenuta ricezione.

Utilizzando la soluzione proposta notiamo come la vulnerabilità non risulta essere più sfruttabile.

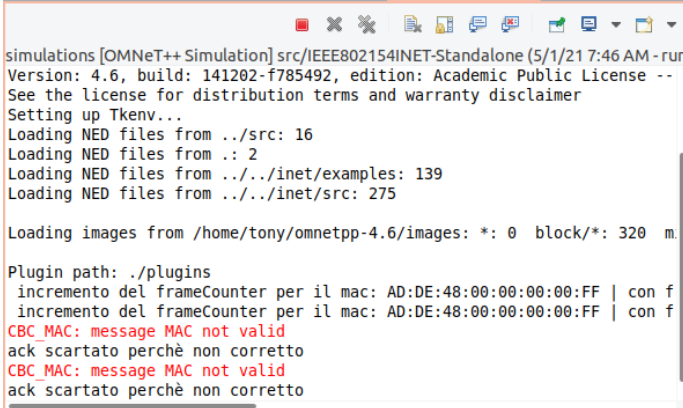
```

● (AirFrame) PD-DATA
Fields  Contents (1)
├── PD-DATA (AirFrame)
│   ├── controlInfo = NULL (cObject)
│   ├── encapsulatedPacket = (ppdu) PD-DATA (cPacket)
│   │   ├── controlInfo = NULL (cObject)
│   │   ├── encapsulatedPacket = (mpdu) DATA.indication
│   │   │   ├── controlInfo = NULL (cObject)
│   │   │   ├── encapsulatedPacket = (cPacket) (cPacket)
│   │   │   │   ├── fcs = 0 [...] (unsigned short)
│   │   │   │   ├── ash (Ash)
│   │   │   │   ├── src = 0A:AA:00:00:00:00:01 (MACAddressExt)
│   │   │   │   ├── srcPANid = 0 [...] (unsigned short)
│   │   │   │   ├── dest = 0A:AA:00:00:00:00:00 (MACAddressExt)
│   │   │   │   ├── destPANid = 0 [...] (unsigned short)
│   │   │   │   ├── sqnr = 128 [...] (unsigned char)
│   │   │   │   ├── fcf = 9271 [...] (unsigned short)
│   │   │   │   ├── isGTS = false [...] (bool)
│   │   │   │   ├── isIndirect = false [...] (bool)
│   │   │   │   ├── payload = 'A2E353CEB0E65D7083983B70E88B0' [...] (string)
│   │   │   │   └── mic = " [...] (string)
│   │   │   ├── base
│   │   │   ├── message
│   │   │   ├── packet
│   │   │   └── sending
│   ├── Preamble[4] (unsigned char)
│   ├── SFD = 229 [...] (unsigned char)
│   ├── PHR = 41 [...] (unsigned char)
│   └── base

```

Event#	Time	Src/Dest	Name	Info
#509	3.001753596467	IEEE802154NodesAttacker[0] --> IEEE802154Nodes[0]	PD-DATA	cPacket:8 bytes mpc
#509	3.001753596467	IEEE802154NodesAttacker[0] --> IEEE802154Nodes[1]	PD-DATA	cPacket:8 bytes mpc
#513	3.00176410067	IEEE802154Nodes[1] --> IEEE802154NodesAttacker[0]	PD-DATA	cPacket:8 bytes mpc
#513	3.00176410067	IEEE802154Nodes[1] --> IEEE802154NodesAttacker[0]	PD-DATA	cPacket:8 bytes mpc
#534	3.003557596467	IEEE802154NodesAttacker[0] --> IEEE802154Nodes[1]	PD-DATA	AckFrame:13 bytes mpc
#534	3.003557596467	IEEE802154NodesAttacker[0] --> IEEE802154Nodes[1]	PD-DATA	AckFrame:13 bytes mpc
#574	3.00528410067	IEEE802154Nodes[1] --> IEEE802154Nodes[0]	PD-DATA	cPacket:8 bytes mpc

In questo caso si nota come il nodo_1 prova a ritrasmettere il pacchetto iniziale, questo perché l'Ack inviata dall'attaccante sarà ignorata in quanto la verifica del MIC non andrà a buon fine.



```
simulations [OMNeT++ Simulation] src/IEEE802154INET-Standalone (5/1/21 7:46 AM - run
Version: 4.6, build: 141202-f785492, edition: Academic Public License --
See the license for distribution terms and warranty disclaimer
Setting up Tkenv...
Loading NED files from ../src: 16
Loading NED files from .: 2
Loading NED files from ../../inet/examples: 139
Loading NED files from ../../inet/src: 275

Loading images from /home/tony/omnetpp-4.6/images: *: 0 block/*: 320 m.

Plugin path: ./plugins
incremento del frameCounter per il mac: AD:DE:48:00:00:00:FF | con f
incremento del frameCounter per il mac: AD:DE:48:00:00:00:FF | con f
CBC_MAC: message MAC not valid
ack scartato perchè non corretto
CBC_MAC: message MAC not valid
ack scartato perchè non corretto
```

5.2.2 Deprecazione livello 4 AES-CTR

Questa soluzione prevede l'eliminazione del livello 4 previsto dallo standard IEEE 802.15.4 perché intrinsecamente insicuro. Questa scelta è obbligatoria in quanto ulteriori meccanismi di sicurezza per ovviare al problema risulterebbero solo un'aggiunta computazionale ad un protocollo che fa dell'efficienza e della semplicità il suo punto di forza. Non è la prima volta che viene identificata una primitiva crittografica inadeguata e la soluzione, storicamente più corretta, è sempre stata quella di deprecarne l'utilizzo (DES etc..). Questa soluzione, a dispetto dell'utilizzo della sola modalità CCM*, risulta essere meno catastrofica in quanto permette di non stravolgere il protocollo, obbligando a ridisegnare buona parte della gestione di sicurezza, ma semplicemente di apportare modifiche a delle funzionalità ritenute dannose per il protocollo stesso e per i suoi obiettivi al fine di renderlo più robusto. L'eliminazione dell'AES-CTR non sconvolge la gestione delle primitive di sicurezza già descritte nel capitolo apposito ma prevede solo la gestione differente del livello 4. In questa soluzione si è deciso di accorparlo al livello 0 ovvero quello senza sicurezza ma diverse scelte possono essere condivisibili e consigliate, per esempio quella di prevedere un'altra tipologia di cifratura (in modalità CCM) pur rimanendo coerente con la gestione prevista dal protocollo (AES a curve ellittiche). Di seguito viene riportata la simulazione che evidenzia la vulnerabilità al *replay protection attack* con l'utilizzo di tale livello di sicurezza e successivamente la simulazione che evidenzia invece l'impossibilità di performare questo attacco sfruttando la soluzione descritta. L'attaccante, nel secondo caso, può comunque intercettare il pacchetto dal mittente ma non potrà più aggiornare il campo frame counter a proprio piacimento in quanto non sarà in grado di generare un MIC valido per quel nuovo pacchetto (la chiave utilizzata per la generazione del MIC è in possesso solo dei due nodi che effettuano la comunicazione) e quindi il pacchetto malevolo verrà scartato dal nodo ricevente in quanto la convalida del MIC non andrà a buon fine.

Di seguito viene riportata l'implementazione dell'attacco su omnet++:


```
Caught Exception...  
HashVerificationFilter: message hash or MAC not valid
```

```
pacchetto scartato perchè non corretto  
cancellato il pacchetto perchè la decifratura/verifica ha dato errore
```

Come mostrato nell'immagine precedente, l'attaccante anche in questo caso intercetta il pacchetto dal nodo_1 e genera un pacchetto malevolo alterando il frame counter e riutilizzando il MAC del mittente, ma in questo caso l'attacco non può andare a buon fine in quanto il destinatario del pacchetto rileva un'incongruenza grazie al meccanismo di autenticazione che è supportato in tutti i restanti livelli di sicurezza. Come si può vedere nella seconda immagine, l'attaccante non è in grado di generare un MIC valido per il pacchetto malevolo in quanto l'algoritmo di verifica solleverà un errore e il pacchetto verrà scartato prima di aggiornare i valori del frame counter.

Come possiamo vedere, entrambe le soluzioni sono quindi strettamente correlate alla **segretezza delle chiavi crittografiche**. Nel documento, come in tutto il progetto, la gestione delle chiavi non è stata presa in considerazione (come viene specificato anche all'interno dello standard IEEE 802.15.4) in quanto la realizzazione di un meccanismo sicuro di generazione e scambio di chiavi è responsabilità dei livelli superiori. Legare la robustezza di tali meccanismi alla segretezza delle chiavi è un modo per eliminare complessità e responsabilità dal protocollo ed è adottato spesso all'interno dei contesti di sicurezza, quindi si è ritenuto corretto demandare questa criticità a protocolli appositi e meccanismi di più alto livello.

6 PERFORMANCE

Nei capitoli precedenti sono state trattate le soluzioni solo da un punto di vista prettamente legato alla sicurezza, ma per tale tipologia di protocollo è stato molto importante anche tener conto del costo, a livello energetico, delle soluzioni adottate. Nei contesti in cui viene utilizzato l'802.15.4, il **consumo energetico** risulta essere un fattore chiave che deve essere tenuto in considerazione quando vengono proposte delle migliorie come in questo elaborato. Verranno quindi trattati in maniera approfondita i vari aspetti che caratterizzano il calcolo delle performance di un protocollo, ovvero l'802.15.4, che fa del risparmio energetico uno dei suoi punti di forza, sfruttando i dati raccolti per valutare l'appesantimento del protocollo nella sua versione sicura.

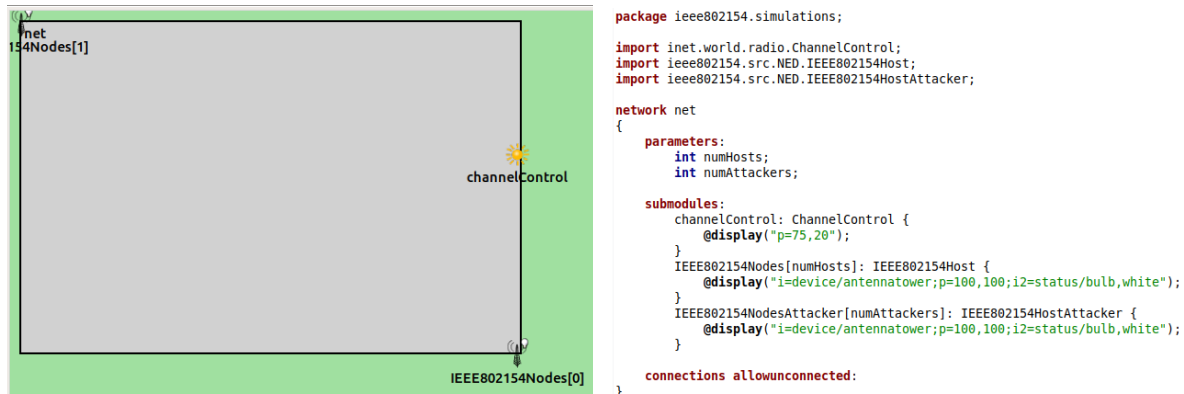
Per una corretta analisi, essendo il progetto realizzato tramite il simulatore **Omnet++**, si è deciso di ipotizzare che la rete sia formata da diversi moduli **CC1310**, appartenenti alla *Texas Instruments*, che supportano lo standard Zigbee e l'802.15.4.



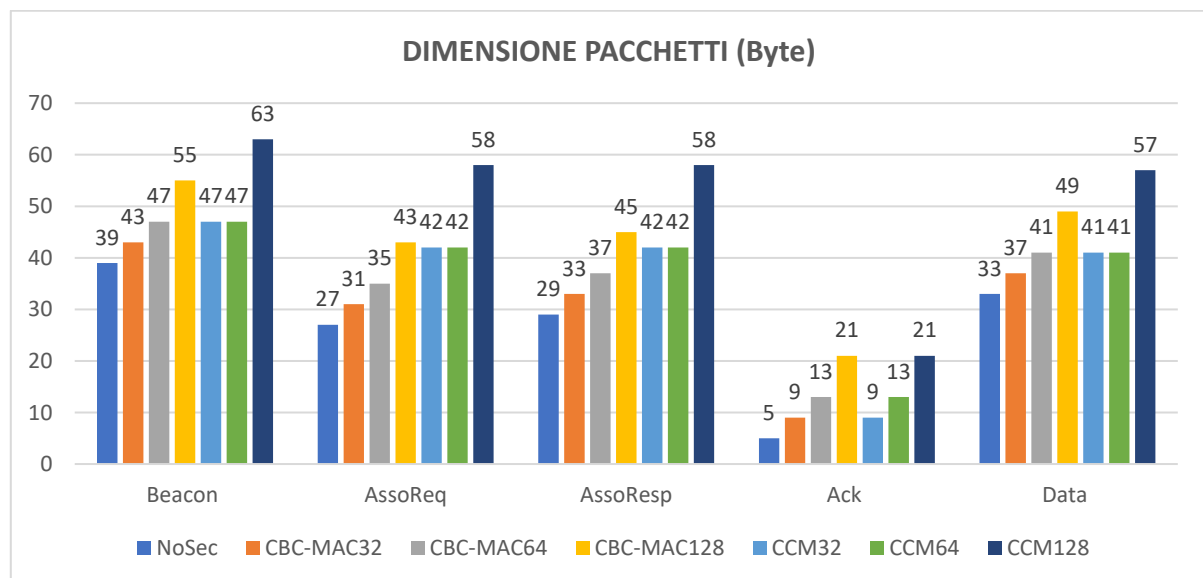
La scelta di tale modulo è legata allo studio effettuato prendendo in considerazione l'articolo [7] che va ad analizzare i tempi ed il consumo energetico del modulo sopracitato in contesti di sicurezza analizzandone le performance con i protocolli CCM, CBC-MAC e CTR, ovvero i protocolli che vengono utilizzati all'interno dello standard. Dal manuale del CC1310 [8] sono stati estrapolati i consumi energetici relativi agli stati di *IDLE*, *SLEEP*, *TX* e *RX*, che sono mostrati nella seguente tabella:

PARAMETER		TEST CONDITIONS	TYP	UNIT
I _{core}	Core current consumption	Reset. RESET_N pin asserted or VDD5 below power-on-reset threshold	100	nA
		Shutdown. No clocks running, no retention	185	
		Standby. With RTC, CPU, RAM, and (partial) register retention. RCOSC_LF	0.7	μA
		Standby. With RTC, CPU, RAM, and (partial) register retention. XOSC_LF	0.8	
		Idle. Supply Systems and RAM powered.	570	mA
		Active. MCU running CoreMark at 48 MHz	1.2 mA + 25.5 μA/MHz	
		Active. MCU running CoreMark at 48 MHz	2.5	mA
		Active. MCU running CoreMark at 24 MHz	1.9	
		Radio RX, 868 MHz	5.5	mA
		Radio TX, 10-dBm output power, (G)FSK, 868 MHz	13.4	
		Radio TX, OOK modulation, 10-dBm output power, AVG	11.2	mA
		Radio TX, boost mode (VDDR = 1.95 V), 14-dBm output power, (G)FSK, 868 MHz	23.5	
		Radio TX, OOK modulation, boost mode (VDDR = 1.95 V), 14-dBm, AVG	14.8	mA
		Radio TX, boost mode (VDDR = 1.95 V), 15-dBm output power, (G)FSK, measured on CC1310EM-7XD-4251, 433.92 MHz	25.1	
		Radio TX, 10-dBm output power, measured on CC1310EM-7XD-4251, 433.92 MHz	13.2	mA

La rete su cui sono state effettuate le analisi è una rete con topologia a stella, composta da 2 nodi, il nodo_0 che simula il **coordinatore** ed il nodo_1 che simula un **sensore**. La scelta dei due nodi è stata fatta in modo tale da ridurre la presenza di collisioni a 0, così da poter analizzare le performance della sicurezza nel modo più preciso possibile, andando a verificare correttamente i tempi di *trasmissione*, di *ricezione*, *cifratura*, *decifratura* e *autenticazione*. Di seguito viene riportato il codice che definisce la rete finora decritta:



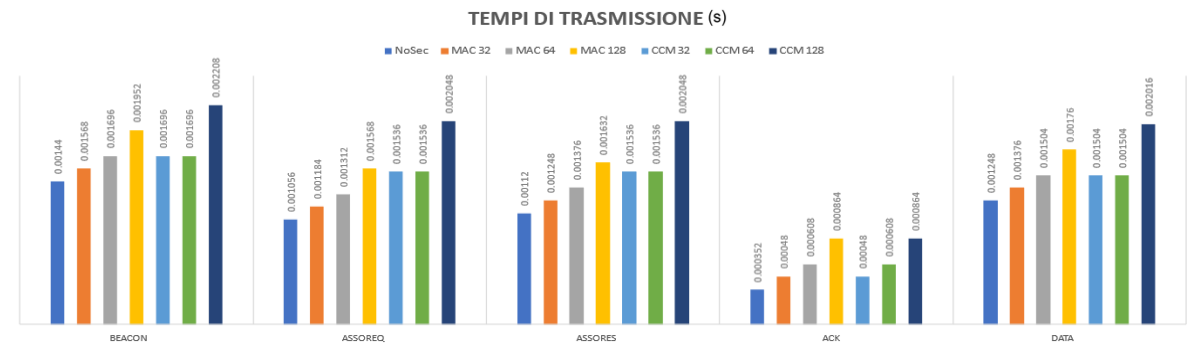
La prima analisi effettuata riguarda l'appesantimento dei pacchetti dovuto all'implementazione della sicurezza nel simulatore. Com'è possibile vedere dal grafico sottostante, i pacchetti relativi al protocollo sono relativamente piccoli e l'aggiunta dei parametri di sicurezza descritti dal protocollo influisce in maniera significativa sulla grandezza dei pacchetti e di conseguenza sui tempi di trasmissione.



Com'è possibile notare, l'overhead dovuto alla sicurezza è estremamente significativo. Mediamente i pacchetti incrementano la propria dimensione (nel livello di sicurezza più pesante, ovvero il CCM128) di circa il 40% rispetto alla versione senza la sicurezza. Possiamo notare come il pacchetto di

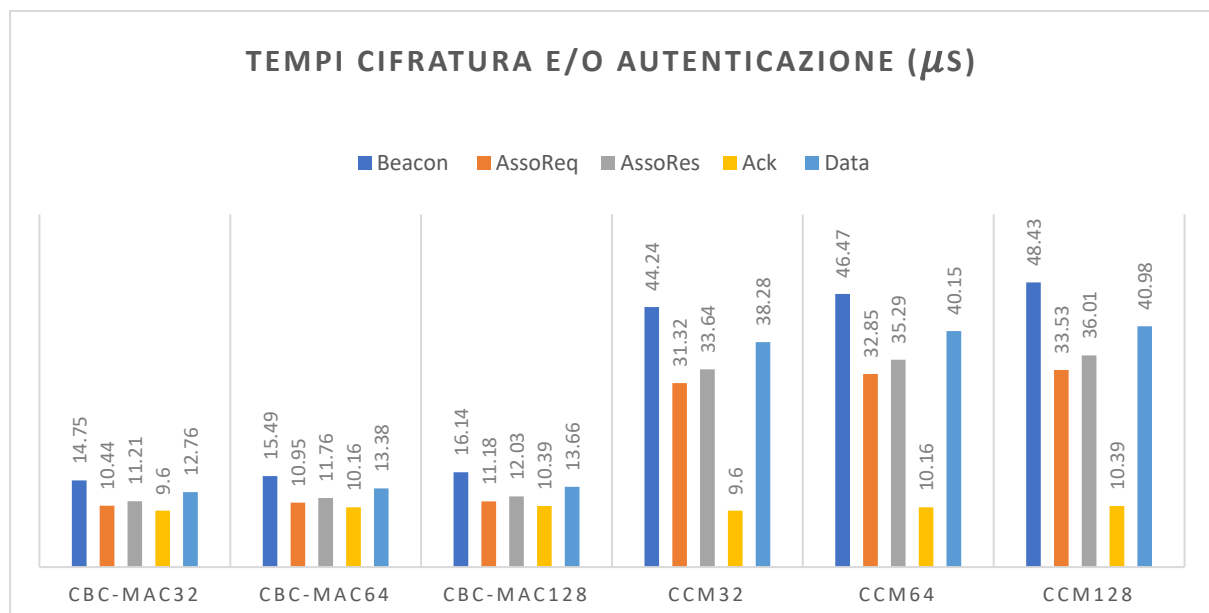
acknowledge risulta invece quasi quadruplicato rispetto alla versione senza sicurezza, e questo probabilmente è il motivo per cui all'interno dello standard non era prevista l'autenticazione dell'Ack.

Una volta calcolato l'appesantimento dei pacchetti, le nuove grandezze sono state inserite all'interno del simulatore in modo tale da poter ottenere le prime analisi relative alle performance dei sensori, che corrispondono ai tempi di trasmissione dei pacchetti e i tempi di autenticazione e cifratura.



Dai dati estrapolati direttamente dal simulatore possiamo notare un netto aumento dei tempi di trasmissione che varia dal 30% all'80%, non tenendo in considerazione l'acknowledge che raddoppia il suo tempo di trasmissione.

Verranno di seguito riportati i tempi di autenticazione e cifratura dei pacchetti, calcolati per ogni livello di sicurezza supportato dallo standard:



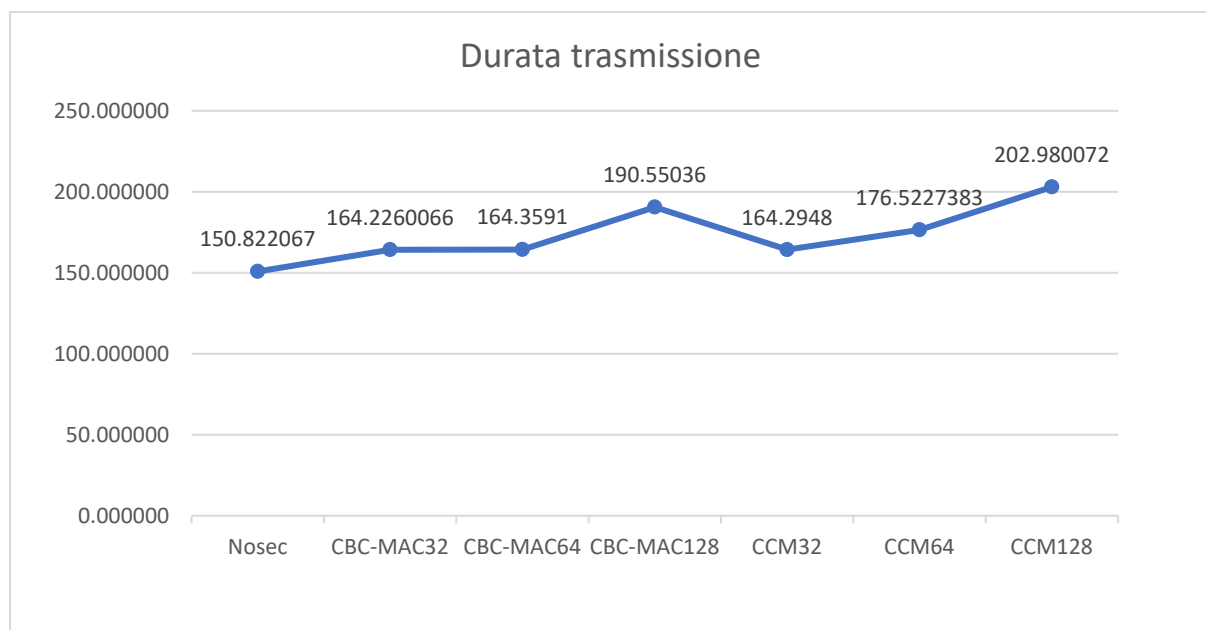
Com'è possibile visualizzare dal grafico precedente, il beacon in tutti i livelli di sicurezza risulta essere il pacchetto che impiega più tempo per essere reso sicuro. Questo fenomeno è dovuto a due fattori:

1. L'autenticazione è applicata all'intero pacchetto ed essendo che il beacon è il pacchetto con dimensione maggiore, allora i tempi risultano essere i più grandi.
2. Il pacchetto beacon, insieme al pacchetto data, è il frame con il quantitativo di byte da cifrare più elevato.

Una volta ottenute le informazioni legate a trasmissione, dimensione e durata di cifratura dei pacchetti, si è passati alla realizzazione di due scenari per analizzare in maniera più approfondita ulteriori informazioni sulle performance.

6.1 PRIMO SCENARIO

Lo scenario in questione è stato realizzato per analizzare l'influenza della sicurezza in termini di durata delle trasmissioni. Proprio per questo, lo scenario risulta essere "anomalo", e prevede l'invio di 20.000 pacchetti con un rate di generazione di 1ms tra un pacchetto e l'altro. Solitamente, nei contesti reali, il rate di generazione risulta essere nettamente inferiore, all'incirca tra il decimo di secondo ed il secondo. Tale scenario è stato avviato inizialmente senza nessuna tipologia di sicurezza, e successivamente le analisi sono state effettuate per ogni livello di sicurezza messo a disposizione dal protocollo, ottenendo diversi tempi di durata della trasmissione. I risultati ottenuti, calcolati in secondi, sono:

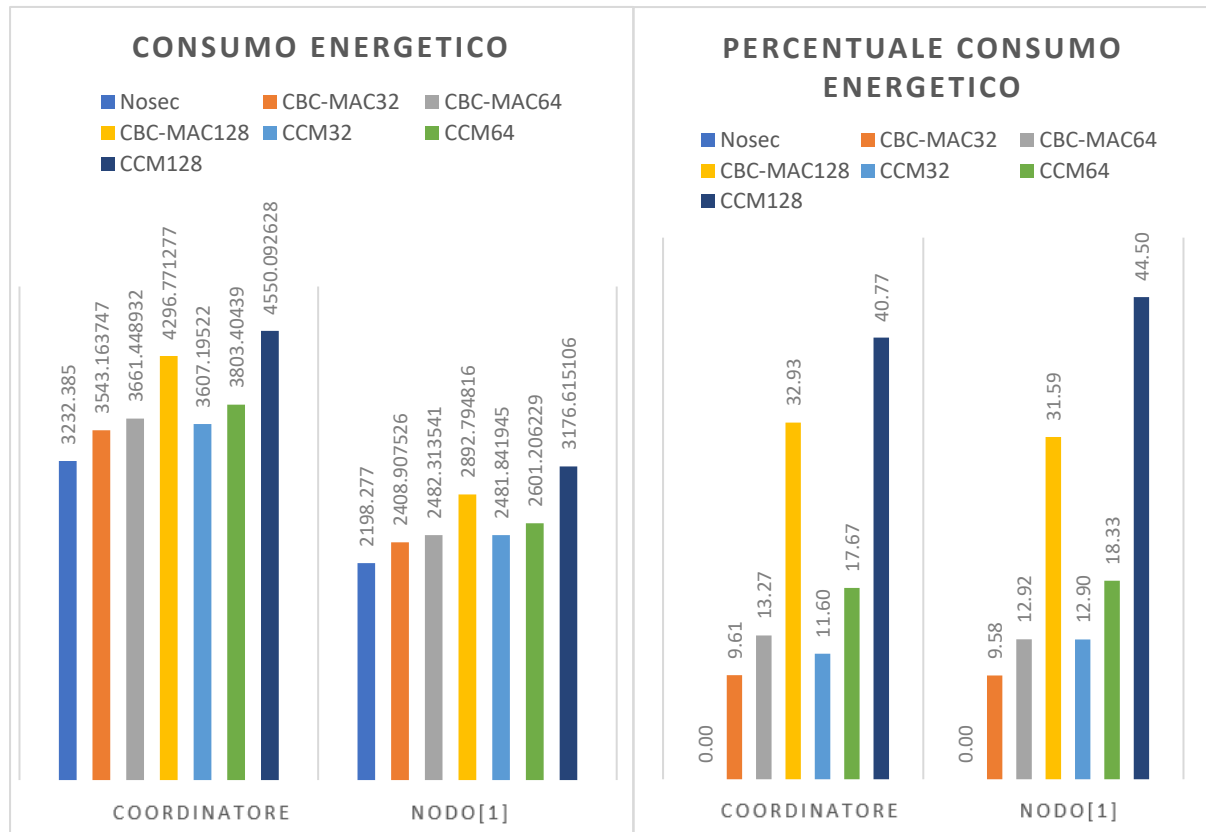


Come illustrato dal grafico, si passa da un tempo di circa 150 secondi per una trasmissione senza sicurezza, ad un tempo di circa 202 secondi per il livello di sicurezza più pesante, ovvero il CCM128. Anche in questo caso, come visto anche nei grafici precedenti, si può notare che i due livelli più dispendiosi sono il livello CBC-MAC128 e il livello CCM128 a causa del maggior peso dei pacchetti e maggiori tempi di cifratura e autenticazione.

Per quanto riguarda invece il consumo energetico, si sono andati a calcolare i valori seguendo la seguente formula:

$$E_{Host} = (T_{idle} * I_{idle} + T_{sleep} * I_{sleep} + T_{recv} * I_{recv} + T_{tx} * I_{tx} + T_{sec} * I_{sec}) * V$$

Dove V corrisponde al voltaggio del modulo utilizzato (valore estrapolato dal manuale e pari a 3,6 V), T_{sec} corrisponde al tempo totale utilizzato per garantire la sicurezza, quindi tempo di autenticazione, cifratura, decifratura e verifica, mentre I_{sec} corrisponde ai valori energetici utilizzati dal modulo per attuare i protocolli di sicurezza. Di seguito è riportata la tabella, in **mJ**, con i risultati ottenuti dall'analisi:



Dai dati ricavati, la prima cosa che si nota è che il nodo coordinatore risulta essere più dispendioso in termini di consumo energetico rispetto al nodo_1. Questo è dovuto alla maggiore mole di dati e azioni che il coordinatore è chiamato ad eseguire rispetto ad un nodo sensore, che risulta essere computazionalmente più semplice. I dati ottenuti dimostrano che, per il nodo coordinatore, nel caso peggiore si ottiene un dispendio energetico che risulta essere il 40% in più rispetto alla versione insicura del protocollo, ed un aumento di circa il 45% per il nodo_1. Nel caso migliore, ovvero in quei livelli di sicurezza in cui viene troncato il MIC, si nota che entrambi i nodi hanno un dispendio energetico del 10% in più rispetto alla versione insicura.

6.2 SECONDO SCENARIO

Per quanto riguarda invece il secondo scenario, l'obiettivo è quello di analizzare il dispendio energetico a lungo termine dei dispositivi, andando così a comprendere l'effettiva durata della batteria. Per tale motivo è stato realizzato uno scenario che rappresenta un contesto più realistico, in cui è presente una trasmissione tra due nodi, la cui generazione dei pacchetti avviene ogni 500ms, avendo pertanto un rate di trasmissione inferiore rispetto al primo scenario. Le considerazioni in questo caso non

verranno effettuate sul numero di pacchetti inviati dal sensore, ma il focus viene spostato sul tempo, prendendo quindi in considerazione un'ora di trasmissione.

Per effettuare tale analisi, si è calcolato il dispendio di corrente da parte di un nodo per un'ora di trasmissione e, fatto ciò, si è ricavato il numero di ore di vita del dispositivo tramite la seguente formula:

$$T = \frac{C}{I_{tot}}$$

Dove C corrisponde alla capacità della batteria, pari a *2000mAh*, mentre I_{tot} corrisponde alla corrente totale utilizzata dal nodo per l'intera trasmissione. Per avere un'indicazione generale sul tempo totale di vita del dispositivo (parametro fondamentale in questo protocollo), sono stati prima calcolati i consumi di corrente per ogni livello di sicurezza presente nello standard:

	CBC-MAC32	CBC-MAC64	CBC-MAC128	CCM32	CCM64	CCM128
Nodo_1	0,1088 mAh	0,1129 mAh	0,1234 mAh	0,1116 mAh	0,1166 mAh	0,1303 mAh

Successivamente, sono stati dilatati questi consumi simulando l'intero ciclo di vita della batteria, ottenendo i seguenti risultati:

	NOSEC	CBC-MAC32	CBC-MAC64	CBC-MAC128	CCM32	CCM64	CCM128
Nodo_1	814 gg	776 gg	738 gg	675 gg	746 gg	715 gg	640 gg

I risultati ottenuti sono coerenti con la media dei dati ottenuti dai vari studi presi in considerazione. Si può notare come questi dispositivi siano progettati per durare negli anni, senza la necessità di intervento da parte dell'uomo. Garantendo il livello di sicurezza più robusto (che corrisponde al più dispendioso), si nota come il costo della sicurezza non sia trascurabile (ovvero 174gg in meno).

Il protocollo stesso garantisce una scelta sul livello di sicurezza adottabile, che in combinazione ai dati che sono stati resi disponibili in questo lavoro, permette ad un progettista di poter prendere una scelta oculata tenendo conto sia del grado di sicurezza necessario per un determinato contesto, e sia il dispendio energetico che ne consegue. Ci saranno contesti in cui sarà più consigliabile dare maggiore importanza alla sicurezza piuttosto che alle performance, e contesti in cui è preferibile il contrario.

7 CONCLUSIONI

Il lavoro realizzato è stato d'aiuto nel comprendere uno dei protocolli su cui si punta molto in ambito **IoT**. Proporre delle soluzioni per questi contesti non è mai semplice, in quanto c'è da trovare il giusto equilibrio tra il livello di sicurezza che si vuole andare a garantire e le esigenze dettate dal mercato in cui questi protocolli vengono utilizzati.

Come ripetuto più volte all'interno dell'elaborato crediamo che in alcuni contesti (come sistemi di rilevazione remota di parametri vitali, o sistemi di sicurezza) tali modifiche debbano essere adottate a discapito delle performance e dei consumi energetici, in quanto le vulnerabilità analizzate possono essere sfruttate dagli attaccanti causando conseguenze catastrofiche. Si spera che questo elaborato possa essere di supporto ai progettisti di sistemi IoT per avere dei parametri con cui scegliere il corretto livello di sicurezza in base a diversi fattori analizzati. Uno sviluppo futuro potrebbe essere quello di analizzare e mettere in pratica i risultati ottenuti da questo elaborato per creare delle **linee guida** e/o standard che, in base alle informazioni sul contesto, suggeriscono il livello di sicurezza più adatto tenendo conto anche del consumo energetico che ne deriva. In conclusione, pensiamo che il lavoro fatto non solo sia stato molto utile per apprendere e mettere in pratica le nozioni di network security apprese durante il corso, ma anche per fornire una base di studio su cui altri colleghi possano partire e sfruttare per creare una guida utile alla comunità.

8 RIFERIMENTI

- [1] «Low-Rate Wireless Personal Area Networks (LR-WPANs),» *IEEE Computer Society*.
- [2] A. M. Ghada Glissa, «IEEE 802.15.4 Security Sublayer for OMNET++».
- [3] G. D. G. A. ROBERTA DAIDONE, «On Evaluating the Performance Impact of the IEEE 802.15.4 Security Sub-layer».
- [4] S. M. Bellovin, «Problem areas for the IP security protocols,» *In Proceedings of the Sixth Usenix UNIX Security Symposium*.
- [5] I. G. a. D. W. Nikita Borisov, «Intercepting mobile communications: The insecurity of 802.11».
- [6] «Weak crc allows packet injection into ssh sessions encrypted with block ciphers».
- [7] W.-T. H. Chung-Wen Hung, «Power Consumption and Calculation Requirement Analysis of AES for WSN IoT».
- [8] t. instruments, «CC1310 SimpleLink™ Ultra-Low-Power Sub-1 GHz Wireless MCU».