



IoT Security

Appunti

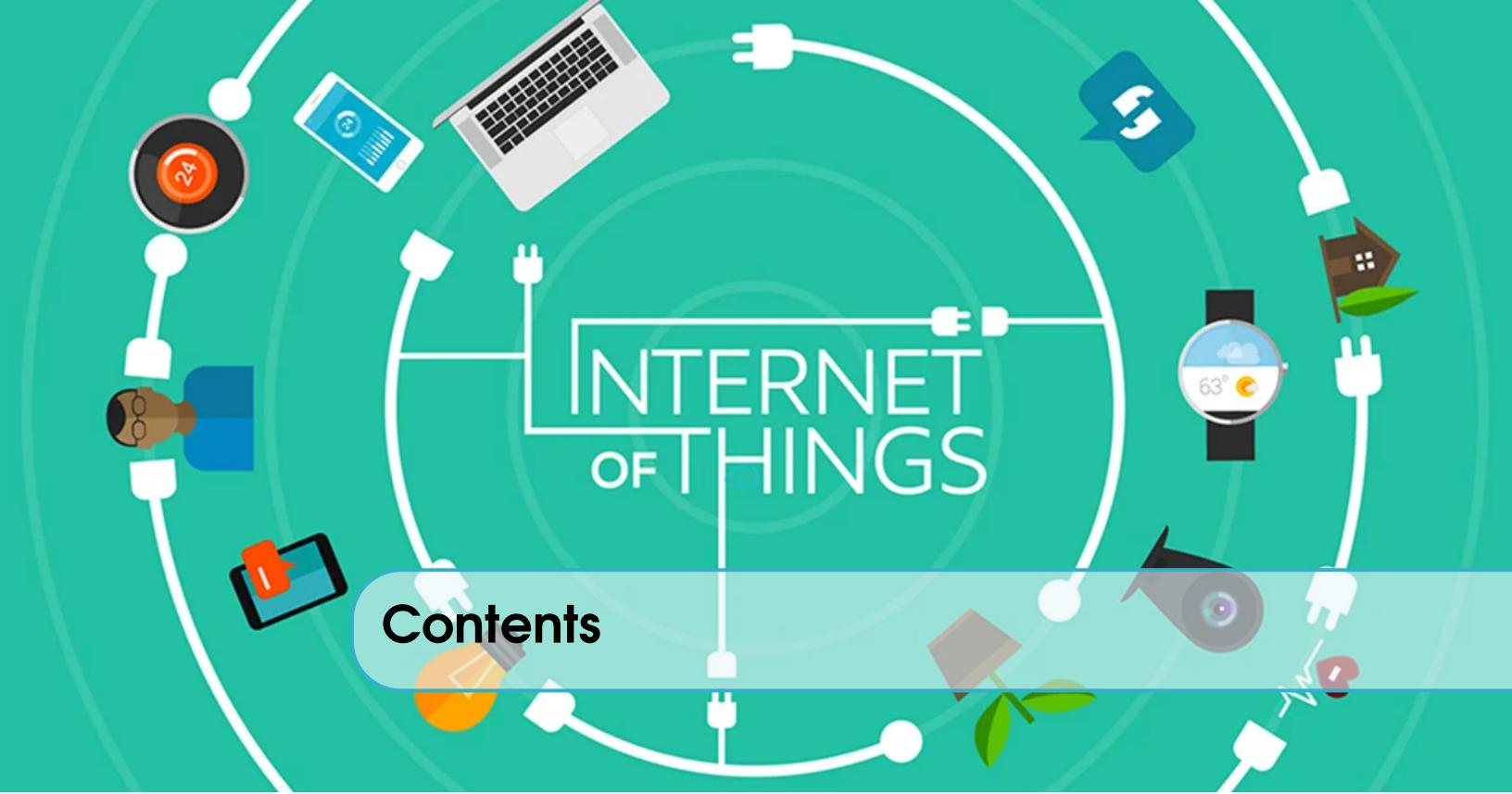
Luca Boffa, Vincenzo Di Leo

APPUNTI DI LUCA BOFFA, UNIVERSITY OF SALERNO

[HTTPS://GITHUB.COM/LUKE31999](https://github.com/Luke31999)

Questi appunti sono stati scritti da Luca Boffa usando le slides del Prof. Christiancarmine Esposito

Scritti durante il primo semestre dell' anno accademico 2022/2023



1	Introduzione al mondo IoT	7
1.1	Sensori: anatomia e tassonomia	8
1.2	Sensor hardware	9
2	IoT Software and Communication	11
2.1	Sensor Software Abstraction	11
2.1.1	Allocazione Memoria	12
2.1.2	Consumo energetico	12
2.2	Tiny OS	13
2.2.1	nesc	13
2.3	Middleware	13
2.4	Communication Platform	14
2.4.1	Tecnologie Wireless	14
2.4.2	RFID	15
2.4.3	5G	15
2.4.4	Low-PowerWide-Area Network	16
2.4.5	IPV6	16
2.4.6	Communication protocols	16
2.5	Protocolli	17
2.5.1	HTTP	17
2.5.2	COAP	18
2.5.3	MQTT	18

3	Security and Privacy for IoT	19
3.1	Security and Privacy	19
3.2	Privacy type	20
3.3	IoT Vulnerabilities and Attacks	21
3.3.1	Attacks against IoT	22
3.4	Privacy Issues	23
3.5	IoT Forensics	24
3.5.1	Introduzione	24
3.5.2	Sfide	24
3.5.3	IoT Forensics Approaches	26
4	Secure IoT Product Design & Coding	27
4.0.1	Introduzione	27
4.0.2	Criticality Assessment (Valutazione della criticità)	28
4.1	IoT Product Development Process	28
4.1.1	Security Requirements	28
4.1.2	Analisi delle minacce alla sicurezza	29
4.1.3	Strategia di sviluppo	29
4.2	Secure Coding	30
5	Exploiting Low Level Communications	31
5.1	Manomissione di dispositivi IoT	31
5.2	UART, I²C, SPI	32
5.2.1	Low-Level Communication	32
5.2.2	UART	33
5.2.3	I ² C and SPI	33
5.2.4	Low Level Communication Exploitation	34
5.3	JTAG and its Exploitation	34
5.3.1	JTAG Exploitation	35
5.3.2	JTAG Protection	36
5.4	Protecting Low Level Communications	36
6	Firmware Hacking and Emulation	37
6.1	Firmware Hacking and Emulation	37
6.1.1	Firmware Hacking	37
6.1.2	Firmware Emulation	38
6.2	Backdooring Firmware	38

7	Physical Attacks and Protections	39
7.1	Physical Attacks	39
7.1.1	Attacchi non invasivi	40
7.1.2	Attacchi invasivi	41
7.1.3	Attacchi semi-invasivi	41
7.2	Protecting IoT Devices from Physical Attacks	41
7.2.1	Tamper Resistance (Resistenza alla manomissione)	41
7.2.2	Tamper Protection (Protezione antimanomissione)	42
8	Physically Unclonable Functions	43
8.1	Introduction to PUFs	43
8.2	Intrinsic PUF	44
8.2.1	Arbiter PUF	44
8.2.2	Ring Oscillator PUF	45
8.2.3	Glitch PUF	45
8.2.4	SRAM PUF	46
8.2.5	POK	46
8.2.6	Controlled PUFs	47
8.2.7	Reconfigurable PUFs	47
8.3	PUFs based identification	47
8.4	PUF-Based Key Gen	47
9	Lightweight Cryptography	49
9.1	Introduction to Cryptography	49
9.2	Introduction to Lightweight Cryptography	50
9.3	Implementazione hardware	50
9.3.1	Memory limit	51
9.4	Implementazione software	51
9.4.1	Robustezza SCA	52
9.4.2	Trends in LC	52
9.4.3	Trade-Offs in LC	53
9.4.4	Hashing	53
9.4.5	Streaming Ciphers	54
9.5	Block Ciphers	54
10	TEE	55
10.1	Trusted Execution Environments	55
10.1.1	Soluzioni Hardware	56
10.1.2	TEE	57

11	Communication Security	59
11.1	Secure Communication Means	59
11.1.1	Secure Communication	59
11.1.2	End-point Authentication	60
11.1.3	Jamming Attacks	61
11.2	Data Link-Level Security	61
11.2.1	WEP	61
11.2.2	WPA	61
11.2.3	IEEE 802.11i o WPA2	62
11.2.4	Bluetooth Security	62
11.2.5	ZigBee Security	63
11.3	Network-Level Security	63
11.3.1	IPSec	63
11.4	Transport-Level Security	64
11.4.1	SSL	64
11.4.2	TLS	64
11.4.3	DTLS	65
11.4.4	HTTPS	65



1. Introduzione al mondo IoT

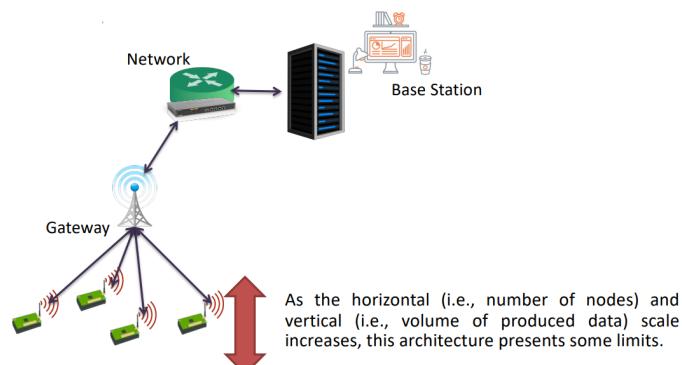
Una rete distribuita di sensori (Wireless Sensor Network - **WSN**) è una nuova classe di rete wireless ed è divenuta popolare grazie ad usi militari volti soprattutto al monitoraggio di parametri fisici.

Una rete di sensori distribuiti contiene una serie di piccoli dispositivi di rilevamento, dei nodi, intercomunicanti e indipendenti che sono utilizzati per il monitoraggio delle condizioni ambientali e fisiche, questi possono comunicare tra loro grazie ad una rete adeguata. In questo modo otteniamo vari tipi di informazioni sull'ambiente, quest'ultime vengono inviate ai nodi gateway, passandoli a una stazione base, con l'obiettivo di attivare un allarme o per interfacciarsi con altri sistemi, quindi prendono decisioni.

In poche parole con il termine IoT indichiamo una rete di oggetti fisici che raccolgono e scambiano dati grazie a Internet.

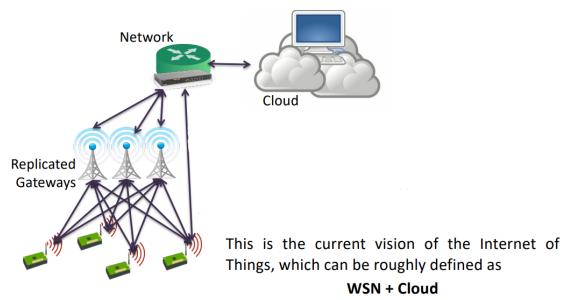
Evoluzione delle architetture

All'inizio le reti di tipo WSN erano progettate su misura ed erano di piccola o media dimensione. In genere ogni le reti possono scalare orizzontalmente (aumentando il numero di nodi) e verticalmente (andando a produrre sempre più dati), e questo tipo di architettura presenta alcuni limiti.



Il singolo gateway in figura rappresenta un "single-point-of-failure" che può essere risolto andando a effettuare una replicazione attiva o passiva. La base station invece può essere sovraccaricata dalla mole di lavoro e per risolvere ciò introduciamo il cloud.

All'aumentare dei nodi delle reti si ha la necessità che alcuni di loro abbiano un accesso diretto al network, bypassando i gateways. Per cui un primo livello di dati viene effettuato "at the edge", al bordo dell'infrastruttura, un livello intermedio di elaborazione/memorizzazione avviene presso un nodo più potente. Infine, un flusso di dati limitato va verso il cloud.



1.1 Sensori: anatomia e tassonomia

Il componente base di una rete di sensori distribuiti è il nodo sensore, un piccolo dispositivo, con una batteria vincolata, in grado di monitorare alcune caratteristiche dell'ambiente in cui si trova, e di elaborare, inviare o conservare le informazioni di monitoraggio. Dati i progressi della tecnologia nei semiconduttori, i costi di questi sensori stanno decrescendo gradualmente nel tempo. Possiamo trovare diverse caratteristiche e definizioni di cosa è il nodo sensore, ma probabilmente il più esaustivo è quello proposto dallo standard IEEE 1451.2. Il sensore deve disporre di una memoria locale in cui i programmi e i dati di funzionamento e monitoraggio siano permanentemente collocati. Questa memoria ha una capacità limitata e il suo utilizzo deve essere ottimizzato. Le memorie flash possono essere aggiunte a quelle esistenti per aumentare la capacità di archiviazione. Un microcontrollore eseguirà algoritmi specifici per l'esecuzione di applicazioni, ma anche routine per attività di autodiagnosi atte a determinare lo stato di funzionamento del dispositivo, effettuare misurazioni nel tempo e nello spazio.

I sensori devono poter comunicare facilmente tra loro e con le stazioni base tramite una modalità plug and play. Per questo scopo, una o più tecnologie di comunicazione wireless possono essere adottate e supportate dal dispositivo. Un sensore viene utilizzato per misurare una o più grandezze fisiche in l'ambiente in cui è collocato, quindi deve essere dotato di uno speciale hardware di misurazione. I dati prelevati devono essere elaborati in modo appropriato attraverso operazioni (es. amplificazione, filtraggio, adattamento di livello). Lo scopo principale è ottimizzarli in modo da essere compatibili con la successiva conversione da analogico a digitale che li rende leggibili tramite strumenti software. Il sensore ha un'alimentazione interna, essendo ricaricabile o non, per mezzo di una fonte rinnovabile, oppure si può collegare il sensore all'alimentazione. Il sensore potrebbe essere dotato di un apposito attuatore che può svolgere azioni ed elaborare strategie di controllo appropriate basate sul monitoraggio dei dati, come ad esempio attivare un allarme ad una determinata condizione.

1.2 Sensor hardware

Attualmente sono disponibili diverse piattaforme hardware per la realizzazione di nodi di sensori da integrare per realizzare reti di sensori distribuite.

Abbiamo assistito alla progressiva miniaturizzazione di questo hardware, soprattutto con l'avvento della tecnologia dei Micro Electronic Mechanical Systems (MEMS), definibili come elementi meccanici ed elettromeccanici miniaturizzati, realizzati tramite microfabbricazione.

Possiamo classificare le piattaforme hardware utilizzabili nelle reti di sensori distribuite in due classi:

- Hardware per uso **generico**: include piattaforme programmabili e generiche (Arduino).
- Hardware **ad hoc** per attività di monitoraggio (sensori).

Per l'hardware ad hoc troviamo 4 componenti fondamentali come:

- **Unità centrale di elaborazione (CPU)**: l'elemento principale con il compito di effettuare operazioni(es. esecuzione del codice). Il MicroController Unit (MCU) fornisce funzionalità aggiuntive come temporizzatori, oscillatori, memorie (di programma come ROM, EPROM, FLASH o dati come RAM ed EEPROM), porte I/O.
- **Apparecchiature per il monitoraggio**: possono essere modificate secondo l'output, analogico significa che abbiamo un segnale da processare attraverso circuiti analogici, e digitale, il cui segnale va processato attraverso circuiti digitali.
- **Interfacce di comunicazione**: comunicazione con dispositivi esterni al nodo sensore, può essere di due tipi:
 - Comunicazione **via cavo**: il nodo sensore è interconnesso con gli altri componenti tramite cavi, ad esempio Ethernet.
 - Comunicazioni **wireless**: il nodo del sensore è interconnesso senza l'uso di cavi, ma per mezzo di tecnologie a infrarossi o a radiofrequenza. Gli esempi sono ZigBee, Bluetooth o 5G. Quale tecnologia usare dipende dallo scopo della comunicazione e dal consumo di energia. Riduce i costi di installazione della rete, ma lo rende più vulnerabile alle interferenze e all'intercettazione di dati. Questa è preferibile poichè l'installazione è più semplice.

- **Batterie**.

Per i sistemi di monitoraggio le interconnessioni digitali con le apparecchiature sono realizzate con protocolli standard:

- **U(S)ART**: Universal (Syncronous) Asynchronous Receive Transmit: modulo per comunicare in modo asincrono e in modo sincrono con un altro modulo sia per trasmettere che per ricevere dati secondo un protocollo. Per la comunicazione sincrona, il processore deve essere sempre attivo per controllare il ritorno del segnale, quindi rimane bloccato nella comunicazione. Per l'asincrona, il processore nota che c'è una risposta grazie ad una interruzione. E' un protocollo di comunicazione uno ad uno ed è a bassa velocità.
- **I²C**: Inter Integrated Circuit: protocollo che prevede la "scambio" di dati tra uno o più Master ed uno o più Slave slave, tramite un bus dati e uno di clock. Sono possibili tre velocità (100 KHz, 400 KHz, 1 Mhz).
- **SPI**: Serial Peripheral Interface: protocollo Master-Slave, con ciascuno dotato di un registro a scorrimento contenente i dati e collegato con un minimo di quattro bus. Il trasferimento comporta lo scambio dei contenuti del registro.

Ev~~e~~ryware IoT



2. IoT Software and Communication

2.1 Sensor Software Abstraction

Per evitare che il progettista debba occuparsi di aspetti di basso livello quando programma i nodi, sono stati proposti una serie di sistemi operativi per le reti di sensori.

Il sistema operativo di solito supporta uno specifico linguaggio di programmazione, che di solito è una derivazione del C o uno specifico ideato per l'IoT. Inoltre, viene fornito un insieme di primitive di comunicazione di basso livello.

I sistemi operativi tradizionali sono sistemi software che gestiscono risorse informatiche, dispositivi di controllo periferici e forniscono astrazioni software per le applicazioni. Sono utilizzati principalmente per gestire processi, memoria, tempo di CPU, file system e dispositivi. Di solito sono realizzati in modo modulare e a livelli, livello basso chiamato kernel e un livello alto di librerie di sistema. I sistemi operativi tradizionali non sono ottimali per le reti di sensori perché queste reti hanno risorse limitate e diverse applicazioni incentrate sui dati, oltre a una topologia variabile. Ci sono vari problemi da affrontare per un sistema operativo riguardo i nodi IoT in una rete:

- **Gestione e schedulazione dei processi:** i sistemi operativi tradizionali allocano i processi in uno spazio di memoria separato, causando tempi di gestione elevati per il passaggio tra i processi e un consumo elevato. Arduino esegue un processo singolo, questo significa che non c'è context switch, in modo da ridurre la memoria utilizzata e il consumo di energia, che avremmo in caso di context switch.
- **Gestione della memoria:** nei sistemi tradizionali la memoria viene allocata esclusivamente a un processo o task, ma questo non è possibile nei nodi sensori che hanno una scarsa capacità di memoria.
- **Kernel:** tradizionalmente, un kernel è composto da vari moduli interdipendenti e da chiamate di sistema per servizi che vengono eseguiti in modalità supervisore, con un modello multi-thread. Questo non è possibile per un Embedded System, perché ciò consuma energia. I sistemi operativi per i sensor node hanno un modello a macchina di stato o a eventi.
- **Interfaccia del programma applicativo (API):** i nodi IoT richiedono API modulari e generali per le loro applicazioni, ma ciò consente anche un facile accesso all'hardware sottostante.

- **Aggiornamento e riprogrammazione del codice:** dato che il comportamento dei nodi sensore e i loro algoritmi necessitano di vari aggiustamenti alla loro funzionalità o una ottimizzazione dei consumi, il sistema operativo deve essere aggiornato e riprogrammato a sua volta.

Un sistema operativo per i nodi IoT deve offrire le seguenti caratteristiche e funzionalità:

- **Deve essere compatto e di dimensioni ridotte**, dato che i nodi sensore hanno poche decine o centinaia di kilobyte di memoria.
- **Deve fornire un supporto in tempo reale**, per questo non è presente il modulo del multi-thread, poiché le applicazioni di monitoraggio hanno vincoli in tempo reale (non si tratta di velocità, ma il vincolo è che sia "predictable"), soprattutto quando sono coinvolti gli attuatori. Lo scheduling è contro il tempo reale perché ho starvation, deadlock.
- **Deve offrire una gestione efficiente delle risorse**, attraverso un algoritmo, al fine di gestire il tempo del microprocessore e la memoria limitata allocandole con attenzione per garantire l'equità. Uno dei servizi che va spento e acceso in modo da risparmiare energia è la comunicazione poiché richiede molte energie.
- **Deve supportare la distribuzione del codice in modo affidabile e efficiente**, in modo da consentire aggiornamenti dopo l'"hot deployment", ovvero in esecuzione.
- **Deve consentire la gestione del consumo elettrico**, in modo da prolungare il ciclo di vita del nodo sensore e migliorarne le prestazioni. Una cosa da fare è stoppare i task in esecuzione che consumano troppa energia.

2.1.1 Allocazione Memoria

Sui dispositivi IoT, la dimensione della memoria è vincolata sia dal punto di vista fisico (dati dal numero di transistor) e pratico (per mantenere bassi i consumi e i costi di produzione, quindi non posso aumentare la memoria quanto voglio). Tradizionalmente la memoria è divisa in due parti, una parte statica che contiene il programma in esecuzione e una parte dinamica che contiene le variabili runtime, i buffer, i dati di ingresso e di uscita e lo stack. La parte statica è solitamente conservata in una memoria di sola lettura (ROM), mentre la parte dinamica si trova in una memoria ad accesso casuale (RAM). Date le caratteristiche fisiche delle architetture esistenti per microcontrollori, la RAM ha un consumo energetico maggiore della ROM e richiede un'area maggiore del chip. Per questi fattori, la RAM è tipicamente più piccola della ROM.

2.1.2 Consumo energetico

I dispositivi IoT hanno una batteria, con una quantità fissa di energia. Il consumo di energia determina il ciclo di vita, rendendo l'energia un fattore critico in una rete di sensori. La gestione dell'energia è essenziale per un sistema operativo di nodi sensore. Per ridurre il consumo, possiamo spegnere i componenti non utilizzati. In Internet Of Things è impossibile avere una sicurezza piena, full, poiché la sicurezza prevede numerose operazioni, come la **criptazione**, le quali sprecano molta energia. Anche il sistema operativo tiene traccia dei consumi, adottando a questo scopo un approccio sia hardware che software. Un approccio consiste nel monitorare il consumo di energia in toto. Un'altra soluzione consiste nel monitorare i componenti del sistema operativo e attribuire ad essi il consumo, questo include anche l'importo speso per l'hardware utilizzato.

2.2 Tiny OS

TinyOS è un sistema operativo open-source, basato su componenti e applicazioni specifiche sviluppato appositamente per le reti di sensori. Può supportare programmi concorrenti con requisiti di basso consumo energetico e ha una bassa occupazione di memoria (circa 400 byte). TinyOS dispone di una libreria di componenti che includono protocolli di rete, servizi distribuiti, dispositivi di monitoraggio e strumenti di acquisizione dati.

TinyOS rappresenta un esempio di architettura non monolitica, utilizzando un modello di componenti che vengono scelti e assemblati in base alle esigenze delle applicazioni. Un componente è un'entità di calcolo indipendente che dispone di una o più interfacce. I componenti hanno tre astrazioni computazionali: comandi, eventi e compiti.

2.2.1 nesc

NesC è un linguaggio di programmazione a eventi, basato su componenti, per sistemi dedicati, e nasce come "dialetto C". NesC è stato progettato per lo sviluppo di TinyOS, quindi viene utilizzato anche per scrivere programmi applicativi per dispositivi IoT che utilizzano TinyOS. nesC ha la caratteristica di essere modulare, permettendo all'utente di tra loro pezzi di codice. Un'applicazione completa può essere realizzata assemblando diversi componenti software (moduli). Separa l'implementazione dalla composizione con le interfacce. I componenti principali di un'applicazione sono i componenti, che sono collegati tra loro attraverso le loro interfacce. Un componente può fornire l'implementazione di una interfaccia o richiederla.

2.3 Middleware

Esiste un divario tra i protocolli di rete e le applicazioni che deve essere colmato offrendo adeguate funzioni di astrazione, al fine di fornire la qualità del servizio alle applicazioni utilizzando le risorse limitate dei nodi ed estendendo il loro ciclo di vita. La soluzione a questa esigenza è la progettazione di un livello **middleware**, che si trova al di sotto delle applicazioni e al di sopra del sistema operativo e dei protocolli di rete.

Con Middleware si intende un software che fornisce alle applicazioni servizi aggiuntivi a quelli offerti dal sistema operativo. Se io devo comunicare con un server avrei bisogno dell'ip address, con il Middleware non ho bisogno di sapere per forza l'ip address poiché quest'ultimo scopre automaticamente nodi e stabilisce connessioni. Qui abbiamo l'IP Multicast, un metodo per inviare datagrammi IP a un gruppo di riceventi interessati, in una singola trasmissione. La comunicazione è un aspetto molto pesante poiché impiega molte risorse. I middleware per le reti di dispositivi, sono soggetti a vincoli diversi rispetto a quelli degli ambienti IT tradizionali e presentano differenze significative.

Le sfide da affrontare quando si progetta un middleware per le reti di sensori sono le seguenti:

- **Controllo della topologia**, per organizzare i nodi dei sensori in una rete interconnessa. Nativamente i nodi sensore possono essere affetti da failure. Nei protocolli di rete standard ho una probabilità bassa di avere una failure.
- Elaborazione basata sui dati con un **consumo energetico ottimizzato**;
- Uso efficiente delle risorse di calcolo e di comunicazione;

Le funzioni principali di un middleware per reti di dispositivi sono:

- Servizi di sistema standardizzati per varie applicazioni. Il Middleware difficilmente si riferisce ad una applicazione precisa.
- Un ambiente che coordina più applicazioni.
- Meccanismi per ottenere l'uso adattativo ed efficiente delle risorse del sistema, cioè implementazioni di algoritmi che gestiscono dinamicamente le limitate e variabili risorse di rete.
- Bilanciamento efficiente tra molteplici parametri di qualità per ottimizzare le risorse di rete necessarie a fornirli. Ad esempio, io non posso avere contemporaneamente massima sicurezza e massime performance, devo per forza effettuare un bilanciamento.

Inizialmente, la comunità non aveva mostrato alcun interesse per il livello middleware, perché la semplicità delle prime applicazioni non richiedeva un'astrazione di questo tipo. Con la rapida evoluzione e il successo di queste reti, la complessità delle applicazioni è aumentata, così come il divario con i livelli sottostanti. Ci sono altre differenze tra il nuovo middleware e quello tradizionale.

- Spesso i middleware nel contesto mobile si preoccupano di soddisfare gli interessi di un determinato nodo mobile, mentre nel contesto delle reti di sensori si deve pensare all'interesse dell'intera rete.
- I middleware tradizionale ha un meccanismo di indirizzamento unico, mentre nelle reti di sensori l'indirizzamento è spesso incentrato sui dati;
- I middleware per le reti di sensori deve essere leggero, mentre quelli tradizionali hanno una sofisticata stratificazione di software.

Per questo motivo, nel corso degli anni sono state proposte una serie di soluzioni ad hoc, incentrate su diversi aspetti e/o esigenze diverse., e non si usa un Middleware tradizionale. Quindi il Middleware si trova al di sotto delle applicazioni e al di sopra del sistema operativo. Serve come estensione del sistema operativo per supportare le astrazioni sottostanti.

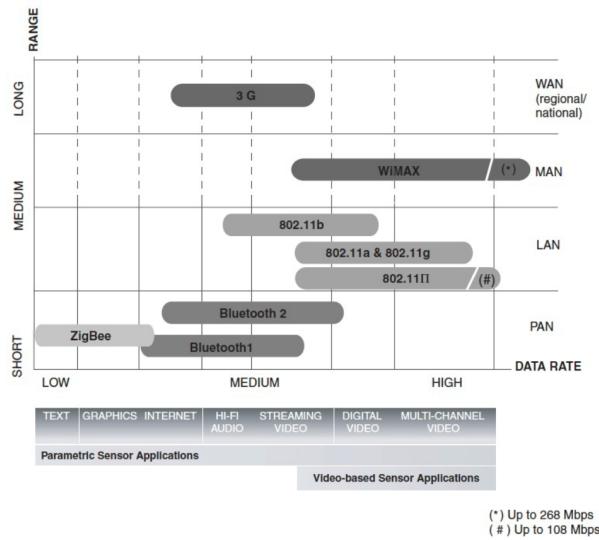
2.4 Comunication Platform

2.4.1 Tecnologie Wireless

Per massimizzare l'opportunità di un maggiore utilizzo e l'efficacia dei costi, è necessario utilizzare le comunicazioni e le infrastrutture wireless commerciali esistenti e/o emergenti, piuttosto che sviluppare un apparato completamente nuovo progettato appositamente per le reti di sensori. Un dispositivo può utilizzare una o più tecnologie wireless.

Il bluetooth è la tecnologia più economica. Arduino, per risparmiare energia, utilizza una sola tecnologia. Il progettista non deve avere una forte conoscenza dei fondamenti radio di queste tecnologie, ma limitarsi a conoscere alcune caratteristiche qualitative come consumo, raggio di copertura dei segnali, larghezza di banda, prestazioni, sicurezza e pochi altri fattori. Le reti di sensori utilizzano tipicamente due bande di frequenza: le bande ISM (Wifi, Bluetooth) e UNII (Unlicensed National. Informazione nazionale non licenziata). Più è alta la frequenza più la rete è vulnerabile agli ostacoli,

alle interferenze. Bisogna effettuare un trade-off tra velocità e robustezza. Le interferenze sono uno dei problemi per quanto riguarda la comunicazione wireless, e le performance e la qualità delle reti dipendono fortemente dalle interferenze. Le interferenze in ambienti interni ed esterni possono essere dovute sia a sorgenti e/o fenomeni naturali (come perdita o attenuazione, assorbimento e fading), ma anche per altri utenti nelle vicinanze che utilizzano queste bande non protette. Tutto ciò comporta perdita di pacchetti. Le più utilizzate sono le seguenti: IEEE 802.15.1 (noto anche come Bluetooth), gli standard LAN wireless IEEE 802.11a/b/g/n, IEEE 802.15.4 (noto come ZigBee), lo standard MAN noto come IEEE 802.16, conosciuto anche come WiMax e l'identificazione a radiofrequenza dei tag (RFID).



Ogni standard ha i suoi vantaggi e i suoi limiti. Ogni standard ha una gamma di copertura e una velocità di trasmissione dei dati. Maggiore è la frequenza maggiore saranno la velocità e i costi per quella tecnologia. Il range è nominale, successivamente ci sono le interferenze, le attenuazioni che diminuiscono il segnale, per cui il range reale non è quello nominale.

2.4.2 RFID

La RFID è una tecnologia per l'identificazione e/o la memorizzazione automatica di dati relativi a oggetti, animali o persone, basata sulla capacità di memorizzazione dei dati da parte di particolari etichette elettroniche, chiamate tag. Le etichette possono rispondere all'interrogazione a distanza da parte di speciali dispositivi speciali chiamati lettori. Negli ultimi anni, lo standard NFC (Near Field Communication, 13,56 MHz e fino a 10 cm, ma con velocità di trasmissione dei dati fino a 424 kBit /...) sta emergendo, estendendo gli standard per permettere lo scambio di informazioni anche tra lettori (comunicazione). Abbiamo gli active tag che possiedono una batteria, i passive senza batteria.

2.4.3 5G

Con il 5G ancora nelle prime fasi di implementazione e non ancora disponibile in tutti i Paesi, si sente parlare di larghezza di banda 5G, di aste per lo spettro 5G, mmW 5G, ecc. Le onde millimetriche, che si trovano nello spettro delle bande alte, hanno il vantaggio di poter trasportare molti dati. Tuttavia, sono anche assorbite più facilmente dai gas presenti nell'aria, dagli alberi e dagli edifici vicini. Le onde mm sono quindi utili in reti densamente popolate, ma non per il trasporto di

dati a lunga distanza (a causa dell'attenuazione). Maggiore è la frequenza maggiore sarà la velocità e minore sarà il range perché è vulnerabile agli ostacoli. Nello spettro 5G, diverse parti dello spettro possono essere utilizzate per massimizzare la distanza, ridurre al minimo i problemi e ottenere il maggior di velocità il più possibile. Una frequenza di 600 MHz, ad esempio, ha una larghezza di banda inferiore, ma, poiché non è influenzata facilmente da fattori come l'umidità nell'aria, non perde potenza così rapidamente ed è in grado di raggiungere i telefoni 5G e altri dispositivi 5G più distanti, oltre che di penetrare meglio nei muri per fornire una ricezione interna. Un fornitore di servizi potrebbe utilizzare frequenze 5G più elevate nelle aree che richiedono più dati, come ad esempio in una città popolare dove ci sono molti dispositivi in uso. Tuttavia, le frequenze a banda bassa sono utili per fornire l'accesso 5G a un maggior numero di dispositivi da una singola torre e a aree che non hanno una linea di vista diretta con una cella 5G.

2.4.4 Low-PowerWide-Area Network

La LPWAN è caratterizzata da un basso consumo di energia e un raggio d'azione piuttosto lungo. Essa utilizza principalmente la banda sub-GHz per ottenere elevata robustezza al rumore, ma offre una bassa velocità di trasmissione dei dati. Le reti LPWAN offrono un lungo range tra i 5 e i 10 km, a scapito di una piccola larghezza di banda (dati inviati) Questo è legato alla loro elevata robustezza ai disturbi e alla loro frequenza principale. Questo porta a velocità di trasmissione dei dati di alcuni kbps. La LPWAN è molto adatta alle applicazioni IoT che devono trasmettere solo piccole quantità di dati piccole quantità di dati a lungo raggio. È altamente efficiente dal punto di vista energetico e poco costosa. Alcune di queste tecnologie sono usate nell'ambito dell'IoT, e sono Sigfox, LoRa e NB-IoT. Le prime due offrono comunicazioni, poco frequenti, a lunghissimo raggio e durata della batteria molto lunga. Al contrario, NB-IoT offre una latenza molto bassa e alta qualità del servizio.

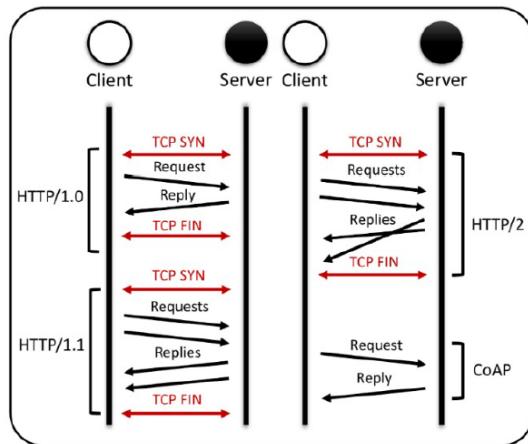
2.4.5 IPV6

Internet Protocol Version 6 (IPv6) è la versione migliorata di IPv4 e può supportare un numero molto elevato di nodi rispetto all'IPv4. Consente 2128 possibili combinazioni di nodi o indirizzi. Semplifica alcune funzioni di rete, come il routing e la mobilità e offre migliori opzioni di sicurezza grazie a una migliore progettazione e a una gestione più calibrata rispetto a IPSec, la componente di sicurezza del protocollo IPv4. A causa dei limiti dello spazio di indirizzi IPv4, l'attuale Internet utilizza Network Address Translation (NAT). La comunità di ricerca sull'IoT ha sviluppato una versione compressa di IPv6 denominata 6LoWPAN, con un meccanismo semplice ed efficiente per accorciare le dimensioni dell'indirizzo IPv6, mentre i router di confine possono tradurre gli indirizzi compressi in indirizzi IPv6 regolari. Parallelamente, sono stati sviluppati piccoli stack come Contiki, che occupa non più di 11,5 Kbyte. Grazie al suo ampio spazio di indirizzi, l'IPv6 consente di estendere Internet a qualsiasi dispositivo e servizio. Fornisce un meccanismo di autoconfigurazione degli indirizzi (meccanismo Stateless). I nodi possono definire i propri indirizzi in modo autonomo. Questo consente di ridurre drasticamente lo sforzo e i costi di configurazione.

2.4.6 Comunication protocols

I protocolli di comunicazione candidati si differenziano per i modelli di interazione. Il modello di comunicazione request-reply è uno dei paradigmi di comunicazione più basici: uno scambio di messaggi in cui un client richiede informazioni a un server che riceve il messaggio di richiesta, lo elabora e restituisce un messaggio di risposta. Questo tipo di informazioni è solitamente gestito e scambiato a livello centrale, e i due protocolli più conosciuti basati sul modello richiesta/risposta

sono REST HTTP e CoAP. Per Http 1.0 posso inviare una sola richiesta, l'1.1 introduce richieste multiple da parte del client e il server risponderà in ordine di arrivo. Il 2.0 introduce un modello in cui le richieste e le risposte possono essere mandate in modo asynchronous via connessione TCP. La sessione TCP va inizializzata e chiusa. CoAP non possiede Three-way handshake.



Il modello publish-subscribe, d'altra parte, è emerso dalla necessità di fornire un sistema di comunicazione distribuito, asincrono e non vincolato tra i generatori di dati e le destinazioni. Il broker è un intermediario, riceve i messaggi e li smista verso chi ha espresso la propria preferenza (subscribe) verso quel tipo di messaggio e verso il contenuto del messaggio.

Grazie al ruolo di mediatore del Notification Service, publisher and subscriber sono fortemente disaccoppiati. Non hanno alcun obbligo di conoscersi e non è necessario che siano attivi nello stesso momento per ricevere le notifiche. Questo migliora la scalabilità.

2.5 Protocolli

2.5.1 HTTP

L'Hyper Text Transport Protocol (HTTP) è il protocollo client-server fondamentale utilizzato per il Web e quello più compatibile con l'infrastruttura di rete esistente. Un client invia un messaggio di richiesta HTTP e il server restituisce quindi un messaggio di risposta, contenente la risorsa richiesta nel caso in cui la richiesta sia stata accettata. Sebbene in generale HTTP presenti una delle opzioni di protocollo più stabili, ci sono ancora alcuni problemi che hanno portato all'esplorazione di soluzioni di protocollo alternative, a causa della complessità HTTP, dei campi di intestazione lunghi e dell'elevato consumo energetico. Inoltre, HTTP utilizza il paradigma richiesta/risposta, che non è adatto per le notifiche push, in cui il server invia notifiche al client senza una richiesta del client. HTTP/2.0 ha introdotto una serie di miglioramenti, alcuni dei quali sono particolarmente rilevanti nel contesto IoT:

- un uso più efficiente delle risorse di rete e una latenza ridotta introducendo intestazioni compresse, utilizzando un formato di compressione della memoria molto efficiente e basso, oltre a consentire più scambi simultanei sulla stessa connessione.

2.5.2 COAP

Il Constrained Application Protocol (CoAP) è simile a HTTP, con l'uso di un'architettura REST collaudata e ben accettata e il supporto del paradigma richiesta/risposta. CoAP è considerato un protocollo leggero, quindi le intestazioni, i metodi e i codici di stato sono tutti codificati in binario, riducendo così l'overhead del protocollo rispetto a molti protocolli. Prevede un livello per la ritrasmissione dei messaggi persi.

2.5.3 MQTT

Message Queue Telemetry Transport Protocol (MQTT) è uno dei protocolli di messaggistica leggeri che segue il paradigma di pubblicazione e sottoscrizione, che lo rende piuttosto adatto per dispositivi con risorse limitate e per condizioni di connettività di rete non ideali. MQTT funziona sopra il protocollo di trasporto TCP, garantendo affidabilità, ma grazie alla sua intestazione più leggera, MQTT ha requisiti energetici molto inferiori rispetto a HTTP, rendendolo una delle soluzioni di protocollo più importanti in ambienti limitati. MQTT definisce tre livelli di QoS. Un'altra caratteristica importante offerta da MQTT è la possibilità di memorizzare alcuni messaggi per i nuovi abbonati impostando un flag di "conservazione" nei messaggi pubblicati.



Privacy vs. Security: What's the Difference?

3. Security and Privacy for IoT

3.1 Security and Privacy

I sistemi basati sull'IoT possono gestire enormi quantità di informazioni e possono essere utilizzati per servizi che spaziano dalla gestione industriale al monitoraggio della salute. Questo ha reso il paradigma dell'IoT un obiettivo interessante per una moltitudine di aggressori e avversari. I potenziali aggressori potrebbero essere interessati a rubare informazioni sensibili e/o compromettere i componenti IoT. Possono sfruttare i dispositivi infetti per spiare una persona di interesse o per un attacco su larga scala.

Il numero di minacce alla sicurezza che colpiscono i dispositivi IoT è aumentato negli ultimi anni, ci sono moltissime vulnerabilità. Data la penetrazione sempre più ampia dell'IoT nelle intere attività quotidiane e delle infrastrutture critiche, il verificarsi di incidenti di sicurezza informatica è destinato ad avere un tasso di crescita. Alla nascita dell'IoT non erano presenti tecniche di sicurezza.

La sicurezza, in generale, tradizionalmente si basa 3 macro categorie:

- **Confidenzialità:** i dati devono essere disponibili solo per gli utenti autorizzati
- **Integrità:** garantire che non vi è manipolazione dei dati
- **Disponibilità:** assicura che i dispositivi siano disponibili a catturare dati e il sistema deve essere in grado di difendersi da attacchi intenzionali.

La CIA-triad ha proposto una estensione dove troviamo anche:

- **Responsabilità:** la capacità di un sistema di ritenere gli utenti responsabili delle loro azioni;
- **Auditability:** la capacità di un sistema di condurre un monitoraggio di tutte le azioni;
- **Affidabilità:** La capacità di un sistema di verificare l'identità e stabilire la fiducia di una terza parte;
- **Non ripudio:** La capacità di un sistema di confermare il chi ha compiuto una determinata azione;
- **Privacy:** Garantire che il sistema rispetti le politiche sulla privacy e di consentire agli individui di controllare le proprie informazioni personali.

Una vulnerabilità in alcuni aspetti o caratteristiche di un sistema rende possibile un exploit o un danno. Ciò può essere dovuto a una cattiva progettazione, errori di configurazione o di tecniche di codifica inadeguate e insicure. Possono essere identificate e corrette. Un attacco è un'azione che sfrutta una o più vulnerabilità realizzando una minaccia, ovvero un effetto negativo o un evento indesiderato che potrebbe danneggiare o compromettere una risorsa o un obiettivo.

SECURITY VERSUS PRIVACY	
Security refers to protection against unauthorized access.	Privacy defines the ability to protect personally identifiable information.
Security provides protection for all types of data and information including the ones that are stored electronically.	Privacy means protecting sensitive information related to individuals and organizations.
Security can be achieved without privacy.	Privacy cannot be achieved without security.
Security program focuses on all sorts of information assets that an organization collects.	Privacy program focuses on personal information such as names, addresses, social security numbers, log in credentials, financial accounts information, etc.
It implements security protocols to provide confidentiality, integrity and availability of information assets.	It refers to protection of privacy rights with respect to processing of personal data.

In Europa il Regolamento UE 2016/679 (GDPR) stabilisce i fattori chiave per proteggere i cittadini dell'Unione dall'uso incontrollato dei loro dati personali, illustrati nel capitolo II (articolo 5.1):

- La legittimità dei dati in possesso: I dati sono trattati in modo lecito, corretto e trasparente nei confronti dell'interessato;
- Finalità limitata: I dati sono raccolti per scopi specifici, esplicativi e legittimi;
- Minimizzazione: I dati raccolti devono essere pertinenti e rilevanti e limitati a quanto necessario in relazione alle finalità per le quali sono stati raccolti e trattati;
- Accuratezza: I dati personali raccolti saranno accurati e, se necessario, aggiornati;
- Conservazione limitata: I dati richiesti devono essere conservati per un periodo di tempo limitato che non superi il raggiungimento delle finalità per cui sono trattati;
- Integrità e riservatezza: I dati sono trattati in modo da garantire un'adeguata sicurezza, compresa la protezione attraverso adeguate misure tecniche e organizzative, da trattamenti non protetti o illeciti e dalla perdita, distruzione o danneggiamento accidentale.

Alcuni di questi principi vanno contro l'IoT perché i sensori monitorano e conservano tutti i dati.

3.2 Privacy type

Privacy by design e privacy by default sono due principi fondamentali della protezione dei dati personali e della privacy e sono stati imposti dal GDPR.

Privacy by design si riferisce alla pratica di progettare prodotti, servizi e sistemi informatici fin dalle prime fasi del loro sviluppo, tenendo conto della protezione della privacy e della sicurezza

dei dati personali come elementi fondamentali. In pratica, significa che la privacy deve essere integrata in ogni fase del processo di progettazione, dallo sviluppo dell'idea alla sua realizzazione e successiva manutenzione. In questo modo, i prodotti e i servizi che vengono creati sono intrinsecamente più sicuri e rispettosi della privacy.

Privacy by default, d'altra parte, si riferisce alla scelta predefinita delle impostazioni di privacy più protettive possibili. Questo significa che, quando un utente accede a un nuovo servizio o prodotto, le impostazioni di privacy dovrebbero essere preimpostate in modo che il livello di protezione dei dati personali sia il più alto possibile, a meno che l'utente non scelga di modificarle. In questo modo, l'utente non è costretto a fare scelte consapevoli per proteggere la propria privacy, ma ha già un alto livello di protezione preimpostato di default.

In sintesi, mentre privacy by design si concentra sulla progettazione di prodotti e servizi rispettosi della privacy fin dalla loro concezione, privacy by default si riferisce alla scelta predefinita delle impostazioni di privacy più protettive possibile. Entrambi i principi sono importanti per garantire la protezione dei dati personali e della privacy degli utenti.

3.3 IoT Vulnerabilities and Attacks

Abbiamo vari tipi di vulnerabilità:

- **Deficient physical security:** Con poco sforzo, un avversario potrebbe ottenere accesso fisico non autorizzato ai dispositivi IoT e quindi prenderne il controllo. Di conseguenza, un attaccante potrebbe causare danni fisici ai dispositivi.
- **Insufficient energy harvesting:** I dispositivi IoT hanno tipicamente un'energia limitata e non possiedono tecnologia o i meccanismi per rinnovarla automaticamente. Un aggressore potrebbe prosciugare l'energia accumulata generando messaggi legittimi o corrotti, rendendo i dispositivi non disponibili per processi o utenti validi.
- **Inadequate authentication:** I vincoli unici di energia e potenza computazionale limitate mettono a dura prova l'implementazione di meccanismi di autenticazione complessi. In tali circostanze, le chiavi di autenticazione scambiate e utilizzate sono sempre a rischio di perdita, distruzione o corruzione.
- **Improper encryption:** Le limitazioni delle risorse dell'IoT influenzano la robustezza, l'efficienza e l'efficacia degli algoritmi di crittografia. A tal fine, un utente malintenzionato potrebbe essere in grado di aggirare le tecniche di crittografia utilizzate per rivelare informazioni sensibili o controllare le operazioni con uno sforzo limitato, e fattibile, dato che le comunicazioni wireless sono facili da intercettare. Bisogna effettuare dei trade-off. I meccanismi di sicurezza consumano moltissima energia.
- **Improper patch management capabilities:** I sistemi operativi IoT e il firmware/software incorporato devono essere patchati in modo da ridurre al minimo i vettori di attacco e aumentare le capacità funzionali. Tuttavia, molti produttori non mantengono periodicamente le patch di sicurezza o non dispongono di meccanismi automatici di aggiornamento delle patch.
- **Weak programming practices:** Vengono rilasciati innumerevoli firmware con vulnerabilità note come le backdoor, mancanza di punti di accesso e la mancanza dell'uso del Secure Socket Layer (SSL). Di conseguenza, un avversario potrebbe facilmente sfruttare le debolezze di sicurezza conosciute per causare buffer overflow, modifiche di informazioni o per ottenere un accesso non autorizzato al dispositivo.

3.3.1 Attacks against IoT

Gli attacchi contro la riservatezza e l'autenticazione mirano a ottenere un accesso non autorizzato alle risorse e ai dati IoT per condurre ulteriori azioni dannose. Questo tipo di attacco è spesso indotto dall'esecuzione di eventi di forza bruta, dall'intercettazione di misure fisiche IoT IoT o falsificando l'identità dei dispositivi.

Abbiamo diversi tipi di attacchi:

- **Dictionary attacks:** consistono in varianti di eventi di forza bruta, che portano a modifiche illecite delle impostazioni o addirittura al pieno controllo delle funzioni delle funzioni del dispositivo. Forza bruta di solito richiede molto tempo, nel mondo dell'Iot invece è la soluzione migliore per attaccare poichè l'hardware è semplice, la memoria è limitata, la lunghezza della chiave è limitata quindi non ci impiegherà molto tempo la forza bruta.

Un esempio è Mirai che è un malware che trasforma dispositivi in rete che eseguono Linux in bot controllati a distanza che possono essere utilizzati come parte di una botnet in attacchi di rete su larga scala. I dispositivi infettati da Mirai effettuano una scansione continua di Internet alla ricerca dell'indirizzo IP dei dispositivi IoT. Mirai identifica poi i dispositivi IoT vulnerabili utilizzando una tabella di oltre 60 nomi utente e password di fabbrica comuni e vi accede per infettarli con il malware Mirai. I dispositivi infetti continueranno a funzionare normalmente, ad eccezione di occasionale lentezza e un maggiore utilizzo della larghezza di banda. Un dispositivo rimane infetto fino a quando non viene riavviato, cosa che può comportare semplicemente spegnere il dispositivo e dopo una breve attesa riaccenderlo. Dopo un riavvio, a meno che non si cambi immediatamente la password di accesso, il dispositivo verrà reinfectato nel giro di pochi minuti. Al momento dell'infezione Mirai identifica qualsiasi malware "concorrente", lo rimuove dalla memoria e blocca le porte di amministrazione remota.

- **Side-channel attacks:** tenta di recuperare le chiavi crittografiche dei dispositivi sfruttando le correlazioni esistenti tra le misure fisiche e gli stati interni dei dispositivi IoT, piuttosto che sulle debolezze dell'algoritmo implementato (ad es, la crittoanalisi).
- **Sybil attacks:** consiste nel fatto che l'attaccante sovverte il sistema di reputazione di una rete peer-to-peer creando un gran numero di identità pseudonime e le utilizza per ottenere un'influenza sproporzionata. Abbiamo un meccanismo di fiducia fra i nodi. I nodi sybil possono essere esterni in gruppo, creano connessioni intercettando messaggi e leggendo il loro contenuto, oppure possono mischiarsi tra i vari nodi con l'obiettivo di violare la privacy e le informazioni degli utenti e manipolare la reputazione del sistema.

Gli attacchi contro l'integrità dei dati consistono nel sabotaggio dei dati IoT.

- **False Data Injection (FDI):** l'attaccante intercetta messaggi onesti e inietta informazioni false alterando lo stato del sistema per causare diverse violazioni dell'integrità. Il lancio di tali attacchi potrebbe fuorviare il processo di stima dello stato di un dispositivo IoT, causando un drammatico impatto economico o addirittura la perdita di vite umane.
- **Modifica del firmware:** mira ad alterare maliziosamente il firmware e indurre un'interruzione funzionale del dispositivo preso di mira. Molti firmware vengono rilasciati con vulnerabilità conosciute, e il tasso di patch che vengono rilasciate è piuttosto basso, il che permette continui attacchi. Effettuando il reverse engineering, l'attaccante può determinare i dettagli del sistema operativo, estrarre la funzionalità di varie routine critiche e individuare le strutture chiave da modificare. L'attaccante fa accesso al firmware e invia il pacchetto malizioso all'utente che vedrà il pacchetto come un aggiornamento.

Gli attacchi contro la disponibilità, o attacchi **Denial of Service (DoS)** contro l'IoT, consistono nell'impedire agli utenti legittimi di accedere tempestivamente a risorse IoT. Questo viene spesso indotto revocando l'accesso del dispositivo alla rete o prosciugando le risorse IoT fino al loro completo esaurimento. Abbiamo attacchi del tipo:

- Di tipo "**sinkhole**": condotti da nodi malintenzionati che hanno capacità di pubblicizzare percorsi di instradamento artificiali in cui agiscono come mediatori tra i nodi onesti. I nodi maligni possono far cadere i pacchetti o inviarli su percorsi più lunghi. Un modo per risolvere ciò è rimuovere quei nodi che hanno una grande reputazione, poiché sono proprio questi i nodi infetti che cercano di intercettare tutte le informazioni. Questo attacco ha effetti sulla disponibilità e sulle performance.
- **Battery Draining**: sono definiti come la trasmissione di un messaggio richiedente una quantità di energia significativamente maggiore da parte della rete e dei suoi nodi per essere impiegato e agito, a differenza di messaggi tipici. Per identificare ciò, posso osservare il comportamento dei nodi e vedere quale sta inviando troppi messaggi. Carousel attacks è quando l'attaccante manda messaggi in loop, un messaggio verrà instradato nello stesso nodo più volte. Stretch attacks è quando l'attaccante manda false informazioni sul percorso ottimale da intraprendere per mandare un messaggio attraverso i nodi, in modo da raggiungere la destinazione utilizzando più nodi di quelli che servirebbero realmente. Gli attacchi di disturbo (jamming) mirano a interrompere le comunicazioni di rete IoT e ridurre la durata dei nodi con vincoli energetici creando interferenze e provocando collisioni di pacchetti.

La maggior parte dei dispositivi IoT prolunga il proprio ciclo di vita seguendo una routine di **sleep** per ridurre il consumo di energia.

3.4 Privacy Issues

L'identificazione delle informazioni personali durante la trasmissione su Internet è un problema serio e dipende dal contesto e dall'applicazione (ad esempio la temperatura non è qualcosa che è relativo alla privacy, ma può dare informazioni sui proprietari e l'attaccante può ricostruire abitudini dei soggetti). L'eventuale fuga di informazioni dell'utente può comportare minacce alla privacy in termini di tracciamento, localizzazione e personalizzazione, in quanto consente la profilazione e il tracciamento dell'utente. L'IoT ha una prospettiva data centrica (prospettiva incentrata sui dati), quindi i tre fattori critici sono **la scalabilità, l'elaborazione distribuita e l'analisi in tempo reale**. Altri problemi della privacy in questo settore riguardano la pubblicazione dei dati, il contesto delle applicazioni, i problemi di utilità, la crittografia. Alcuni problemi di privacy:

- Durante la raccolta di grandi insiemi di dati grezzi, è difficile bilanciare la conservazione della privacy nella pulizia dei dati e la riduzione intenzionale della qualità dei dati e dello scopo originale senza perdere le informazioni necessarie per il data mining e l'analisi.
- I dati raccolti potrebbero essere utilizzati e pubblicati per scopi diversi da quello originario senza il consenso dell'utente.
- Poiché i supporti di archiviazione per computer possono archiviare grandi volumi di dati, offrono un'elevata disponibilità a basso costo. Di conseguenza, una volta che le informazioni sono state generate, è molto probabile che vengano archiviate all'infinito e quindi l'"oblio digitale" può portare a violazioni della privacy dal punto di vista dei proprietari dei dati.

3.5 IoT Forensics

Con la proliferazione di dispositivi connessi che aprono nuove strade per tracciare e raccogliere dati, il confine tra "servire la giustizia" e proteggere la privacy è diventato sfocato. Il problema dei dispositivi IoT è che, per funzionare correttamente e soddisfare le esigenze dei loro utenti, devono raccogliere ed elaborare grandi quantità di dati personali sensibili sulle nostre vite, preferenze e attività quotidiane. Inutile dire che se quei dati venissero in qualche modo compromessi o finissero nelle mani sbagliate, sarebbe ovviamente un'enorme invasione della privacy personale. Ma cosa succede se i dati archiviati sui nostri dispositivi potrebbero potenzialmente contenere prove chiave in un procedimento penale?

3.5.1 Introduzione

La disciplina della Digital Forensics (DF) è una branca della scienza forense tradizionale. Riguarda la scoperta e l'interpretazione di dati elettronici. I professionisti di DF si occupano dell'identificazione, raccolta, recupero, analisi e conservazione delle prove digitali, presenti su vari tipi di dispositivi elettronici, senza contaminare queste prove e i dispositivi. Sebbene ci siano alcune variazioni nel modo in cui i diversi studiosi dividono il ciclo di indagine in fasi, non dovrebbe mai mancare un dettaglio importante: l'intero ciclo dovrebbe essere eseguito utilizzando strumenti convalidati e metodologie scientificamente provate. L'IoT Forensics potrebbe essere percepita come una suddivisione del DF e un'area relativamente nuova e inesplorata, per identificare ed estrarre informazioni digitali in modo legale e forense. Oltre che da un particolare dispositivo o sensore IoT, i dati forensi potrebbero essere raccolti dalla rete interna (ad esempio un firewall o un router) o dal cloud. Si potrebbe vedere una differenza fondamentale tra DF e IoT Forensics. A differenza del tradizionale DF, dove i soliti oggetti di esame sono computer, smartphone, tablet, server o gateway e il tutto si limita alla scena, in IoT Forensics le fonti di prova potrebbero essere molto più ampie (e potrebbero essere ovunque quindi difficili da localizzare), inclusi i sistemi di monitoraggio di pazienti, Infotainment dei veicoli, semafori e persino impianti medici in esseri umani e animali, quindi l'IoT Forensics non si limita alla scena. Se pensiamo al cloud, le evidenze potrebbero essere lontane dalla scena e trovarsi nel cloud. Proteggere ogni singolo sensore, dispositivo di comunicazione e archiviazione cloud all'interno della rete IoT è quasi impossibile. Se si verifica un incidente, una delle prime attività eseguite dai professionisti della scientifica è definire l'ambito della compromissione. Tuttavia, a differenza delle pratiche di sicurezza IoT, le tecniche forensi non mirano a ridurre al minimo il danno, ma a identificare l'origine dell'attacco/deficit o le responsabilità delle diverse parti.

3.5.2 Sfide

Il campo dell'IoT Forensics sta affrontando una serie di sfide, nessuna delle quali ha una soluzione semplice. La prima e probabilmente la parte più essenziale di qualsiasi esame forense è la ricerca di prove. L'identificazione nel contesto IoT è particolarmente difficile, poiché gli esaminatori potrebbero non sapere nemmeno dove sono fisicamente archiviati i dati esaminati. Anche un semplice compito come trovare il dispositivo IoT compromesso e ricostruire la scena del crimine potrebbe essere impegnativo. Ci sono diverse sfide:

- Ambito del compromesso e ricostruzione della scena del crimine: Il contesto IoT, tuttavia, implica un'interazione in tempo reale e autonoma tra i vari nodi, il che rende quasi impossibile ricostruire la scena del crimine e identificare l'entità del danno, a causa della natura altamente dinamica della comunicazione. I dati potrebbero essere archiviati su diverse macchine virtuali (VM), il che significa che importanti dati forensi come voci di registro o file temporanei

potrebbero essere completamente cancellati non appena la VM viene riavviata o spenta.

- **Proliferazione dei dati:** il numero crescente di dispositivi interconnessi e la quantità di dati forensi digitali che richiedono analisi sono stati discussi per molti anni. I professionisti forensi non devono solo identificare ciò che è utile per l'indagine, ma anche scartare i dati irrilevanti, il che rende difficile l'analisi tempestiva.
- **Posizione dei dati:** durante il funzionamento, i dispositivi IoT potrebbero migrare frequentemente tra diverse posizioni fisiche. Pertanto, quando tentano di individuare le prove, i professionisti della digital forensics devono affrontare sfide considerevoli. Anche se la posizione è nota, l'acquisizione del sistema non è priva di complicazioni perché potrebbe interessare altri clienti che utilizzano la stessa architettura. Inoltre, le risorse possono essere soggette a più giurisdizioni con numerose e persino contraddittorie normative sulla protezione dei dati e intrusioni non autorizzate.
- **Tipo di dispositivo:** in contrasto con la tradizionale scienza forense digitale, in cui gli oggetti di interesse forense sono solitamente limitati a diversi tipi di sistemi informatici o telefoni cellulari, la fonte di prove nei casi incentrati sull'IoT potrebbe essere eterogenea.

Partendo dal presupposto che il dispositivo IoT pertinente sia stato identificato con successo, il secondo passaggio sarebbe quello di raccogliere i dati delle prove. Tuttavia, a questo punto gli investigatori devono affrontare un altro problema: fino al momento attuale, non esiste una guida o un metodo standardizzato per la raccolta di prove da un dispositivo IoT in modo forense.

- **Mancanza di formazione:** le forze dell'ordine dovrebbero organizzare programmi di formazione per i loro primi soccorritori al fine di istruirli su come acquisire prove digitali in modo forense. Ciò rende l'acquisizione di prove da dispositivi IoT uno dei passaggi più trascurati nella pratica.
- **Specifiche software e/o hardware eterogenee:** ogni produttore adotta hardware e sistemi operativi differenti. I dati delle prove possono essere archiviati in modo crittografato o in un formato non standard per il quale attualmente non esiste un visualizzatore applicabile.
- **Privacy e considerazioni etiche relative all'accesso ai dati personali:** al di là delle sfide tecniche, la privacy è una questione importante da considerare durante la raccolta dei dati. I dispositivi IoT trattano informazioni personali sensibili, comprese le cartelle cliniche, le prescrizioni o lo stato di salute attuale degli utenti, e possono anche contenere dati di clienti che non sono collegati all'indagine.

Nel caso in cui gli investigatori trovino un dispositivo potenzialmente compromesso riescano a raccogliere dati potenzialmente utili, dovranno affrontare un'altra sfida: come preservare i dati raccolti e garantirne l'integrità.

- **Garantire la catena di custodia:** il termine "catena di custodia" potrebbe essere definito come l'accurato controllo delle prove. Nella medicina legale convenzionale, inizia quando gli investigatori raccolgono un elemento di prova sulla scena del crimine e termina con la presentazione del materiale di prova in tribunale. La catena di custodia fornisce informazioni chiare su quando e come le prove sono state raccolte, conservate, analizzate e presentate. Inoltre, dimostra che il materiale probatorio non è stato alterato o modificato durante tutte le fasi dell'indagine forense. Nel caso di IoT Forensics, i dati delle prove devono essere raccolti da più server remoti, il che complica notevolmente la missione di mantenere un'adeguata catena di custodia. A causa della sua natura immateriale, le prove digitali sono particolarmente vulnerabili alla manipolazione e, pertanto, devono essere ampiamente documentate e protette.

durante tutte le fasi del processo forense. Utilizzando la blockchain una certa informazione digitale può essere conservata e indirizzata verso la sua destinazione finale, il tribunale.

- **I problemi forensi del cloud:** la sinergia tra cloud e IoT è emersa perché il cloud possiede attributi che abilitano e avvantaggiano l'espansione dell'IoT. Tuttavia, il cloud è costituito da un'enorme quantità di problemi di sicurezza. Pertanto, i dati conservati nel cloud hanno un valore forense limitato, poiché potrebbero essere stati alterati da un utente malintenzionato che ha sfruttato le vulnerabilità. Naturalmente, nessun sistema è immune agli attacchi.

3.5.3 IoT Forensics Approaches

All'interno dell'IoT, le tecniche investigative tradizionali hanno un tasso di successo molto basso. Ci sono vari quadri teorici tra cui scegliere, anche se tutti adottano fasi principali simili. L'approccio 1-2-3 zone potrebbe non essere tra gli sviluppi più recenti, ma è forse il framework teorico più citato nella scienza forense digitale. Il metodo riduce la complessità e la tempistica delle indagini, il che significa che le autorità possono concentrare la loro attenzione su compiti sostanziali e, così facendo, ottenere una maggiore efficienza.

Il metodo divide l'IoT Forensics in tre zone:

- **Prima zona:** copre l'intera area composta da hardware, software e reti. Qui vengono raccolte le prime prove.
- **Seconda zona:** Quest'area comprende principalmente dispositi e infrastrutture pubblici, sistemi di prevenzione e rilevamento delle intrusioni e firewall di rete. Durante l'indagine, è necessario raccogliere il massimo delle prove richiedendo l'assistenza dei rispettivi fornitori.
- **Terza zona:** Copre tutto l'hardware e software che si trova al di fuori della rete in questione come cloud e internet service provider.

Gli approcci della digital forensics non dovrebbero essere utilizzati solo nelle attività post-incidente, ma anche per aumentare le possibilità di ottenere buoni risultati o spendere meno risorse in indagini future. I criminali informatici sono consapevoli del modo in cui funzionano gli strumenti di digital forensics. Man mano che i meccanismi difensivi diventano sempre più efficienti, ci si dovrebbe aspettare tecniche di crittografia e offuscamento ancora più sofisticate.

TOP IoT SECURITY PRODUCTS



4. Secure IoT Product Design & Coding

@ImaginovationCo

imaginovation
IMAGINATION TURNS TO INNOVATION

La sicurezza del prodotto è fondamentale nello sviluppo e nella manutenzione di sistemi connessi sia nel mercato industriale che in quello consumer. Il processo deve iniziare con una politica di sicurezza che fornisca le basi per l'applicazione e l'adesione alle legislazioni di sicurezza informatica appropriate che si applicano al sistema IoT. La sicurezza operativa e dell'infrastruttura si basa su dispositivi intrinsecamente sicuri. Perché ciò avvenga, la base per infrastrutture IoT robuste e affidabili dipende dalla produzione e dalla sicurezza del dispositivo.

4.0.1 Introduzione

Il costo della correzione di un bug di sicurezza varia a seconda di dove viene scoperto e quando. Se viene scoperto nell'ambiente di produzione, il costo per risolverlo includerebbe i costi tangibili di tutti gli attori coinvolti e il costo intangibile della reputazione e della fiducia dei clienti. Se viene scoperto durante la fase di progettazione, è molto facile correggere il difetto di progettazione e introdurre una misura di sicurezza durante la fase di sviluppo. Il costo della correzione di un difetto in post-produzione è circa quattro volte superiore rispetto alla correzione in fase di sviluppo. Bisogna trovare e scovare il bug il prima possibile.

La garanzia della sicurezza per le applicazioni IoT può essere raggiunta al meglio attraverso l'adozione di una strategia di difesa in profondità, che, a sua volta, garantisce l'adozione di una pratica SDLC (Secure Development Life Cycle). L'intento principale è quello di creare la sicurezza all'interno del ciclo di vita di queste applicazioni da zero, che potenzialmente e gradualmente riduca i difetti di sicurezza, progettazione, implementazione e distribuzione. La corretta adesione a tali migliori pratiche di garanzia si tradurrà in applicazioni prive di vulnerabilità che potrebbero essere state introdotte accidentalmente o intenzionalmente in qualsiasi momento del loro ciclo di vita.

Lo sviluppo "agile" è ampiamente utilizzato nei progetti relativi all'IoT e presenta diverse varianti, come Scrum, XP e Kanban. L'idea alla base della sicurezza in agile è che i requisiti di sicurezza sono guidati dalla frequenza e non dalla fase.

4.0.2 Criticality Assessment (Valutazione della criticità)

Un vettore di attacco descrive i passaggi che un utente malintenzionato intraprenderà per realizzare una minaccia. L'avversario rappresenta l'attore dell'attacco. È l'entità il cui vero obiettivo è causare danni a un sistema bersaglio. E' importante capire la natura dell'attaccante, riconoscere la sua preparazione, le sue capacità, i suoi strumenti, le motivazioni dell'attacco. Essendo nella maggior parte dei casi l'anello più debole della catena di sicurezza, un dispositivo IoT sarà solitamente utilizzato dall'avversario come punto di ingresso iniziale, per ottenere l'accesso ai servizi critici.

- Abbiamo due tipi di accesso all'IoT:
 - **Accesso fisico:** Un insider è un avversario che ha accesso fisico diretto, o caratterizzato dalla vicinanza fisica, al dispositivo IoT. Un estraneo non ha accesso fisico diretto o vicinanza al dispositivo IoT di destinazione, ma può tentare di acquisire conoscenza manomettendo un altro dispositivo IoT dello stesso tipo.
 - **Accesso logico:** Gli avversari con accesso privilegiato possono connettersi logicamente al dispositivo IoT tramite un'interfaccia disponibile. In generale, è meno probabile che si verifichino attacchi che richiedono un accesso logico privilegiato al dispositivo IoT, poiché l'avversario dovrà aggirare i controlli di autorizzazione.
- **Capabilities** è caratteristica che modella le abilità e le risorse richieste da un avversario per attaccare con successo il sistema bersaglio. Alcuni tipi di attacchi possono essere effettuati solo da esperti che hanno determinate capacità.
- **Motivation** descrive il potenziale guadagno che un avversario trarrebbe vantaggio da un attacco riuscito, in combinazione con la penalità prevista per un avversario che viene rintracciato. È vista come la motivazione reale dell'attacco effettuato, ad esempio per soldi, per fama, per intrattenimento, e questa motivazione varia in base all'attacco.

Un attacco alla sicurezza può essere valutato in base al rischio per la sicurezza che può causare a un sistema di destinazione. Il livello di minaccia misura la misura in cui un sistema è minacciato dall'attacco. Il livello di vulnerabilità misura i punti deboli sfruttati da un avversario per realizzare l'attacco. Il livello di impatto rappresenta il potenziale danno che sarebbe causato dall'attacco, le sue conseguenze.

4.1 IoT Product Development Process

4.1.1 Security Requirements

I requisiti sono alla base del ciclo di sviluppo dell'IoT e riflettono l'uso previsto del software e verranno tradotti in specifiche che guideranno le decisioni di progettazione, sviluppo e manutenzione/implementazione in una fase successiva. La sicurezza deve essere considerata fin da questa prima fase di sviluppo del software, per garantire che la sicurezza non arrivi come un ripensamento. Poiché software e hardware sono strettamente correlati, i requisiti del software di sicurezza possono avere determinate implicazioni nella selezione del supporto fisico (hardware). I gateway ricevono documenti e controllano se sono applicabili e considerabili o no, ciò dipende dal contesto dell'applicazione, dai dati.

Con l'IoT, il mondo digitale e quello fisico non sono più separati l'uno dall'altro e questa natura fisica introduce minacce aggiuntive. Qualcosa nel mondo virtuale può avere effetti nel mondo fisico. È inoltre necessario un criterio di aggiornamento per proteggere il dispositivo dagli attacchi di rollback del software, che possono verificarsi se le versioni software precedenti contengono una vulnerabilità di sicurezza.

4.1.2 Analisi delle minacce alla sicurezza

Una volta presa in considerazione l'architettura del prodotto IoT per quanto riguarda connettività e hardware, è importante considerare le possibili minacce al prodotto IoT. Un'analisi delle minacce fa parte del processo di sviluppo e considera i vettori di attacco che potrebbero essere disponibili per i malintenzionati. L'analisi delle minacce e il restante processo di sviluppo aiuteranno nella scelta degli elementi hardware da utilizzare nella progettazione. In questa fase, per l'analisi delle minacce, considero molti aspetti e ci chiediamo quale sistema operativo è usato, se è presente il cloud, quali sono i dispositivi, qual è la comunicazione ecc ecc.

Abbiamo sei categorie di minacce al software:

- **Spoofing:** utilizzare le credenziali di qualcun altro (contraffatte o rubate) per guadagnare accesso a beni altrimenti inaccessibili.
- **Manomissione:** modifica i dati per sferrare un attacco. In questa situazione, un utente malintenzionato potrebbe alterare i file, violando così la sicurezza del sistema.
- **Ripudio:** si verifica quando un utente nega di aver eseguito un'azione.
- **Divulgazione di informazioni:** la divulgazione di informazioni a un utente che non ha il permesso di accedervi.
- **Denial of Service:** gli attacchi minacciano la capacità di accesso degli utenti validi a risorse.
- **Elevazione del privilegio:** un attacco che può verificarsi se un utente senza privilegi ottiene lo stato privilegiato.

4.1.3 Strategia di sviluppo

Una volta definiti i requisiti di sicurezza e identificate le minacce alla sicurezza, è importante che lo sviluppatore del prodotto IoT crei un piano di implementazione della sicurezza che identifichi le credenziali di sicurezza critiche specifiche per il prodotto IoT, nonché altri elementi di sicurezza critici come il ciclo di vita e il development process.

Le fasi chiave del flusso di sviluppo del prodotto sono:

- **Identity:** grazie a un'identità del dispositivo forte e univoca, un dispositivo IoT può essere autenticato quando tenta di connettersi online, e può garantire comunicazioni sicure e crittografate tra dispositivi, servizi e utenti. L'identità (chiave privata del dispositivo, certificato del dispositivo) è l'elemento critico per garantire l'implementazione di una PKI (Public Key Infrastructure).
- **Provisioning** (Approvvigionamento): è necessario che il prodotto IoT abbia un certificato digitale firmato da qualcuno di fidato.
- **Mastering:** l'immagine software finale viene rilasciata, quindi impacchettata (firmata e crittografata), pronta per essere inviata a una struttura di programmazione sicura. L'output di questo processo è un pacchetto masterizzato che comprende un'immagine software crittografata/firmata e un pacchetto di protezione dell'aggiornamento separato firmato dalla chiave privata del produttore originale (OEM). L'output del processo di mastering è una coppia di file crittografati che vengono caricati in un processo di produzione sicuro.
- **Deployment** (Distribuzione): questo è il processo mediante il quale il pacchetto software masterizzato viene consegnato al prodotto IoT. Nel caso di un aggiornamento sicuro, il pacchetto software masterizzato viene consegnato direttamente al dispositivo IoT che si trova sul campo (ad es. aggiornamento over-the-air o OTA).

4.2 Secure Coding

La maggior parte delle minacce alla sicurezza rilevate sono dovute a vulnerabilità nel codice. Pertanto, la riduzione al minimo dell'utilizzo di comandi non sicuri gioca un ruolo importante nella protezione dei sistemi da potenziali attacchi.

I linguaggi di programmazione C/C++ sono preferiti quando si tratta di scrivere codice per dispositivi IoT, ma le loro vulnerabilità di sicurezza, come la vulnerabilità dei numeri interi, buffer overflow e la vulnerabilità delle stringhe, devono essere affrontate ed evitate dall'inizio durante la scrittura del codice sorgente. Il processo di protezione dei sistemi software da tali problemi di vulnerabilità viene eseguito rimuovendo completamente comandi e funzioni non sicuri noti o sostituendoli con quelli che alcuni chiamano sostituzioni sicure. La convalida verifica se l'input soddisfa una serie di criteri (ad esempio, se il valore rientra in un determinato intervallo). Per sanificazione si intende la modifica dell'input per assicurarne la validità, rimuovendo qualsiasi carattere illegale.

I principali sistemi operativi e compilatori hanno adottato una serie di protezioni di overflow dello stack. Non rendono l'attacco impossibile, ma solo tecnicamente più impegnativo. La prevenzione dell'esecuzione dei dati funziona contrassegnando ogni parte della memoria come eseguibile o non eseguibile.

Con "**Address Space Layout Randomization**" (ASLR), il sistema operativo decide in modo casuale dove caricare il programma da eseguire. L'ASLR è un'ulteriore fonte di incertezza per l'attaccante, ma non rende l'attacco impossibile, piuttosto probabilistico. Gli indirizzi possibili di trasloco sono pochi. L'utente malintenzionato potrà attaccare attraverso brute force.

Nella tecnica dello **stack canary**, il prologo di ogni funzione inserisce un valore canary nello stack tra le variabili locali e l'indirizzo di ritorno. Se un utente malintenzionato vuole sovrascrivere l'indirizzo di ritorno con un buffer overflow, deve corrompere anche questo dato. Ci sarà una funzione (epilogo) che controllerà se il canary (imprevedibile dall'attaccante) ha ancora il valore corretto e se non lo ha genera un errore. Questa è la difesa più efficace contro lo stack overflow, perché sono difficili da aggirare o da indovinare.

La programmazione sicura è un insieme di pratiche e sicure per diminuire le vulnerabilità e gli errori di programmazione. La codifica sicura dipende molto dal linguaggio di programmazione usato. Come regola generale, più basso è il livello del linguaggio di programmazione, più è soggetto a errori e quindi più suscettibile di vulnerabilità. I linguaggi C e C++ sono particolarmente soggetti a errori, perché sono pensati per essere leggeri.

- Un comportamento indefinito (undefined behavior) è un comportamento per il quale lo standard C/C++ non pone requisiti, ad esempio in caso di accesso al buffer fuori limite, dereferenziazione del puntatore nullo o overflow di interi con segno.
- Un comportamento non specificato (unspecified behavior) è un comportamento in base al quale lo standard C/C++ offre due o più possibilità, ad esempio l'ordine di valutazione degli argomenti in una chiamata di funzione.

La maggior parte delle vulnerabilità del software si basa su comportamenti indefiniti. L'attaccante cerca di indurre il programma a eseguire comportamenti indefiniti per mezzo di input speciali predisposti. Quindi, l'attaccante cerca di danneggiare in qualche modo il sistema. La maggior parte delle vulnerabilità in C sono legate alla manipolazione delle stringhe di buffer overflow. Nella maggior parte dei casi, ciò comporterebbe un errore di segmentazione (segmentation fault), ma valori di input dannosi appositamente predisposti, adattati all'architettura e all'ambiente, potrebbero comportare l'esecuzione di codice arbitrario.



5. Exploiting Low Level Communications

5.1 Manomissione di dispositivi IoT

Abbiamo vari livelli, partendo dai nodi, arrivando al network e al cloud. Passeremo su tutti i livelli per osservare tutti i possibili attacchi. Uno dei possibili attacchi ai dispositivi che compongono il livello di percezione di un IoT è la cattura del nodo. Descrive il caso di un avversario che ottiene il pieno controllo dei dispositivi IoT attraverso il diretto accesso fisico. Questo è fondamentalmente diverso dall'ottenere il controllo da remoto tramite un bug del software. Poiché tutti i dispositivi in genere eseguono lo stesso software, se viene rilevato e sfruttato un bug del software, l'avversario è in grado di controllare tutti i dispositivi che eseguono lo stesso software. Un'acquisizione del nodo dovuta alla manomissione durante un accesso fisico può essere condotta contro una piccola porzione della rete di sensori, a differenza di un attacco da remoto che può influenzare tutta la rete. Per cui con un attacco fisico, non attaccherò tutta la rete. L'impatto di questo attacco può essere marginale se vengono rubate informazioni trascurabili, o maggiore se vengono interessati il routing, il provisioning dei dati e/o altre capacità critiche. L'esecuzione di un attacco **node capture** richiede una conoscenza precisa del dispositivo vittima, in modo da trovare le interfacce fisiche disponibili per accedere all'hardware. A meno che le informazioni desiderate non siano accessibili tramite Internet, o le specifiche hardware ottenibili, viene eseguita un'ispezione visiva esterna, osservando come funziona l'I/O del dispositivo, quali interfacce sono esposte all'esterno. Successivamente, viene eseguita un'ispezione interna aprendolo e osservando i suoi componenti interni e ricavando informazioni, ad esempio, dal tipo di processore, o dalla ROM.

Queste azioni consentono all'avversario di escogitare un approccio mirato per manomettere il dispositivo, alterandone il software, conservando i dati e così via. Un aspetto particolarmente importante è rappresentato dalle porte e interfacce di debug. I dispositivi espongono le interfacce di comunicazione che possono essere sfruttate per ottenere l'accesso al dispositivo per eseguire azioni dannose.

5.2 UART, I^2C , SPI

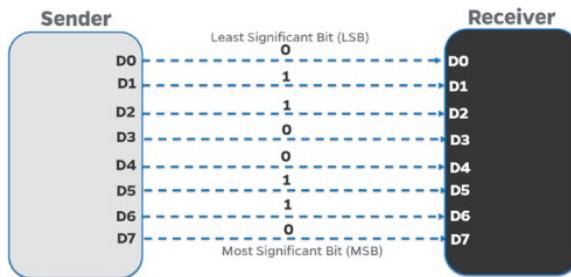
5.2.1 Low-Level Communication

Per qualsiasi dispositivo embedded, i diversi componenti devono interagire tra loro e scambiare dati. La comunicazione seriale e la comunicazione parallela sono i due modi in cui i componenti di un dispositivo si scambiano i dati.

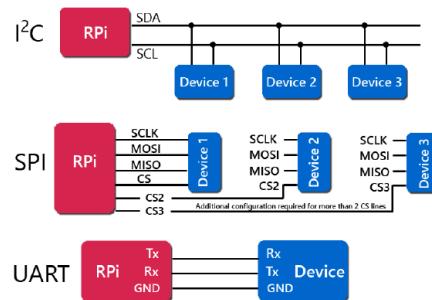
- La comunicazione seriale viene utilizzata per trasferire un bit alla volta attraverso un dato mezzo. Implementata con due shift register, inseriamo i dati negli shift register e un elemento alla volta viene trasferito.



- La comunicazione parallela consiste nel trasferire un blocco di dati contemporaneamente.



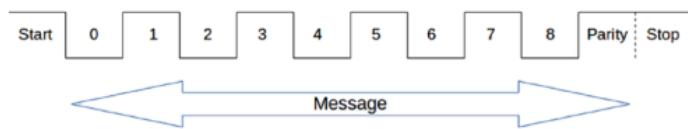
La comunicazione parallela è più veloce, ma richiede anche una maggiore occupazione di spazio sulla scheda. Questo è il motivo per cui la comunicazione seriale è un metodo più comune: richiede solo una singola linea per facilitare lo scambio di dati. Alcuni dei più diffusi canali di comunicazione seriale sono: Universal Serial Bus (USB), High-Definition Multimedia Interface (HDMI), Ethernet, Serial Peripheral Interface (SPI), Inter Integrated Circuit (I^2C), Controller Area Network (CAN), ecc. Visti i recenti progressi tecnologici, la comunicazione seriale sta diventando più economica, più veloce e più affidabile. I^2C , SPI E UART rappresentano protocolli di comunicazione standard disponibili attraverso i pin GPIO (General Purpose Input/Output) delle schede principali. UART è



usato per la comunicazione uno ad uno, per cui non necessita di un clock, il quale è richiesto nel caso in cui avessi una comunicazione uno a molti. I^2C si piazza nel mezzo, è più veloce di UART, qui non abbiamo problemi di sincronizzazione poichè mandaremo il clock. SPI è il più veloce dei tre e abbiamo anche qui il clock.

5.2.2 UART

L' **Universal Asynchronous Receiver/Transmitter** (UART) è un metodo di comunicazione seriale che consente a due diversi componenti di un dispositivo di comunicare tra loro senza la necessità di un orologio. Il termine asincrono significa semplicemente che, a differenza di un protocollo sincrono (es. SPI), non ha un clock che si sincronizza per entrambi i dispositivi tra i quali avviene la comunicazione. Soltanto all'inizio si sincronizzano insieme. La struttura dei pacchetti UART:



Il bit di partenza simboleggia che i dati UART saranno i prossimi. Il messaggio viene convertito e trasferito in 8 bit. Il bit di parità permette di eseguire il controllo degli errori e della corruzione dei dati contando il numero di valori alti o bassi nel messaggio e, in base al fatto che si tratti di una parità dispari o pari, indicherebbe che i dati non sono corretti. Il bit di stop è l'ultimo e indica che la trasmissione del messaggio ha avuto fine. Una porta UART potrebbe essere basata sia su hardware che su software. Gli UART basati su software sono necessari quando bisogna connettere più dispositivi tramite UART a un determinato dispositivo che ha solo set limitati di pin UART hardware. La velocità con cui i dati vengono trasferiti tra i dispositivi è nota come baud rate. È necessario saperlo perché non esiste una linea di clock nel caso della comunicazione UART, quindi entrambi gli endpoint di comunicazione devono concordare su un'unica velocità di trasmissione durante l'intero processo di scambio dei dati UART. Uno dei passaggi iniziali sarà sempre quello di identificare il baud rate del dispositivo di destinazione. Le velocità comuni sono 9600, 38400, 19200, 57600 e 115200 bit al secondo.

5.2.3 I²C and SPI

- L' Inter-Integrated Circuit (I^2C) è un protocollo multi-master con solo due fili necessari per consentire lo scambio di dati: dati seriali (SDA) e orologio seriale (SCL). Tuttavia, I^2C è solo half-duplex, poiché può solo inviare o ricevere dati in un determinato momento. I^2C , a differenza di UART, è sincrono, ed è utilizzato per le distanze brevi. UART riguarda long range distance. La sfida con UART è la limitazione nel facilitare la comunicazione tra solo due dispositivi in un dato momento. Una struttura di pacchetti UART include un bit di avvio e di arresto, che si aggiunge alla dimensione complessiva del pacchetto di dati che viene trasferito, influenzando anche la velocità dell'intero processo. L'UART era originariamente destinato a fornire comunicazioni su grandi distanze, interagendo con dispositivi esterni tramite cavi. Al contrario, I²C è pensato per comunicare con altre periferiche situate sulla stessa scheda.
- La Serial Peripheral Interface (SPI) è un altro protocollo bus sincrono per le comunicazioni tra diversi componenti in un circuito di dispositivi embedded. È full-duplex ed è composto da tre fili - SCK, MOSI e MISO - e un ulteriore chip select/slave select. Un solo master controlla tutti gli slave e il master controlla l'orologio per tutti gli slave. SPI ha velocità di trasmissione dati più elevate rispetto a I^2C , ma l'unico grande svantaggio di SPI è la necessità di più pin, che aumenta il fabbisogno complessivo di spazio. Sia SPI che I^2C sono protocolli comuni quando si parla di archiviazione dei dati tramite la memoria di sola lettura programmabile cancellabile elettricamente (EEPROM).

5.2.4 Low Level Communication Exploitation

Un exploit rappresenta un attacco non etico o illegale che sfrutta le vulnerabilità presenti in applicazioni, reti o hardware. L'attacco è formato in genere da un software o un codice, con l'obiettivo di acquisire il controllo di un sistema informatico o di rubare i dati memorizzati su una rete.

- Per eseguire un **UART-based exploitation**, è necessario un dispositivo in grado di emulare una connessione seriale per accedere al dispositivo di destinazione, come Attify Badge (ma è possibile utilizzare anche un normale USB-TTL o BusPirate). Attify Badge è uno strumento multiuso che ti aiuta a comunicare con altri dispositivi IoT/embedded su varie interfacce di comunicazione, come UART, SPI, I2C e persino standard simili come JTAG. Utilizza un chip FTDI che gli consente di convertire il protocollo di comunicazione hardware in un linguaggio che è compreso da un programma in esecuzione su un computer tradizionale. Per effettuare le connessioni, è necessario identificare dove si trova la porta UART sul dispositivo o quali sono i pin UART ed identificare la velocità di trasmissione corretta.
- **Exploiting I²C** significa leggere o scrivere i dati del dispositivo utilizzando una EEPROM I²C in un dispositivo reale. Per questo, è possibile utilizzare qualsiasi dispositivo che disponga di un chip flash funzionante sul protocollo di comunicazione I²C. La EEPROM deve essere collegata all'Attify Badge. Una volta completata l'operazione di lettura, la connessione I2C viene chiusa e i dati salvati vengono scritti in un file.
- **SPI simile a I2C**

5.3 JTAG and its Exploitation

Il Joint Test Action Group (JTAG) è un'associazione creata a metà degli anni '80 quando un gruppo di aziende si è riunito per risolvere il problema del debugging e del test dei chip affrontando la crescente complessità dei dispositivi. Lo sforzo manuale necessario per testare centinaia di chip con più pin in ciascuno di essi e verificare se ciascuno di essi funziona bene e comunica correttamente ha iniziato a essere pesante poiché la complessità e la produzione su vasta scala di chip stavano crescendo. Per superare questo problema, i produttori hanno escogitato uno standard denominato IEEE 1149.1 per incorporare un componente hardware nel chip stesso per consentire test più semplici. JTAG è solo un modo per testare diversi chip presenti sul dispositivo e eseguirne il debug utilizzando una tecnica nota come boundary scan. Questo viene fatto aggiungendo un pezzo di componente chiamato boundary scan cells vicino a ciascun pin del chip da testare. I vari pin I/O del dispositivo sono collegati in serie per formare una catena. Abbiamo un JTAG input che ci permette di accedere alla catena, e alla fine un JTAG output. Noi mandiamo un dato sulla catena, questo passerà su tutti i pin e alla fine leggeremo l'output. L'input e l'output dovranno essere uguali.

TAP è un nome collettivo dato alle interfacce JTAG presenti su un dispositivo. Ci sono cinque segnali utilizzati da TAP:

- **Test clock (TCK)**: per sincronizzare le operazioni interne e per sincronizzare i dati seriali nelle varie catene (boundary cells)
- **Test data-in (TDI)**: il pin dei dati seriale di input sulle celle di scansione
- **Test data-out (TDO)**: invia i dati dalle celle di scansione.
- **Test mode select (TMS)**: per controllare lo stato del controller.
- **Test reset (TRST, optional)**: Quando il pin reset è premuto, ripristinerà la macchina a stati interna.

I pin TCK, TMS e TRST guidano il controller TAP che gestisce lo scambio generale di dati e istruzioni. Il controller TAP è **una macchina a stati finiti** a 16 stadi che procede da uno stato all'altro, in base ai segnali TMS e TCK. Controlla il test data register che manda informazioni lungo la catena e l'instruction register che manda i segnali di controllo. Se deve essere inviata un'istruzione, l'orologio (TCK) viene attivato e il reset viene impostato su active low per il ciclo di clock. Una volta fatto ciò, il segnale di reset viene quindi disattivato e il TMS viene modificato per attraversare la macchina a stati per successive ulteriori operazioni.

Questi test possono essere utilizzati per trovare problemi che vanno da un semplice difetto di fabbricazione, a componenti mancanti in una scheda, a pin scollegati o posizionamento errato del dispositivo e persino condizioni di guasto del dispositivo. JTAG viene utilizzato per eseguire diverse attività di test e debug. Poiché JTAG è disponibile nei sistemi fin dall'inizio, non appena il sistema si avvia, è estremamente utile per tester e ingegneri esaminare tutti i vari componenti presenti nel dispositivo embedded. Per i penetration tester e i ricercatori sulla sicurezza, è estremamente utile in quanto ci consente di eseguire il debug del sistema di destinazione e dei suoi singoli componenti. Se la scheda di destinazione ha l'accesso JTAG disponibile e contiene un chip flash integrato, è possibile scaricare il contenuto dal chip flash tramite JTAG.

5.3.1 JTAG Exploitation

Identificare i pin JTAG può essere un po' più complicato rispetto a quanto visto in precedenza, dove tutto ciò che serve è cercare un set di tre o quattro pin e quindi utilizzare il multimetro per identificare i singoli pin. Nel caso di JTAG, dovremo utilizzare strumenti aggiuntivi (es.JTAGulator) per determinare efficacemente i singoli pinout presenti nel nostro dispositivo di destinazione. Un'altra cosa da notare mentre si lavora con JTAG è che nella maggior parte dei dispositivi troverai i pad JTAG, invece di pin JTAG o pad con fori, il che rende anche importante per noi avere un po' di esperienza di saldatura. Possiamo identificare pinout JTAG utilizzando due approcci:

- **JTAGulator:** hardware open source, che ci aiuta a identificare i pin JTAG per un determinato dispositivo di destinazione. Dispone di 24 canali I/O che possono essere utilizzati per il rilevamento dei pin e possono essere utilizzati anche per rilevare i pin UART. Utilizza un chip FT232RL che gli consente di gestire l'intero protocollo USB su un singolo chip e ci consente di collegare semplicemente il dispositivo e farlo apparire come una porta seriale virtuale con la quale possiamo quindi interagire utilizzando uno schermo.
- **JTAGEnum:** l'utilizzo di Arduino ha eseguito il flashing con JTAGEnum, un'alternativa molto più economica rispetto a JTAGulator, ma ha qualche limitazione, ad esempio che la scansione è estremamente lenta e non ha la capacità di rilevare pinout UART come fa JTAGulator. Bisogna interfacciarsi con Arduino tramite una connessione seriale tramite il monitor seriale presente nell'IDE di Arduino e si avvia la scansione, JTAGulator alla fine dirà i pin JTAG dei vari fili collegati ad Arduino. La connessione all'interfaccia JTAG consentirà il debug del dispositivo di destinazione e dei programmi in esecuzione su di esso. OpenOCD aiuta in tale attività di debug. Si tratta infatti di un software open source che si interfaccia con la porta JTAG di un debugger hardware per svolgere le seguenti azioni: debug dei vari chip presenti sul dispositivo; impostare breakpoint e analizzare registri/stack in un dato momento; analizzare i flash che si trovano sul dispositivo; programmare e interagire con i flash; scaricare firmware e altre informazioni sensibili.

Sul lato hardware, il debug e lo sfruttamento di JTAG possono essere eseguiti con Attify Badge.

5.3.2 JTAG Protection

Un modo per proteggere un chip dallo sfruttamento JTAG è disabilitare completamente l'accesso JTAG. Ciò si ottiene fondendo il segnale TMS. Le funzioni JTAG sono disabilitate e non è possibile riattivarle. Una soluzione migliore consiste nel fare in modo che una parte di produzione sia naturalmente in uno stato "bloccato" e relativamente immune da attacchi, ma avere una parte della strumentazione sbloccata in situazioni in cui è fondamentale far eseguire un motore di debug o di test prezioso per l'analisi delle cause di un eventuale errore.

5.4 Protecting Low Level Communications

Un Security Controller (SC) viene utilizzato per fornire diversi protocolli di comunicazione (ad es. UART, I2C, SPI,...) con prestazioni di comunicazione stabili ma veloci e permette di avere un livello di sicurezza mediante un certificato di sicurezza combinato con elevate prestazioni crittografiche. Un SC è un modulo hardware discreto, che può essere programmato per fornire un insieme definito di funzioni relative alla sicurezza che operano sulle credenziali crittografiche memorizzate in memoria o generate nell'hardware. Un SC fornisce meccanismi di protezione contro attacchi locali e fisici, per offrire questo livello di resistenza alle manomissioni, gli SC possono essere impiegati all'interno di un nuovo circuito integrato sicuro e protetto con concetti di doppia CPU, bus di comunicazione criptati e coprocessori.

Instruction-Set Randomization (ISR) è un approccio generale per proteggere i sistemi da qualsiasi tipo di code-injection attack alterando casualmente le istruzioni utilizzate da una macchina host, un'applicazione o un'esecuzione. Ad esempio, l'opcode 0xa può denotare l'istruzione XOR su un'applicazione, ma potrebbe essere non valida su un'altra applicazione. Ciò impedisce a un utente malintenzionato di utilizzare lo stesso exploit su più bersagli. ISR implementa un random ISA usando la cifratura. In memoria di solito le istruzioni sono cifrate e vengono decifrate prima dell'esecuzione.

7FC0h:	F4	10	8A	98	9F	FC	CB	7F	72	76	93	D6	BE	3D	A2	C4	ö.Š~ÜÜ.rv"Ö%=&Ä
7FD0h:	FB	A6	FC	07	3B	72	E4	DA	3D	36	B0	44	D6	4B	B8	63	û;ü.;räÜ=6°DÖK,c
7FE0h:	70	6B	63	50	BE	3E	A8	56	14	24	31	28	D9	47	CF	3F	pkcP>"V.\$1(ÙGÍ?
7FF0h:	AF	81	E4	F4	65	34	61	65	3D	4F	67	8C	B9	DC	C9	34	—.äöe4ae=OgŒ¹ÜÉ4
8000h:	02	C5	76	18	73	75	6C	50	34	31	30	32	34	30	31	30	.Åv.sulP41024010
8010h:	A3	A9	04	F1	6C	3F	EE	ED	4F	7D	0B	51	51	AD	2F	F8	£@.ñ1?íiO}.QQ-/ø
8020h:	30	33	30	4A	6F	72	65	5A	73	75	6C	50	01	DC	84	C9	030JoreZsulP.Ü,,É
8030h:	34	3C	31	31	64	72	47	28	70	14	20	24	45	85	81	56	4<11ldrG(p. \$E...V
8040h:	30	49	1E	3A	2F	2C	15	51	59	61	7D	6F	6B	75	2D	50	0I.:/, .QY{oku-P
8050h:	75	31	30	32	34	30	2F	FE	F5	EE	D1	BC	70	54	20	61	u10240/põiÑ4pt a
8060h:	68	7C	61	A5	30	4B	50	59	6B	BA	F9	64	3D	52	53	5A	h a¥OKPYk°ùd=RSZ
8070h:	3D	D3	93	F2	60	D1	08	8F	F5	2F	31	30	64	6F	44	22	=Ó"ò`Ñ..õ/10doD"
8080h:	D0	52	1E	33	CA	67	1C	8D	30	4C	50	5D	36	3B	30	8C	ĐR.3Êg..OLP]6;0€
8090h:	6F	30	65	59	18	F6	C6	D3	02	11	20	32	9E	D2	B9	8D	o0eY.öæÓ.. 2žò¹.

6. Firmware Hacking and Emulation

6.1 Firmware Hacking and Emulation

6.1.1 Firmware Hacking

Il firmware è il componente più critico di un dispositivo IoT, poiché consente la ricerca e lo sfruttamento senza avere alcun accesso fisico diretto al dispositivo. Si tratta di un dato binario, che risiede nella sezione non volatile del dispositivo, che consente e abilita il dispositivo a svolgere diverse attività richieste per il suo funzionamento, nonché a gestire l'operatività di vari componenti hardware per il dispositivo IoT, è come un sistema operativo. Il firmware è diviso in 3 parti:

- Un **bootloader** inizializza la RAM per l'archiviazione di dati volatili, inizializza le porte seriali, rileva il tipo di macchina, imposta l'elenco dei tag del kernel, carica il filesystem RAM iniziale e chiama l'immagine del kernel. Il bootloader inizializza i driver hardware tramite un Board Support Package (BSP), sviluppato da una terza parte.
- Il **kernel** è uno dei componenti principali dell'intero dispositivo embedded, implementa le funzionalità del dispositivo IoT, e funge da strato intermedio tra l'hardware e il software. In Arduino, quando chiamiamo SPI, ci sono librerie che rappresentano il nucleo del firmware dell'arduino.
- Esistono molti tipi di file system utilizzati all'interno del firmware e talvolta vengono utilizzati anche tipi di file proprietari a seconda del dispositivo.

Oltre ai diversi tipi di file system, esistono anche diversi tipi di compressione in uso per risparmiare spazio di archiviazione sul dispositivo. Alcune compressioni comuni che vediamo nei dispositivi IoT sono LZMA, Gzip, Zip, Zlib, ARJ. A seconda del tipo di file system e del tipo di compressione utilizzato da un dispositivo, il set di strumenti che utilizzeremo per estrarre informazioni dal firmware sarà diverso. Ci sono vari modi per estrarre il file system da un dispositivo:

- **Scaricarlo** da internet: molti produttori decidono di mettere online il loro pacchetto binario del firmware.
- È possibile **estrarre il firmware** direttamente dalla vittima una volta che l'attaccante ha accesso fisico al dispositivo. Utilizzando varie tecniche di exploitation dell'hardware, il firmware può essere scaricato dal chip flash del dispositivo.

- La tecnica dello **sniffing** (intercettazione di dati) consente di ottenere il pacchetto binario del firmware durante l'esecuzione di un aggiornamento.
- Attraverso **reverse engineering** effettuando l'inversione delle applicazioni che può essere eseguita esaminando le applicazioni del dispositivo IoT e da lì trovando un modo per ottenere il firmware.
- Da un'immagine del firmware è possibile estrarre il file system utilizzando un approccio manuale o automatizzato. Consideriamo una data immagine del firmware, che generalmente ha .bin come estensione, è possibile usare determinati comandi come "file", "hexdump", "grep", per capire la posizione di dove inizia il file system. Tale offset può essere utilizzato per scaricare selettivamente il file system dal file binario utilizzando un comando chiamato "dd". Tramite unsquashfs è possibile rivelarne il contenuto.

Le informazioni più importanti da estrarre all'interno del file system sono i valori sensibili, come credenziali hardcoded, accesso backdoor, URL sensibili, token di accesso, API e chiavi di crittografia, algoritmi di crittografia, percorsi locali, dettagli dell'ambiente, meccanismi di autenticazione e autorizzazione. In quasi tutti i casi, il firmware non è compresso, ma crittografato poichè io devo proteggere le informazioni. Posso crittografare con AES oppure con XOR. Binwalk fallisce in questo caso. La semplice esecuzione di un "hexdump" consente di vedere se ci sono stringhe ricorrenti, questo accade quando effettuo lo XOR di una lettera con una chiave. In questo caso, posso decrittare il firmware invertendo lo xor.

6.1.2 Firmware Emulation

Una volta che un firmware con un file system estratto è disponibile, la prima cosa da fare è guardare i singoli binari (file ".bin") e vedere se ci sono delle vulnerabilità. I dispositivi IoT funzionano su architetture diverse e non necessariamente x86, quindi è necessario comprendere e analizzare le differenti piattaforme. Può essere utilizzata un'utilità nota come Qemu per emulare dispositivi embedded e i file binari. Il comando "readelf -h" su qualsiasi binario all'interno del file system DVRF permette di identificare l'architettura. Il passaggio successivo consiste nell'eseguire un binario emulando l'architettura. Un binario del firmware che originariamente doveva essere eseguito solo su architetture basate su MIPS (ad esempio) è stato emulato ed eseguito. In questo modo siamo riusciti ad emulare solo una porzione del file system. Con Qemu è possibile emulare l'intera immagine del firmware. Questo è utile poichè da accesso a tutti i singoli binari nel firmware, consente di eseguire attacchi di rete al firmware ecc ecc. Fare ciò non è semplice come emulare una singola porzione poichè il firmware durante l'avvio potrebbe richiedere configurazioni e informazioni aggiuntive dalla RAM non volatile (NVRAM) e l'esecuzione del firmware potrebbe dipendere dai componenti hardware fisici. Questa sfida può essere risolta utilizzando il proxy.

6.2 Backdooring Firmware

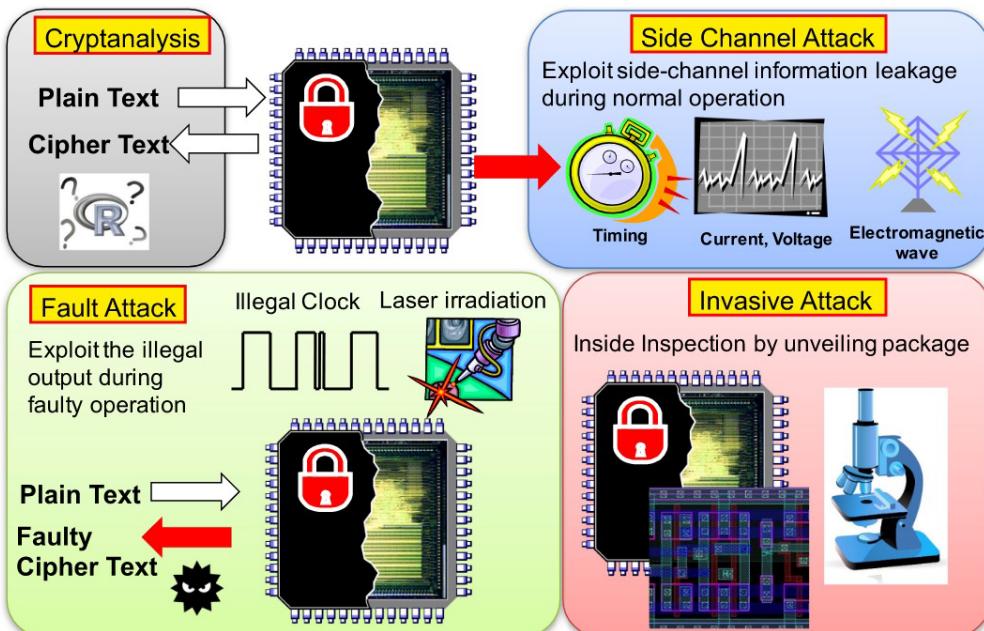
La backdoor è una dei problemi di sicurezza che il firmware deve affrontare se il dispositivo non dispone di controlli di integrità sicuri e di convalida della firma, questo permette di bypassare i controlli ed accedere alle funzionalità del firmware senza avere le autorizzazioni per farlo. In quanto attaccanti, il file system viene estratto dal firmware e quindi quest'ultimo viene modificato aggiungendo la nostra backdoor. Qui abbiamo 3 problemi: dobbiamo analizzare il firmware, scrivere il codice e compilarlo effettuando cross-compilation (ad esempio abbiamo linux e dobbiamo compilare per un MIPS). Per estrarre il file system dal firmware, invece di utilizzare Binwalk, è possibile utilizzare uno strumento chiamato Firmware Mod Kit.

Top 10 IoT Security Vulnerabilities



7.1 Physical Attacks

La Cryptanalysis consiste nel dare il plain text al device e ricevere il cipher text. Effettuiamo il mapping tra plain e cipher in modo da scoprire quali primitive crittografiche sono usate. Lo scopo è rubare informazioni importanti a livello crittografico.



7.1.1 Attacchi non invasivi

Non penetrano il dispositivo attaccato, quindi non lasciano segni di manomissione o prova. Possono essere effettuati attraverso strumenti come multmetro, oscilloscopio, PC con scheda di acquisizione dati, ovvero tools che analizzano dati e il comportamento dei dispositivi (il tutto a livello software). Abbiamo attacchi passivi e attivi. Nei primi osserviamo che dati vengono prodotti, negli attivi forziamo dei glitch per comprendere il comportamento e come passare i meccanismi di sicurezza. Forza bruta, fault injection, side-channel attacks sono tipi di attacchi non invasivi.

Side Channel Attack

Un attacco side-channel è qualsiasi attacco che si basa sull'ottenere informazioni monitorando svariati elementi. Di solito non tocco questi elementi, ma li monitoro soltanto. I tools utilizzati sono multmetro, oscilloscopio, schede di acquisizione dati ecc.

Abbiamo diversi tipi di attacchi di questo tipo:

- **Timing attacks** mirati a diversi tempi di calcolo: Ogni operazione ha un tempo di esecuzione specifico all'interno di una data architettura, e misurando il tempo è possibile determinare che tipo di operazione è stata eseguita e quante volte. Durante l'attacco, l'avversario mantiene l'input, l'output e il tempo consumato e controlla la correlazione tra le misurazioni del tempo della chiave indovinata o dell'input e il risultato empirico (spesso statisticamente) in modo da ricostruire un programma, una primitiva. L'attacco a tempo viene spesso utilizzato per compromettere sistemi crittografici a chiave pubblica come RSA.
- **Power analysis**: misurazione del consumo di energia nel tempo in modo da comprendere che tipo di operazione si sta effettuando. Richiede un set di apparecchiature molto semplice, un PC con un oscilloscopio e un piccolo resistore nella linea di alimentazione. Questo è molto efficace contro molti algoritmi crittografici e schemi di verifica delle password. Abbiamo SPA e DPA. Gli attacchi SPA (Simple Power Analysis) si basano sull'interpretazione delle tracce di potenza per rivelare, ad esempio, la sequenza delle istruzioni eseguite. Il consumo energetico di un circuito CMOS è costituito da due componenti:
 - potenza statica: dovuta alla corrente di dispersione dei transistor e dipende dal progetto del circuito.
 - potenza dinamica: dovuta alla commutazione dei transistor e dipende dai dati elaborati e dall'operazione eseguita.

La SPA può essere utilizzata per attaccare l'algoritmo RSA: l'obiettivo dell'aggressore è quello di estrarre la chiave privata d , che viene utilizzata durante la decrittazione.

Poiché l'analisi della potenza utilizza la relazione tra il consumo di energia e i dati elaborati, la potenza dinamica è quella rilevante. Poiché la potenza statica è per lo più costante, la variazione della potenza totale è dovuta esclusivamente alla potenza dinamica. La potenza dinamica viene consumata quando si verifica la commutazione nei transistor. Cioè se una cella CMOS cambia da 0 a 1 o da 1 a 0, la commutazione avviene nei transistor e si consuma energia. Un modello di potenza viene utilizzato per dedurre matematicamente il consumo di energia di un circuito utilizzando le informazioni che abbiamo sul circuito. Il modello della distanza di Hamming è molto semplice e si può dedurre dal monitoraggio del consumo di energia. Se il valore iniziale su un componente hardware, come il bus, era v_0 e il valore dopo la modifica è v_1 , quindi la distanza di hamming può essere trovata semplicemente contando il numero di transizioni da 0 a 1 e da 1 a 0. Tuttavia, il consumo energetico dipende anche dai dati elaborati. Infatti, gli attacchi di analisi della potenza differenziale (DPA) si basano su

- indagini statistiche di più tracce per dedurre informazioni sui dati elaborati.
- Electro-magnetic analysis (EMA): misurazione dell'emissione simile al precedente, ma al posto del resistore viene utilizzata una bobina magnetica.

Data Remanence Attack

Consiste nella rappresentazione residua dei dati dopo la cancellazione. Dal punto di vista informatico i dati sono cancellati, ma dal punto di vista elettronico, no. La permanenza dei dati a bassa temperatura è pericolosa per i dispositivi a prova di manomissione che memorizzano chiavi e dati segreti in una batteria di backup. Devo quindi preoccuparmi di queste memorie non volatili ampiamente utilizzate nei microcontrollori, come EPROM, EEPROM e Flash che conterranno questi dati, i quali possono essere file del file system, file di backup ecc., da cui un attaccante può estrarre informazioni sensibili per poter sferrare un attacco.

Fault Injection Attack

Si dividono in glitching (il segnale cambia randomicamente molto velocemente, l'elemento non riesce a raggiungere uno stato stabile, ciò permette di corrompere dati e indurre problemi di alimentazione) e bumping (altera determinati valori, ad esempio il segnale del clock in modo che il flip flop non legga ciò che dovrebbe, dando input differenti).

Brute Force

Consiste nel provare tutte le possibili combinazioni di una chiave o di una password o di ascoltare una comunicazione per recuperare informazioni nascoste. I chip moderni scoraggiano la maggior parte degli attacchi di forza bruta, chiavi più lunghe rendono la ricerca impossibile.

7.1.2 Attacchi invasivi

Sono attacchi penetranti che lasciano segni di manomissione. Sono attacchi di questo tipo l'**internal fault injection**, il **chip modification**, **reverse engineering** (comprendere la struttura di un dispositivo a semiconduttore e le sue funzioni), decapsulazione ecc. Questa volta i tools devono poter manipolare l'hardware ed effettuare operazioni, vengono utilizzati saldatrice, microscopio, oscilloscopio, sistemi di taglio laser per rimuovere chip o parti metalliche oppure per intercettare segnali all'interno di un chip e iniettare segnali di test per osservare la reazione dei dispositivi, quindi viene modificato il chip stesso.

7.1.3 Attacchi semi-invasivi

Gli attacchi semi-invasivi sono utili a colmare il divario tra attacchi non invasivi e invasivi meno dannosi, ad esempio effettuando decapsulazione senza penetrazione, quindi minori danni al dispositivo. Sono attacchi più facili da configurare e ripetere rispetto agli attacchi invasivi. Questo viene effettuato attraverso l'utilizzo di tecniche ad infrarossi, scansioni laser ad infrarossi, microscopi speciali e fault injection.

7.2 Protecting IoT Devices from Physical Attacks

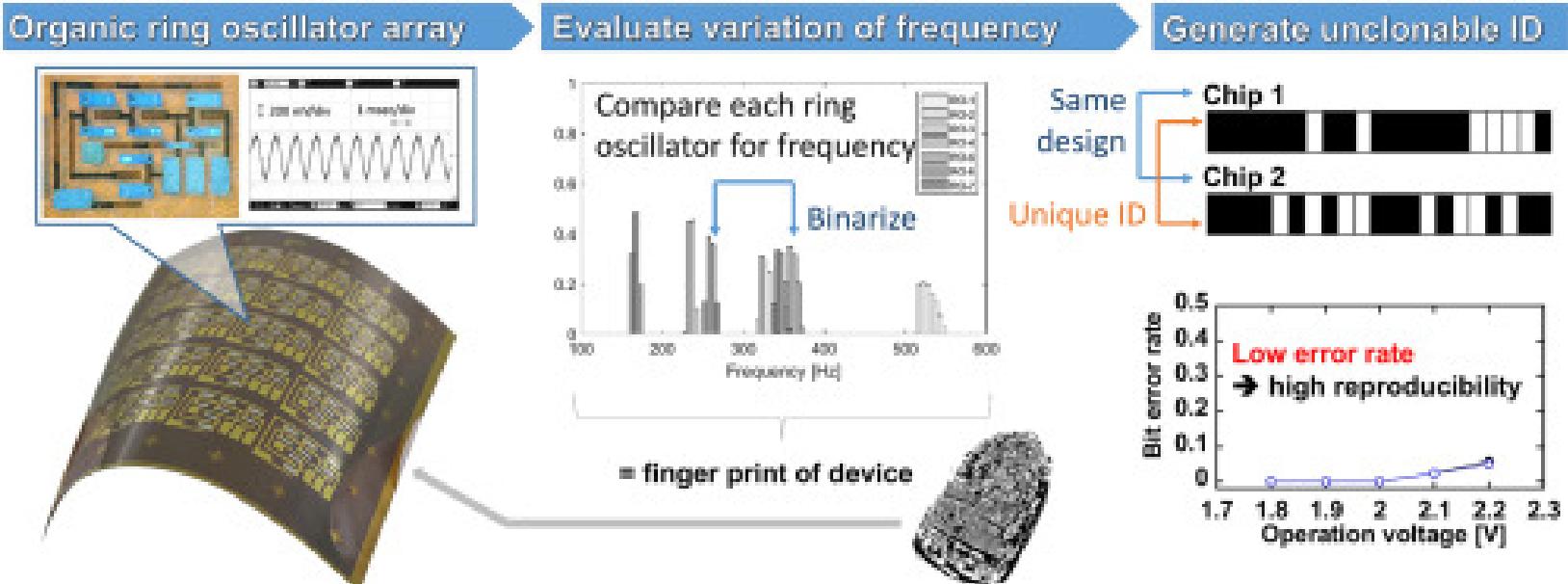
7.2.1 Tamper Resistance (Resistenza alla manomissione)

La resistenza alla manomissione consiste principalmente nell'imballaggio (costruzione) di un dispositivo progettato per renderne difficile la manomissione. Abbiamo varie tecniche per ottenere ciò, come ad esempio involucri d'acciaio, viti, serrature. La maggior parte di queste soluzioni sono a

prova di manomissione, il che significa che i cambiamenti fisici possono essere osservati visivamente, in modo da rendere evidente che il prodotto è stato manomesso. È fondamentale eseguire una panoramica del design per eventuali falle di sicurezza e testare i prodotti contro gli attacchi noti.

7.2.2 Tamper Protection (Protezione antimanomissione)

La protezione consiste nel design del chip, nel diminuire le dimensioni dei transistor per rendere gli attacchi meno fattibili, nell'inserire più strati di metallo per bloccare qualsiasi accesso diretto.



8. Physically Unclonable Functions

8.1 Introduction to PUFs

I PUF possono essere paragonati alle tecniche di biometria per autenticare un soggetto, solo che in questo caso vogliamo autenticare l'hardware. In generale i PUF sono operazioni probabilistiche che dato un input, o una challenge, danno un output o una risposta.

Le challenge-response sono chiamate CRP, e la risposta di questa CRP sarà una variabile random e tramite la distribuzione di probabilità del PUF sarà possibile risalire al dispositivo.

I PUF sono utilizzati in due fasi:

- Di inizializzazione: si effettuano un numero di CRP su un PUF e si conservano i risultati in un database;
- Di verifica: si effettua randomicamente una CRP su un PUF e si compara la risposta con quella presente nel database.

La distribuzione delle risposte può essere analizzata in due modi:

- Intra-distance: è una variabile random che descrive la distanza tra due risposte della stessa challenge sullo stesso PUF;
- Inter-distance: è una variabile random che descrive la distanza tra due risposte della stessa challenge su diversi PUF.

Si usa la distanza di Hamming se le risposte sono in bit. La intra-distance dovrebbe essere molto piccola rispetto l'inter-distance che dovrebbe essere più grande (Hamming distance 50/La sicurezza dei PUF si classifica in:

- PUF Strong: dato un avversario che ha accesso al PUF per un tempo abbastanza lungo, ci sarà con molta probabilità una challenge a cui non saprà la risposta;
- PUF Weak: tutto ciò che non è un PUF strong.

Un esempio estremo di PUF è chiamato POK (Physically Obfuscated Key) è un PUF con una singola challenge che da in output una singola risposta e la manda direttamente al processore, senza conservarla in memoria.

Tuttavia, costruire praticamente (intrinseco) un PUF forte e sicuro è molto difficile.

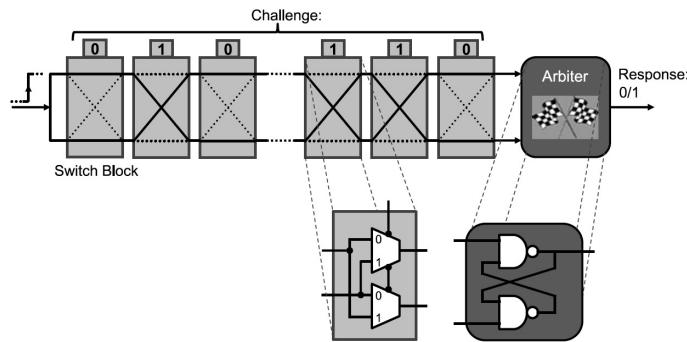
8.2 Intrinsic PUF

Tutte le costruzioni di PUF intrinseci sono "silicon based", ovvero si basano su variazioni di processo casuali che si verificano durante le fasi di fabbricazione dei chip di silicio.

- Data-Delay Based PUFs: misurano le variazioni casuali sul ritardo del circuito;
- Memory Based PUFs: Utilizzano variazioni casuali tra dispositivi accoppiati chiamati anche mismatch, in elementi di memoria.
- Intrinsic Silicon PUFs: Cercano di quantizzare una conversione analogico digitale, producendo una risposta digitale che ha una certa randomness.

8.2.1 Arbiter PUF

Nello specifico, un tipo di Data-Delay Based PUFs è denominato **Arbiter PUF**, la cui idea è quella di introdurre una race condition tra due digital paths su un chip.

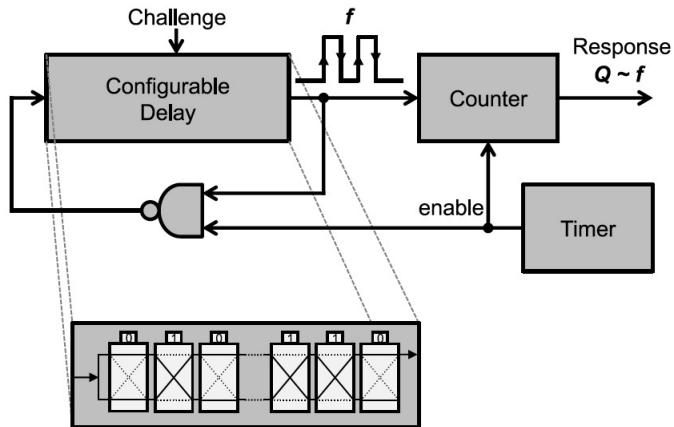


(In basso a destra è un Flip-Flop, mentre ogni Switch Block è un multiplexer 2 a 1).

Un multiplexer si fa con 2 AND (di cui uno negato) e 1 OR. I Flip-Flop si dividono in: RS, D, T, JK. Nel nostro caso ha due input, quindi RS. La randomness in questo tipo di PUF deriva da quale dei due path arriva prima (**delay**), infatti se applichiamo la stessa challenge il risultato sarà diverso. La risposta dipende da quale porta AND sarà attivata per prima.

8.2.2 Ring Oscillator PUF

Un altro tipo di Data-Delay Based PUFs è il **Ring Oscillator PUF**. E' una variante del precedente il quale viene trasformato in un Oscillatore aggiungendo una fase di feedback. Il valore delle oscillazioni vengono inviate a un counter il quale conta il numero di oscillazioni in un intervallo di tempo. Di

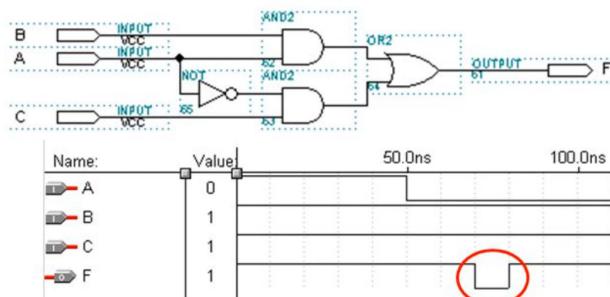


solito questa costruzione dipende dalla fase di fabbricazione del chip o anche dalle influenze esterne come, ad esempio, il cambio della temperatura che può infierire nel voltaggio del circuito. Ad esempio con l'aumento della temperatura si può forzare un PUF verso una soluzione precisa. Una possibile soluzione è avere multipli oscillatori e multipli counter.

8.2.3 Glitch PUF

Un circuito puramente combinatorio non ha uno stato interno: la sua uscita in stato stazionario è interamente determinata dai suoi segnali di ingresso. Tuttavia, quando il valore logico dell'ingresso cambia, possono verificarsi effetti transitori, cioè può passare del tempo prima che l'uscita assuma il suo valore di stato stazionario. Questi effetti sono chiamati glitch e il verificarsi di glitch è determinato dalle differenze di ritardo dei diversi percorsi logici dagli ingressi a un segnale di uscita. La randomicità è influenzata, anche in questo caso, da come è costruito il PUF e dai valori che entrano in input nel circuito combinatorio, faceendo azionare le porte in modo casuale.

- $F = AB + A'C$: assume all gate delays = 10ns

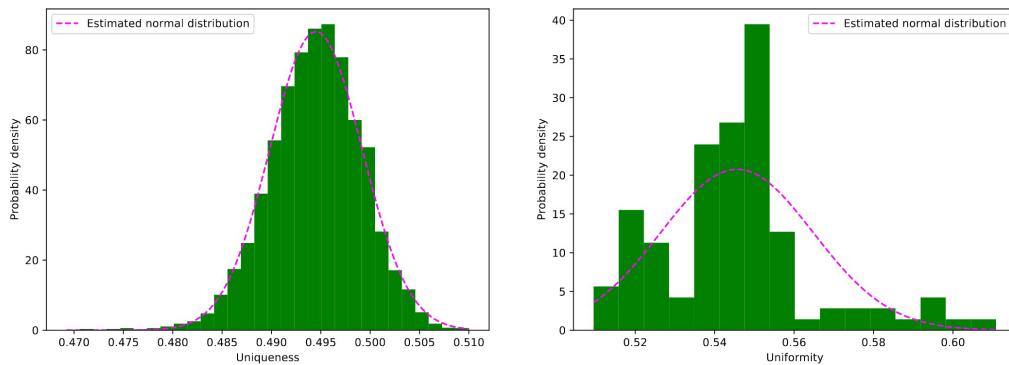


8.2.4 SRAM PUF

Fino ad ora non siamo riusciti a costruire un vero PUF poichè abbiamo ricorso a manipolazione dei dati sui chip. Ad esempio su Arduino possiamo utilizzare un vero PUF, un SRAM PUF, che dipende dalla randomness della memoria, quando quest'ultima viene accesa. L'SRAM (Static Random-Access Memory), in una tipica CMOS implementation, è costruita con sei transistors.

SRAM PUF with Arduino

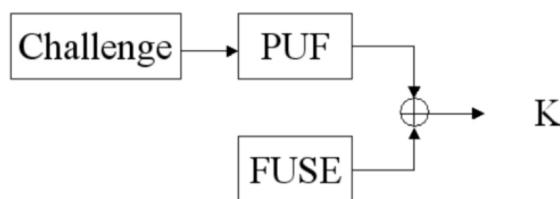
E' possibile leggere il contenuto della SRAM in Arduino in modo da costruire un PUF, solo che c'è una piccola limitazione, ovvero quando accendiamo Arduino, il bootloader inizializza il contenuto della memoria quindi distrugge il contenuto random precedente e lo inizializza con tutti 0. Quindi servono due Arduino connessi tra di loro, tra cui uno carica nell'altro un bootloader. Così facendo possiamo ripetutamente accendere la scheda usando un secondo Arduino per leggere il contenuto stampato sul serial communication bus.



Il grafico a sinistra riporta l'intra-distance, mentre a destra troviamo l'inter-distance. Possiamo vedere come la seconda è

8.2.5 POK

POK (Physically Obfuscated Key) è un modo per avere una chiave in hardware. Ciò implica la presenza di un PUF e di una challenge con una risposta a questa challenge. Con il POK si vuole una chiave che non dipende dalla variabilità del PUF, perché vogliamo che la chiave sia sempre la stessa. Quindi alla risposta del PUF si fa uno XOR con un'altra cosa così da ottenere sempre la stessa chiave. Questa cosa toglie la randomness della risposta del PUF.



8.2.6 Controlled PUFs

Il CPUF (Controlled PUF) è un modo di usare un PUF insieme a primitive crittografiche. Ad esempio un hash si usa per generare la challenge per il PUF in modo da prevenire attacchi su determinate challenge. Quindi dall'esterno non vediamo più la challenge ma solo l'input e l'output del PUF. Si usano algoritmi di error-masking per ridurre la probabilità di errori nelle risposte del PUF.

8.2.7 Reconfigurable PUFs

Questo tipo di PUF consiste nell'estendere la classica challenge-response di un PUF con una operazione in più, ovvero la riconfigurazione.

8.3 PUFs based identification

Ci sono 2 modi per utilizzare i PUFs:

- Identificare l'hardware
- Per generare chiavi

In genere l'identità di qualcosa può essere **assegnata** (come un username o codice a barre) o può essere **intrinseca**, cioè una caratteristica che deriva dall'soggetto (ad esempio l'impronta digitale è intrinseca).

Con i PUFs silicon based è possibile trovare funzionalità uniche per l'identificazione. Nella prima fase si assegna l'identità che prima di tutto deve essere generata assicurandoci che sia unica. Quando si assegna l'identità vogliamo che essa sia permanente e quindi si richiede che sia flashata sul dispositivo (si effettua in fase di costruzione).

8.4 PUF-Based Key Gen

In PUF possono essere generati per creare chiavi perché sono fonte di randomness. Il vantaggio sta di non salvare chiavi in memoria, dato che è difficile proteggerla, ma di richiederla a un PUF ogni volta che serve a un processore.

4BD03C00	887525C1	01A07700	37D14D00
B7125G0	024FG002	53D03C00	AD722500
BD03C00	887525C1	4F553D	53414242
F4F3D41	4242434E	3D4A6	64692041
96C2F4F	553D4553	414	4F3D414
425604	00312E30	3424	0003424
003042	4C0	024E4E4F	00B1D3
2254F1	21	8833B0CC	2957EE
3ECAA	CB3EE8EF	DF038D7F	A14217
2AA4D	04143B75	4F571C83	535C0
7DED9	B571C83	07	FA49F

9. Lightweight Cryptography

9.1 Introduction to Cryptography

La crittografia riguarda la costruzione e l'analisi di protocolli che impediscono la lettura di messaggi privati e garantiscono vari aspetti della sicurezza delle informazioni come la riservatezza dei dati, l'integrità dei dati, l'autenticazione e il non ripudio. La crittografia è la conversione di informazioni da uno stato leggibile a un'apparente assurdità utilizzando una funzione matematica e una stringa casuale di bit difficile da indovinare. Solo chi è autorizzato potrà convertire il testo da cifrato a decifrato.

Nella crittografia classica, il testo cifrato costituisce una permutazione del testo in chiaro, cambio l'ordine delle lettere, cambio l'alfabeto. Per la cifratura a blocchi, per appunto cifriamo di blocco in blocco e non di bit in bit. Per la crittografia asimmetrica, in ciascuna operazione vengono utilizzate due chiavi e sono diverse, ma matematicamente correlate, in modo tale che sia possibile recuperare il testo in chiaro decrittando il testo cifrato. Lo standard utilizzato è **AES**, si basa su un principio di progettazione noto come rete di sostituzione-permutazione, con una dimensione del blocco fissa di **128 bit** e una dimensione della chiave di 128, 192 o 256 bit. Si hanno 3 tipi di chiave per problemi di efficienza, tempi di esecuzione, uso della memoria. Ognuno dei 3 avrà un numero di round totale differente. Rivest Cipher 4 (RC4) è il più utilizzato tra gli stream cipher.

L'**hashing** è il processo di conversione di un pezzo di dati di dimensioni arbitrarie in un altro valore di dimensione fissa utilizzando un algoritmo appropriato, che dovrebbe essere unidirezionale in modo che l'hash non possa essere riconvertito nell'input originale. L'output cambierà di molto anche in caso di un semplice cambio di bit nell'input. Secure Hash Algorithms (SHA) sono una famiglia di funzioni hash crittografiche. Anche AES e SHA funzionano bene insieme all'interno dei sistemi informatici, hanno difficoltà nel mondo dell'Internet delle cose perché occupano troppa potenza di elaborazione, troppo spazio fisico, troppa carica della batteria consumata. Situazione simile al teorema CAP, devo fare un trade-off. Di norma, due qualsiasi dei tre obiettivi di progettazione possono essere raggiunti facilmente.



In sostanza, abbiamo due tipi di crittografia, Conventional cryptography e Lightweight cryptography. La seconda prende gli algoritmi della prima cercando di ridurre i costi modificando il design **introducendo nuovi protocolli per avere maggiore efficienza**.

9.2 Introduction to Lightweight Cryptography

Nella crittografia leggera, si vedono spesso blocchi di dimensioni **più piccole** (in genere 64 bit o 80 bit), chiavi più piccole (spesso meno di 90 bit) e round meno complessi (e dove le S-box spesso hanno solo 4 bit per via della dimensione della tavola in modo da utilizzare meno memoria). E' stato identificato come i metodi tradizionali abbiano punti deboli contro i side-channel attack. Per la crittografia leggera, i principali vincoli che abbiamo sono in genere correlati ai requisiti di alimentazione (considerando che non è possibile ricaricare o sostituire la batteria), gate equivalents (GE) e ai tempi di esecuzione (che dovranno ovviamente essere bassi). Spesso i metodi di crittografia leggera bilanciano le prestazioni (throughput) con il consumo di energia e GE. I metodi di crittografia leggera devono anche avere bassi requisiti di RAM e ROM (dove il metodo è memorizzato sul dispositivo).

Nell'IoT, molti dispositivi interconnessi con risorse limitate non sono progettati per eseguire costosi calcoli crittografici classici, il che rende difficile implementare funzioni crittografiche sufficienti. Diverse soluzioni di crittografia leggera sono state standardizzate come ISO/IEC 29121. Per comunicazioni sicure, le primitive leggere sono state incorporate nei protocolli esistenti, come IPSec e TLS, e sono state rilasciate alcune librerie incorporate, come wolfSSL, sharkSSL, ecc.

Più in generale e indipendentemente dalla piattaforma, un'implementazione di questi algoritmi è un compromesso tra sicurezza, prestazioni e costi. Un chip economico ed efficiente può essere vulnerabile ai side-channel attack, uno economico e resistente a questo attacco può risultare più lento e infine ci si può aspettare che uno veloce e sicuro sia costoso. Il ruolo degli algoritmi leggeri è quello di fornire compromessi, ad esempio riducendo la difficoltà di un'implementazione hardware molto efficiente.

9.3 Implementazione hardware

Se la primitiva è implementata nell'hardware, le seguenti metriche descrivono l'efficienza dell'implementazione:

- Il **consumo di memoria** e la **dimensione dell'implementazione** sono raggruppati nella relativa area di gate, misurata in Gate Equivalents (GE) (questo valore definisce la complessità della tecnologia di produzione, indipendentemente dalla complessità dei circuiti elettronici digitali). Quantifica quanto è grande l'area fisica necessaria per un circuito che implementa la primitiva. Più è basso, meglio è.
- Il **throughput**, misurato in bit o byte al secondo, corrisponde alla quantità di testo in chiaro elaborato per unità di tempo. Più è alto, meglio è.
- La **latenza**, misurata in secondi, corrisponde al tempo impiegato per ottenere l'output del circuito avendo dato l'input. Più è basso, meglio è.
- Il **consumo energetico**, misurato in Watt, quantifica la quantità di energia necessaria per utilizzare il circuito. Più è basso, meglio è.

Questi quattro criteri competono tra loro, ad esempio una bassa latenza tende a implicare una gate area più alta. Il compromesso ottimale tra queste quantità dipende molto dal contesto.

Indipendentemente dalla piattaforma esatta, una data primitiva può essere implementata utilizzando approcci diversi. Consideriamo le permutazioni basate su round. Un approccio diretto consiste nel memorizzare lo stato interno completo (insieme allo stato chiave, se presente) e quindi eseguire un round utilizzando un circuito che opera sullo stato completo in una sola volta. Ad ogni round darò x ed avrò y . Ad ogni round implemenerò la f . Questa è una esecuzione parallela, per cui avrò una bassa latenza.

Le implementazioni seriali funzionano in modo diverso. Nella seriale avrò una singola implementazione della f , guadagno efficienza, e genero solo una parte dell'output, il che consumerà meno memoria, e avrò più latenza essendo più lento come meccanismo. Aggiornano solo una piccola parte dello stato alla volta.

9.3.1 Memory limit

La memoria è spesso la parte più costosa dell'implementazione di una primitiva leggera. Di conseguenza è preferibile operare su blocchi piccoli utilizzando una chiave piccola, questo diminuisce però la sicurezza. Tuttavia, è possibile risparmiare spazio codificando o "masterizzando" le chiavi nel dispositivo. Cioè, invece di utilizzare la memoria di lettura/scrittura per archiviare la chiave, utilizzare strutture di sola lettura che non sono utilizzate dal programma, questo è il concetto del **Secure Element(SE)**.

9.4 Implementazione software

Le primitive leggere possono anche essere implementate nel software, tipicamente per l'uso su microcontrollori, attraverso l'uso di librerie. In questo caso, le metriche rilevanti sono:

- **consumo di RAM**: corrisponde alla quantità di dati che viene scritta in memoria durante ogni round della funzione,
- **dimensione del codice**: la quantità fissa di dati necessaria per valutare la funzione indipendentemente dal suo input, ciò dipende dalle librerie usate,
- **throughput**: misura la quantità media di dati che viene elaborato durante ogni ciclo di clock.

I primi due sono misurati in byte e dovrebbero essere ridotti al minimo mentre il secondo è misurato in byte per ciclo e dovrebbe essere massimizzato. Di nuovo, queste tre quantità non sono indipendenti. Ad esempio, il caricamento delle informazioni dalla RAM nei registri della CPU è un'operazione costosa, così come il suo inverso. Pertanto, limitare il numero di tali operazioni comporta una diminuzione sia del consumo di RAM che di un aumento (secondo me diminuzione) del throughput. I microcontrollori operano su piccole parole di 8, 16 o 32 bit, per cui alcune operazioni possono avere costi controintuitivi. Di conseguenza le rotazioni hanno prezzi molto diversi e non trascurabili. È quindi abbastanza diverso dal caso hardware, dove tutte le permutazioni di bit sono estremamente economiche. Dalla fase di design io dovrò capire se implementare via software o hardware considerando le operazioni che dovrò fare. Proprio come nel caso dell'hardware, ci sono diversi trade-off e tecniche di implementazione. Ad esempio, le sottochiavi di un cifrario a blocchi possono essere precalcolate per risparmiare tempo a scapito dello spazio in memoria durante una crittografia. È anche possibile utilizzare tecniche molto più sofisticate come il bitslicing che, sebbene non siano sempre applicabili, possono portare a sostanziali miglioramenti delle prestazioni. La tecnica del bit-slicing offre un modo per ridurre l'overhead.

9.4.1 Robustezza SCA

I **side-channel attack** utilizzano alcune conoscenze speciali sull’implementazione di un cifrario per violarne la sicurezza. Ad esempio, l’osservazione del consumo energetico di una crittografia può far trapelare informazioni sul peso di Hamming dell’output di una S-Box. Gli algoritmi leggeri sono spesso costruiti in modo da ridurre la vulnerabilità della loro implementazione a tali attacchi. Le SCA (side-channel attack) sfruttano il fatto che le diverse operazioni eseguite durante una crittografia corrispondono a processi fisici che possono far trapelare informazioni su un segreto come la chiave. Di conseguenza, misurare con precisione il consumo energetico di questo dispositivo può rivelare alcune informazioni sull’ingresso della S-Box e, quindi, sullo stato interno del cifrario. Sia l’implementazione hardware che quella software possono essere vulnerabili. Per proteggersi da un simile attacco, tutte le operazioni possono essere mascherate utilizzando una fonte esterna di dati casuali per rendere casuali gli input delle diverse operazioni che potrebbero far trapelare informazioni. Più formalmente, garantisce che l’input di ciascuna operazione sia statisticamente indipendente da dati segreti come lo stato interno di un cifrario a blocchi o la chiave usata (la quale fa parte proprio dello stato, dato che quest’ultimo è composto dalla round key e dall’input i quali producono l’output con una nuova round key). Diverse operazioni possono essere più o meno facili da mascherare. Pertanto, è possibile e in effetti diventa una tendenza nella crittografia leggera semplificare l’implementazione di tali contromisure. Molti implementatori hanno lavorato all’ottimizzazione dell’implementazione dell’AES con un certo successo sia nell’hardware che nel software. Alcuni dispositivi vengono talvolta forniti con un modulo di accelerazione hardware che fornisce la crittografia e la decrittografia AES, aggiungendo di fatto un nuovo set di istruzioni interamente dedicate a una rapida valutazione di questo codice a blocchi. Tuttavia, l’accelerazione hardware della cifratura a blocchi ha i suoi limiti. La crittografia con accelerazione hardware utilizzata da molti dispositivi è vulnerabile a varie forme di attacchi side-channel. Senza supporto hardware, l’AES ottimizzato per il software è piuttosto veloce. Tuttavia, la sua implementazione richiede la memorizzazione, e quindi un elevato consumo di risorse, almeno della tabella di ricerca completa della sua S-Box a 8 bit o del codice completo per la sua implementazione bit-slice, nessuna delle quali può essere resa molto piccola. Mentre l’AES è un cifrario a blocchi ragionevolmente leggero, la sua grande S-Box, le grandi dimensioni del blocco e la vulnerabilità intrinseca alla SCA significano che ci sono obiettivi di progettazione estremi che è improbabile che soddisfi, come una dimensione del codice molto piccola o un’implementazione mascherata molto efficiente. Un altro caso in cui sono necessari algoritmi leggeri dedicati è per l’hashing. In effetti, le funzioni hash standard necessitano di grandi quantità di memoria per memorizzare sia i loro stati interni che il blocco su cui operano. Per cui, allo stesso modo dovrò ottimizzare questo algoritmo.

9.4.2 Trends in LC

La leggerezza può essere vista come un insieme di vincoli specifici di progettazione vincoli di progettazione. Questi vengono affrontati in modo diverso dai vari algoritmi ma alcune tendenze emergono quando si guarda all’evoluzione dei cifrari a blocchi leggeri.

S-box (substitution-box) è un componente di base che esegue sostituzioni. Vengono tipicamente utilizzate per oscurare la relazione tra la chiave e il testo cifrato (proprietà di confusione di Shannon).

La non linearità è una proprietà necessaria di qualsiasi primitiva crittografica. Può essere fornito da S-Box o attraverso l’uso di operazioni aritmetiche non lineari o “S-Box a fette di bit” che

utilizzano operazioni bit per bit di base come XOR, AND e OR.

Un vantaggio delle S-Box è il semplice argomento di sicurezza basato ad esempio sulla strategia del sentiero largo che consentono.

Una semplice implementazione a fette di bit è relativo anche ad una piccola area per la implementazione hardware, significato che tali algoritmi possono essere previsti per funzionare bene anche nell'hardware. Le **Slice-S-Box** sono una scelta popolare per la progettazione di leggerezza algoritmi. Gli algoritmi basati su ARX si basano sull'addizione modulare per fornire non linearità mentre le rotazioni word-wise e XOR forniscono diffusione, da qui il nome: Addizione, Rotazione, XOR.

9.4.3 Trade-Offs in LC

Buone prestazioni e buona sicurezza sono in contrasto l'una contro l'altra. Sebbene questo compromesso non sia specifico della crittografia leggera, è più urgente in questo contesto a causa dei maggiori vincoli di prestazioni che si verificano in questo caso. Un compromesso più specifico per la crittografia leggera è quello tra specializzazione e versatilità. Alcune soluzioni sono state ottimizzate per un basso numero di gate nell'implementazione hardware. D'altra parte, l'enorme numero di round significa che un'implementazione a bassa latenza sarebbe molto difficile da progettare. I cifrari a blocchi a bassa latenza utilizzano una funzione di round più complessa che utilizza diverse matrici binarie per diverse parti dello stato interno, il che complicherebbe un'implementazione serializzata. Alcuni algoritmi sono stati esplicitamente progettati per fornire buone prestazioni su molte piattaforme diverse facendo affidamento solo su operazioni logiche e rotazioni a livello di parola. Gli algoritmi per i quali un livello di sicurezza inferiore è stato ritenuto sufficiente dai progettisti tendono a essere progettati tenendo conto di una piattaforma e di un caso d'uso specifici. D'altra parte, gli algoritmi più generalisti fanno scelte relativamente più prudenti. Vediamo quindi due sottoinsiemi di algoritmi, alcuni molto specializzati e che compiono volentieri scelte progettuali più audaci per migliorare le prestazioni, mentre altri cercano versatilità e margini di sicurezza più conservativi.

9.4.4 Hashing

Nel mondo IoT spesso misuriamo la capacità di memoria in pochi kilobyte e dove i processori a 8 bit vanno per la maggiore. Pertanto, le funzioni hash crittografiche MD5 e SHA-1 e la maggior parte dei nostri altri metodi hash moderni non sono efficienti per i dispositivi IoT. Il NIST ha quindi raccomandato nuovi metodi di hashing in grado di produrre un'impronta di memoria molto più piccola e avere come target un input di soli 256 caratteri (mentre tipicamente le funzioni hash supportano fino a 264 bit).

- **SPONGENT** usa la funzione sponge, dove c'è una permutazione (o trasformazione) di lunghezza fissa e una regola di riempimento, che trasforma l'input in blocchi di r bit. Questa costruzione quindi prende un input di lunghezza variabile e lo associa a un output di lunghezza variabile.
- **Lesamnta-LW** utilizza un design basato su AES in modo da ottenere chiari vantaggi rispetto a noti progetti basati su cifrari a blocchi come SHA-256 e MAME. Lesamnta-LW utilizza un cifrario a blocchi di 64 round E che accetta come input una chiave a 128 bit e un testo in chiaro a 256 bit.
- **PHOTON** è un metodo di crittografia leggero per l'hashing e si basa su un approccio di tipo AES.

9.4.5 Streaming Ciphers

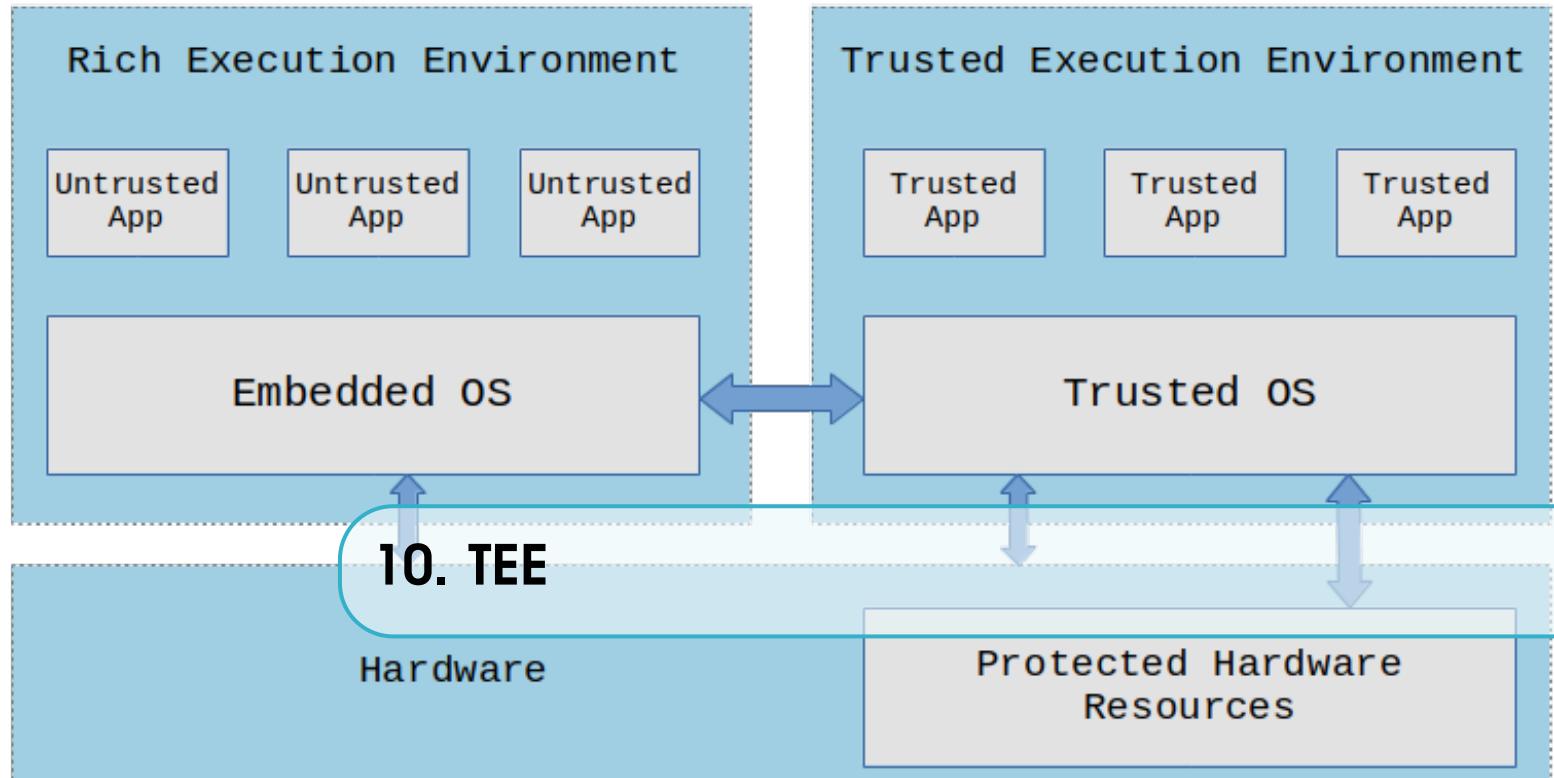
- **RC4** è un cifrario a flusso e un algoritmo a chiave di lunghezza variabile, che crittografa un byte alla volta (o unità più grandi alla volta). Un generatore di bit pseudocasuali produce un numero di flusso a 8 bit imprevedibile senza la conoscenza della chiave di input. Il flusso di chiavi viene combinato un byte alla volta con il cifrario del flusso di testo in chiaro utilizzando l'operazione X-OR. Alcuni meccanismi possono essere introdotti per ridurre i costi di implementazione di RC4 o altri cifrari a flusso:
 - **Linear Feedback Shift Register (LFSR)**;
 - **Feedback con Carry Shift Register (FCSR)**: Una variazione dell'LFSR in cui viene utilizzato un registro secondario per memorizzare il riporto. FCSR ha due implementazioni, quella di Fibonacci consiste in un'unica funzione di feedback che dipende da più input o il Galois ha più funzioni di feedback con un ingresso comune. Viene utilizzata la configurazione F-FCSR Galois in quanto più efficace.
- **Grain** è un cifrario a flusso sincrono orientato ai bit, il cui design è molto facile da implementare nell'hardware e richiede solo una piccola area del chip. Consiste di due registri a scorrimento a 80 bit in cui uno ha un feedback lineare (LFSR) e l'altro un feedback non lineare (NFSR).
- **BEAN** ha alcune somiglianze superficiali con Grain: NFSR e LFSR sono sostituiti da due FCSR. LFSR in Grain viene utilizzato per fornire un periodo ampio e per garantire proprietà di tipo casuale mentre NFSR viene utilizzato per fornire non linearità

9.5 Block Ciphers

- **PRESENT** è un algoritmo sostituto per AES per la crittografia leggera che utilizza blocchi di dimensioni più piccole e il potenziale per chiavi più piccole (come per una chiave a 80 bit). Utilizza una crittografia a 80 bit (10 caratteri esadecimali) o a 128 bit e opera su blocchi a 64 bit e con una rete di permutazione di sostituzione (SPN). Opera su blocchi e applica una chiave e quindi utilizza un numero di round con caselle di sostituzione (caselle S) e caselle di permutazione (caselle P). Il processo di decrittazione è quindi l'inverso dei cicli di crittografia. PRESENT è più compatto di AES, circa 2,5 volte più piccolo
- **CLEFIA** è un algoritmo di crittografia come alternativa sicura ad AES. L'algoritmo è costituito da due parti componenti: una parte di elaborazione dei dati e una parte di pianificazione chiave utilizzando una rete Feistel generalizzata di tipo 2 a 4 rami e una rete Feistel generalizzata a 8 rami. Il vantaggio di questa costruzione è che le funzioni sono più piccole di quelle della tradizionale struttura Feistel
- **SPARX** è una famiglia di cifrari a blocchi a 64 e 128 bit basati su ARX. Per implementare qualsiasi versione sono necessarie solo l'addizione modulo 2¹⁶ XOR a 16 bit e le rotazioni a 16 bit. I cifrari SPARX sono stati progettati secondo la Long Trail Strategy, che è una controparte della Wide-Tail Strategy, adatta per algoritmi costruiti utilizzando una S-Box grande e debole piuttosto che una piccola e forte.
- **KATAN** è stato progettato per ottimizzare l'impronta fisica

Non-Secure World

Secure World



10.1 Trusted Execution Environments

Il **Trusted Computing** (TC) è un insieme di tecnologie, protocolli e standard che mirano a garantire la sicurezza e la privacy dei dati e delle informazioni gestiti da un computer o da un sistema informatico. Ha lo scopo di garantire la sicurezza mediante la realizzazione di una sorta di "**cassaforte virtuale**" attorno ai dati ed ai programmi. Se dati o programmi esterni vogliono avere accesso a questa cassaforte devono ottenere la chiave dal sistema di TC e solamente dati e programmi autentificati possono disporre delle risorse del sistema: dall'hard disk, alla memoria RAM, alla CPU. Per applicare il TC ci sono diverse tecniche tra cui vi è la **Software Fault Isolation**, si crea una area di memoria per eseguire un codice non attendibile, così se viene rilevato un comportamento anomalo si lancia un errore e non si rischia di intaccare le componenti software. Isolare l'esecuzione del codice è uno degli approcci fondamentali per ottenere la sicurezza, è come avere due contesti differenti e ciò che isolo non dovrebbe intaccare il sistema ospitante. La virtualizzazione è un processo che crea un ambiente virtuale isolato, separato dal sistema hardware, al fine di emulare o simulare l'esecuzione di un SO, server, ecc. La virtualizzazione non è la soluzione migliore per via di alcune limitazioni:

- Dipendenza da hypervisor: un hypervisor è un software che permette di creare e gestire ambienti virtuali (VMs) su un sistema fisico e per questo motivo rappresenta una Trusted Computing Base (TCB) nella virtualizzazione. L'hypervisor fa da strato tra il s.o. guest e s.o. ospite assegnando risorse hardware (come CPU, memoria e spazio su disco) alle VMs e gestendo le richieste di accesso alle risorse da parte delle VMs. Gli hypervisor possono essere hardware o software. Una TCB di un sistema è l'insieme di tutte le componenti hardware e software critiche per la sicurezza e che quindi se intaccate compromettono il sistema. Più è grosso lo strato software, più problemi ha, maggiore è la probabilità che ci sia un bug. Quindi, più grande è il TCB, più problemi di sicurezza ho;
- Mancata gestione dell'hypervisor o dei rootkit del firmware: un rootkit è una raccolta di software dannoso che consente di accedere dove non è permesso (es. trojan). Ci sono dei rootkit che permettono ad un attaccante di accedere dalla macchina fisica alle macchine virtuali o viceversa;

- Sovraccarico del sistema: la virtualizzazione sovraccarica il sistema fisico e causa un calo di prestazioni. Questo perché sopra ad un SO ne vengono installati altri.

Per questi motivi si è passati a soluzioni container-based(es. Docker) e non si parla più di macchine virtuali e virtualizzazione ma di container e containerizzazione. La containerizzazione permette di raggruppare tutti i componenti e le dipendenze di un'applicazione in un singolo contenitore isolato, detto "container". In questo modo, l'applicazione, completa di tutte le componenti necessarie, può essere spostata ed eseguita in tutti gli ambienti e infrastrutture. Lo strato software è molto più piccolo, ha un isolamento più efficiente, però questo non mi protegge da tutte le varie problematiche. Ci sono dei possibili attacchi. Questa forma di virtualizzazione si basa sui daemon per gestire i container e le immagini delle applicazioni. Proprio come l'hypervisor per la virtualizzazione, il daemon può essere vulnerabile a causa di bug software o misconfiguration. Quindi gli ambienti di isolamento solo software sono sempre attaccabili a causa di bug e vulnerabilità.

10.1.1 Soluzioni Hardware

Combinando il concetto di esecuzione isolata mediante **containerizzazione** con tecnologie hardware si ottiene un TTE (ambiente sicuro), la soluzione sia ai problemi di isolamento, sia ai problemi di sicurezza. Un TEE consente l'archiviazione e l'esecuzione sicura delle applicazioni. Un ambiente sicuro deve offrire 3 operazioni:

- Esecuzione isolata:ogni applicazione viene eseguita in maniera indipendente dalle altre;
- Archiviazione sicura: sono garantite l'integrità e la segretezza di tutti i dati;
- Provisioning sicuro: garantisce la sicurezza delle applicazioni di terze parti fornitori di software e dati. Durante il trasferimento dei dati devo assicurare che non ci siano alterazioni.

Le soluzioni di sicurezza hardware hanno il vantaggio di ridurre notevolmente le intrusioni e gli attacchi, poiché l'attaccante deve trovarsi vicino alla macchina, deve avere conoscenze riguardo il suddetto hardware e deve usare strumenti avanzati. Lo svantaggio delle soluzioni hardware è il costo elevato che non è sempre sostenibile.

- Il **Secure Element (SE)** è un chip a microprocessore in grado di memorizzare dati sensibili ed eseguire app sicure come i pagamenti. Agisce come un vault, proteggendo ciò che è all'interno dell'SE (applicazioni e dati) dagli attacchi malware tipici dell'host (ovvero il sistema operativo del dispositivo). E' solo un ambiente di esecuzione, non di programmazione, poiché io non posso aggiungerci niente e ed è limitata la possibilità di effettuarci operazioni. L'aggiornamento richiederebbe la riprogettazione del SE. Solo le applicazioni firmate dal produttore possono essere eseguite in un SE.
- Un **Encrypted Execution Environment (E3)** è un ambiente di esecuzione in cui il software è crittografato e consente l'esecuzione senza rivelare le istruzioni che compongono l'applicazione. Anche se un attaccante ascoltasce il bus seriale tra memoria e CPU vedrebbe tutto cifrato.
- Un **Trusted Platform Module (TPM)** è un'evoluzione della precedente soluzione. Si tratta di un ambiente a microcontrollore composto da diversi coprocessori che si occupano della parte crittografica. In questo modo la CPU non verrà rallentata in quanto l'ALU dovrà svolgere solo le operazioni inerenti al programma e non quelle di cifratura/decifratura. Il vantaggio dell'utilizzo di un TPM è che lo sviluppatore non deve sapere nulla sull'implementazione di questi algoritmi, poiché il TPM fornisce un'API. Il TPM è in grado di eseguire operazioni crittografiche molto complesse, dalla crittografia simmetrica alla crittografica asimmetrica RSA. Il TPM è in grado di calcolare valori hash di piccole porzioni di dati, come chiavi

crittografiche e certificati, ma non è sufficiente per grandi quantità di dati.

10.1.2 TEE

Il **TPM** si limita ad eseguire le operazioni crittografiche. Quindi non possiamo affidarci a soluzioni puramente in hardware, ma bisogna trovare un equilibrio tra hardware e software. Un TEE è una soluzione, è una componente che **combina hardware e software** e fornisce un ambiente di esecuzione protetto per l'elaborazione di dati sensibili e garantisce che i dati riservati vengano mantenuti protetti da accessi non autorizzati. Questo ambiente è separato dal sistema operativo principale del dispositivo e utilizza una propria memoria, processore e sistema operativo. Il TEE divide il sistema in due ambienti di esecuzione:

- **Trusted:** rappresenta il sistema operativo sicuro responsabile dell'esecuzione di operazioni sensibili. Il TCB è estremamente ridotto così da minimizzare gli attacchi;
- **Rich Execution Environment:** abbiamo il SO tradizionale e le applicazioni tradizionali. Qui il TCB è molto ampio.

La comunicazione tra queste due parti e con le periferiche avviene tramite un canale di comunicazione sicuro, in particolar modo viene protetta la comunicazione con periferiche di input e display. Il TEE, oltre a fornire un'esecuzione sicura, fornisce un'archiviazione sicura. I dati critici e sensibili sono isolati dai dati dell'utente così da migliorarne la sicurezza. I dati sono memorizzati in maniera protetta, ma devono anche essere scambiati in maniera protetta, cosa che il Secure Element non fa. Quest'ultimo comunica col processore con I2C o SPI, due protocolli seriali non sicuri (se io sono in ascolto su quel bus leggo tutto). È inoltre necessario un processo di avvio sicuro, devo assicurarmi che non vengano eseguite operazioni dannose durante l'accessione della macchina: prima di caricare il bootloader bisogna verificare l'autenticità e l'integrità del sistema operativo sicuro. Il TEE oltre a fornire le classiche modalità user e kernel ne fornisce una terza, ovvero monitor. Questa viene utilizzata per eseguire il salvataggio del contesto ed il passaggio dall'ambiente sicuro all'insicuro. Faccio un passaggio dall'ambiente non sicuro al monitor che mi verifica se ci sono le condizioni, per poi fare il passaggio all'ambiente sicuro. Il sistema operativo sicuro ha un set di istruzioni limitato, perché il TCB deve essere più piccolo possibile, le funzionalità del sistema operativo sicuro sono limitate: generazione di chiavi, cifratura, decifratura e generazione di firme. Rispetto ad altre soluzioni il TEE fornisce tutte le funzionalità tranne la resistenza alle manipolazioni. Il TEE non è pensato per rendere la memoria protetta da compromissioni come fa il Secure Element poiché i dati di quest'ultimo sono piccoli. Il TEE è un ambiente di esecuzione e ciò richiede una memoria più grande, rendere tamper resistance un banco di memoria grande costa. SE e il TPM quindi hanno più funzionalità di sicurezza fisica rispetto a TEE, essendo più piccoli. Tuttavia, TEE fornisce una maggiore potenza di calcolo che consente modelli di sicurezza più complessi.

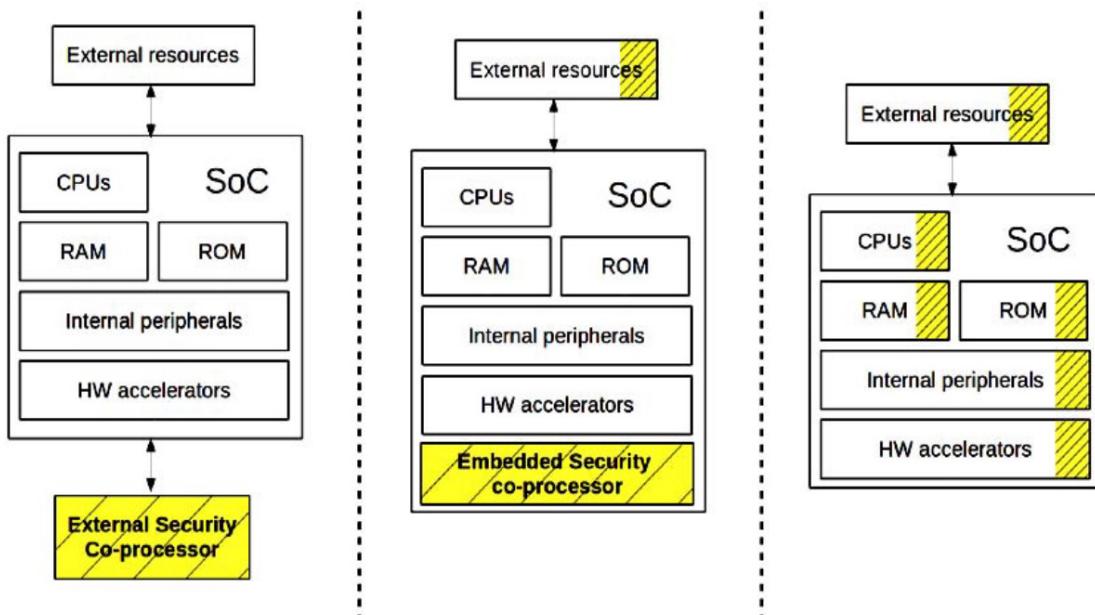
Comparison between the hardware solutions.

Criteria	SE	TEE	TPM
Tamper resistance	✓		✓
Secure input and display		✓	
High computation power		✓	✓
High storage capacity		✓	
Dependency to manufacturer	✓	✓	✓
Proven security level	✓	✓	✓

La base hardware per realizzare un TEE è il coprocessore, ovvero un **processore specializzato che es-**

egue algoritmi crittografici a livello hardware così da migliorare la velocità di cifratura/decifratura ed una conseguente miglior sicurezza dei dati. Ci sono molti modi in cui questi TEE possono essere realizzati:

- **coprocessore esterno**: il coprocessore esterno esegue le operazioni crittografiche e le attività critiche. Il vantaggio di questa soluzione è che l'esecuzione avviene isolata e possibilmente in simultaneo con il processore principale. Lo svantaggio è il sovraccarico causato dal continuo trasferimento di dati tra coprocessore e processore principale. Il co processore esterno è una soluzione economica, le chiavi devono essere nel processore esterno
- **coprocessore interno**: il coprocessore crittografico si trova all'interno del processore. Un esempio sono i processori ARM TrustZone;
- **Separazione dell'architettura**: è un approccio non basato sull'hardware ma architettonico. Il TEE è realizzato per separazione dell'architettura. Il processore impedisce alle informazioni critiche di uscire ed entrare tramite contenitori logici. Un esempio, è l'utilizzo di Docker.





11. Communication Security

**Is Your IoT
Device Safe and Secure?**
Managing Security Risks Still a Concern for Many!

11.1 Secure Communication Means

11.1.1 Secure Communication

Alice, il mittente, vuole inviare i dati a Bob, il destinatario. Per scambiare i dati in modo sicuro, Alice e Bob si scambieranno messaggi generalmente crittografati. Alice e Bob comunicano su un canale insicuro e un intruso può potenzialmente eseguire varie operazioni come intercettazione o cancellazione. Bob vuole anche essere sicuro che il messaggio che riceve da Alice sia stato effettivamente inviato da Alice, e Alice vuole assicurarsi che la persona con cui sta comunicando sia davvero Bob (autenticazione). Bisogna anche assicurarsi che i contenuti dei loro messaggi non siano stati alterati durante il transito (integrità) e che il destinatario sia l'unico a poter comprendere il messaggio (confidenzialità). Abbiamo diversi livelli di attacco (dal basso verso l'alto) mappati sul modello ISO/OSI, questo è un fattore da considerare quando effettuo il design del mio sistema con annesse scelte in base alla tecnologia da usare, poiché ogni tecnologia è suscettibile ad attacchi specifici che variano in base ad essa:

- Il livello 1 si riferisce all'aspetto fisico del networking, in altre parole, il cablaggio e l'infrastruttura utilizzati per la comunicazione tra le reti. Gli attacchi di livello 1 si concentrano sull'interruzione di questo servizio in ogni modo possibile, provocando principalmente attacchi Denial of Service (DoS). Questa interruzione potrebbe essere causata dal taglio fisico del cavo fino all'interruzione dei segnali wireless intercettando i pacchetti (sniffing).
- Il livello 2 si concentra su frame di dati, qui un possibile attacco è l'interferenza nel meccanismo di switching.
- Gli attacchi al livello 3 possono essere eseguiti in remoto tramite Internet, mentre gli attacchi di livello 2 avvengono principalmente all'interno di una LAN. Gli attacchi verso il protocollo di livello 3 sono principalmente tentativi DoS contro i router. Lo sniffing dei pacchetti può essere eseguito per intercettare dati sensibili.
- Il livello 4 include protocolli come Transport Control Protocol (TCP) e Universal Data Protocol (UDP). La scansione delle porte, ovvero l'identificazione delle porte aperte per ottenere l'accesso a un dispositivo vittima e l'inserimento di dati contraffatti in una connessione TCP

ATTACK SCENARIOS	IMPORTANCE LEVEL
1. Against the network link between controller(s) and actuators	High – Crucial
2. Against sensors, modifying the values read by them or their threshold values and settings	High – Crucial
3. Against actuators, modifying or sabotaging their normal settings	High – Crucial
4. Against the administration systems of IoT	High – Crucial
5. Exploiting protocol vulnerabilities	High
6. Against devices, injecting commands into the system console	High – Crucial
7. Stepping stones attacks	Medium – High
8. DDoS using an IoT botnet	Crucial
9. Power source manipulation and exploitation of vulnerabilities in data readings	Medium – High
10. Ransomware	Medium – Crucial ⁷⁰

esistente sono un esempio di attacchi di livello 4.

- Ai livelli superiori dello stack ISO/OSI esiste una serie di protocolli in cui possono verificarsi attacchi. Il dirottamento della sessione è l'attacco più comune al livello 5, ma la maggior parte degli attacchi avviene al livello 7, dove si trovano i protocolli di comunicazione a livello di applicazione. Poiché il traffico HTTP sta dominando Internet, gli attacchi DoS contro il server HTTP sono comuni. L'utilizzo più comune del protocollo HTTP è una richiesta GET e un avversario può eseguire un attacco DoS generando un volume elevato di richieste in modo da sovraccaricare il server e interrompere l'elaborazione di richieste GET legittime.

11.1.2 End-point Authentication

L'autenticazione dell'endpoint è il processo di un'entità che dimostra la propria identità a un'altra entità su una rete di computer. Se Alice ha un indirizzo di rete noto (ad esempio un indirizzo IP) da cui comunica sempre, Bob potrebbe tentare di autenticare Alice verificando che l'indirizzo di origine sul datagramma IP che trasporta il messaggio di autenticazione corrisponda all'indirizzo noto di Alice indirizzo. Questa soluzione è vulnerabile allo spoofing IP alterando il contenuto dell'header (IP spoofing è la denominazione con la quale si indica una tecnica di attacco informatico che utilizza un pacchetto IP nel quale viene falsificato l'indirizzo IP del mittente. Nell'header di un pacchetto IP si trova uno specifico campo, il Source Address, il cui valore indica l'indirizzo IP del mittente: modificando questo campo si può far credere che un pacchetto IP sia stato trasmesso da una macchina host diversa). Un approccio più sicuro utilizzerebbe una password segreta tra l'autenticatore e la persona da autenticare. Viene naturalmente eseguita la cifratura della password (infrastruttura PKI) facendo in modo che Alice e Bob condividano una chiave segreta simmetrica, ma questa tecnica non è del tutto sicura poiché l'attaccante deve solo origliare la comunicazione di Alice, registrare la versione crittografata della password ed inviarla a Bob per fingere di essere Alice. Bob non è in grado di distinguere tra l'autenticazione originale di Alice e la successiva riproduzione dell'autenticazione originale di Alice. Per risolvere il problema viene utilizzato un nonce crittografato, che è un numero che sarà utilizzato solo una volta nella vita. Una volta che un protocollo utilizza un nonce, non utilizzerà mai più quel numero. Quest'ultima strategia è comunque vulnerabile agli attacchi men in the middle. In questo caso una possibilità è usare un meccanismo di challenge-response, alla challenge potrà rispondere soltanto Alice.

11.1.3 Jamming Attacks

Un attacco di disturbo è la trasmissione di segnali radio che interrompono le comunicazioni, favorendo l'attenuazione del segnale, diminuendo il rapporto segnale-inferenza-più rumore (SINR), che è il rapporto tra la potenza del segnale e la somma della potenza dell'interferenza da altri segnali interferenti e della potenza del rumore. L'attacco danneggia le comunicazioni wireless mantenendo il mezzo occupato e corrompendo i segnali ricevuti. Durante l'attacco, l'attaccante sceglie una frequenza, in modo da compromettere con successo le comunicazioni tra alcuni dei nodi.

Una possibile soluzione è utilizzare una bassa potenza di trasmissione per rendere difficile ai dispositivi di disturbo il rilevamento di trasmissioni legittime, oppure impiegare il Frequency Hopping Spread Spectrum (FHSS) per eludere i disturbi passando rapidamente tra le diverse frequenze dei canali di trasmissione. Il mittente non utilizza un'unica frequenza per trasmettere i dati, ma vengono utilizzate più frequenze. La frequenza viene modificata periodicamente seguendo una sequenza specifica, nota come sequenza di hopping o codice di diffusione. La quantità di tempo trascorso su ciascuna frequenza è nota come tempo di permanenza. FHSS offre vantaggi come:

- I segnali FHSS sono altamente resistenti alle interferenze a banda stretta.
- Lo sniffing è difficile se non si conosce lo schema del salto di frequenza.

11.2 Data Link-Level Security

11.2.1 WEP

I meccanismi di sicurezza inizialmente standardizzati nella specifica 802.11 sono noti collettivamente come Wired Equivalent Privacy (WEP), che ha lo scopo di fornire un livello di sicurezza simile a quello trovato nelle reti cablate. WEP, per fornire l'autenticazione e la crittografia dei dati tra un host e un punto di accesso wireless, utilizza un approccio a chiave condivisa simmetrica. WEP non specifica un algoritmo di gestione delle chiavi. L'autenticazione viene eseguita come segue:

- Un host wireless richiede l'autenticazione da parte di un punto di accesso;
- Il punto di accesso risponde alla richiesta di autenticazione con un valore nonce di 128 byte;
- L'host wireless crittografa il nonce utilizzando la chiave simmetrica che condivide con il punto di accesso.
- Il punto di accesso decrittografa il nonce crittografato dall'host. Se il valore nonce decrittografato corrisponde al valore nonce originariamente inviato all'host, l'host viene autenticato.

11.2.2 WPA

Wi-Fi Protected Access è un miglioramento di WEP con 256 chiavi (invece di 128 in WEP) e un IV più lungo. È un protocollo provvisorio prima dello sviluppo del più sicuro Wi-Fi-protected access 2 (WPA2) o IEEE 802.11i. Lo standard WPA funziona in una delle due modalità:

- Chiave **WPA personale** o **WPA-PSK**: Il suo sistema è semplice da configurare e tutti gli utenti potrebbero voler utilizzare una passphrase. Tuttavia, se un dispositivo viene compromesso, tutti i dispositivi sulla rete dovrebbero cambiare le proprie password.
- Chiave **WPA enterprise**: gli utenti devono utilizzare le proprie identità personali per accedere alla rete tramite un server RADIUS (Remote Authentication Dial-In User Service) è un protocollo per l'autenticazione, l'autorizzazione per gli utenti che si connettono e utilizzano un servizio di rete. È un protocollo client/server basato su TCP o UDP. I punti di accesso alla rete di solito dispongono di un client RADIUS che comunica con il server RADIUS

per negare/consentire una richiesta utente a seconda delle attestazioni confrontate con un'origine appropriata. Se un dispositivo viene violato, gli amministratori possono annullarne l'accesso indipendentemente dagli altri dispositivi.

Gli elementi chiave sono i seguenti:

- Temporal Key Integrity Protocol (TKIP): CRC è sostituito da MICHAEL, un protocollo migliore per MIC (Message Integrity Check), progettato per evitare l'ipotesi iterativa e il bit flipping a cui WEP è vulnerabile. Inoltre, si basa sull'intero frame, evitando così gli attacchi di frammentazione. Un segreto condiviso, ovvero una password o una chiave di autenticazione, viene trasformato in una chiave master pair-wise (PMK) di 256 bit. Le chiavi transitorie a coppie (PTK) per ciascuna sessione sono formate utilizzando una funzione pseudocasuale che agisce su PMK, indirizzi di terminali di sessione e nonce.
- Gli aggressori tentano di modificare i frame e di inviarli e vedere se i frame modificati vengono scambiati per autentici. Il più delle volte falliscono. Con WEP, un frame non decifrabile viene eliminato silenziosamente, senza alcun danno. Per evitare che questi attacchi abbiano successo, WPA aggiunge il concetto di contromisure. Se vengono ricevuti due frame con MIC difettosi in un intervallo di 60 secondi, il punto di accesso avvia tutti i client e richiede loro di rinegoziare nuove chiavi. Questo passaggio drastico introduce una dolorosa vulnerabilità di negazione del servizio in TKIP, ma è necessario per impedire agli aggressori di ottenere facilmente informazioni.

11.2.3 IEEE 802.11i o WPA2

802.11i è un miglioramento dei meccanismi di sicurezza applicati alla famiglia dello standard 802.11. Mentre WEP fornisce una crittografia relativamente debole, un solo modo per eseguire l'autenticazione e nessun meccanismo di distribuzione delle chiavi, IEEE 802.11i fornisce forme di crittografia molto più potenti (utilizza l'algoritmo di crittografia AES anziché TKIP), un set estensibile di meccanismi di autenticazione e un meccanismo di distribuzione delle chiavi. 802.11i definisce un server di autenticazione con cui l'AP può comunicare. CCM mode Protocol (CCMP) è un protocollo di crittografia progettato per IEEE 802.11i, basato sulla Counter Mode con CBC-MAC (modalità CCM) dello standard Advanced Encryption Standard (AES), che fornisce sia l'autenticazione che la riservatezza.

11.2.4 Bluetooth Security

La sicurezza Bluetooth è molto importante e viene fornita sui vari collegamenti wireless, solo sui percorsi radio. In altre parole, è possibile fornire l'autenticazione e la crittografia del collegamento, ma non è possibile una vera sicurezza end-to-end senza fornire soluzioni di sicurezza di livello superiore oltre al Bluetooth. Le ultime versioni di Bluetooth hanno aumentato i livelli di sicurezza per combattere la minaccia degli hacker. Ogni dispositivo Bluetooth deve funzionare in una delle tre modalità:

- modalità 1: questa è una modalità **non sicura**, non implementa alcun tipo di meccanismo per evitare che altri dispositivi bluetooth possano stabilire connessioni;
- modalità 2: abbiamo un **controllore che gestisce gli accessi**;
- modalità 3: le **procedure di sicurezza** sono inizializzate prima che venga stabilito il canale.

La chiave di collegamento viene generata durante una fase di inizializzazione quando un utente inserisce un PIN identico in entrambi i dispositivi, mentre due dispositivi Bluetooth che stanno comunicando sono "associati". Bluetooth fornisce la crittografia per proteggere i payload dei

pacchetti scambiati tra due dispositivi. La funzione di crittografia accetta come input l'identità del master, il numero casuale, un numero di slot e una chiave di crittografia, che inizializzano gli LFSR prima della trasmissione di ciascun pacchetto, se la crittografia è abilitata. La procedura di autenticazione Bluetooth è sotto forma di uno schema "challenge-response". Il protocollo challenge-response convalida i dispositivi verificando la conoscenza di una chiave segreta, una chiave di collegamento Bluetooth. Oltre alle tre modalità di sicurezza, Bluetooth consente due livelli di affidabilità e tre livelli di sicurezza del servizio. I due livelli di attendibilità sono "affidabile" e "non attendibile". I dispositivi fidati sono quelli che hanno una relazione fissa e quindi hanno pieno accesso a tutti i servizi. I dispositivi non attendibili non mantengono una relazione permanente; ciò si traduce in un accesso limitato al servizio. Per i servizi sono stati definiti tre livelli di sicurezza.

- Livello di servizio 1: richiedono autorizzazione e autenticazione. L'accesso automatico è concesso solo ai dispositivi attendibili. I dispositivi non attendibili richiedono l'autorizzazione manuale.
- Livello di servizio 2: richiedono solo l'autenticazione. L'accesso ad un'applicazione è consentito solo dopo una procedura di autenticazione. L'autorizzazione non è necessaria.
- Livello di servizio 3: sono aperti a tutti i dispositivi. L'autenticazione non è richiesta e l'accesso viene concesso automaticamente.

11.2.5 ZigBee Security

La sicurezza delle comunicazioni è uno dei punti di forza di ZigBee e segue quella definita in IEEE 802.15.4, fornendo meccanismi che controllano l'accesso ai dispositivi di rete (autenticazione), la crittografia (crittografia a chiave simmetrica) e l'integrità, utilizzando i controlli di integrità dei messaggi (MIC). L'architettura di sicurezza di ZigBee si basa sull'uso della crittografia a chiave simmetrica e dispone di un elaborato protocollo di gestione delle chiavi. ZigBee definisce i livelli di rete e applicazione sopra il MAC:

- Il livello di rete ZigBee garantisce l'integrità e la crittografia dei frame trasmessi applicando la crittografia AES e ne garantisce l'integrità utilizzando un codice di autenticazione del messaggio concatenato a blocchi di cifratura.
- Il livello dell'applicazione consente di basare la sicurezza del frame sulle chiavi di collegamento o sulla chiave di rete. La chiave di rete viene utilizzata per proteggere la comunicazione broadcast, mentre la chiave di collegamento viene utilizzata per proteggere la comunicazione unicast.

11.3 Network-Level Security

11.3.1 IPSec

Il protocollo di sicurezza IP, o IPSec, fornisce sicurezza a livello di rete proteggendo i datagrammi IP tra due entità a livello di rete, inclusi host e router. I punti deboli sono mancanza di integrità e autenticazione che possono portare ad un IP spoofing, e mancanza di confidenzialità che comporta sniffing dei pacchetti. IP Spoofing è la creazione di pacchetti IP (Internet Protocol) con un falso indirizzo IP di origine, allo scopo di impersonare un altro sistema informatico. Viene utilizzato più frequentemente negli attacchi denial-of-service. Questo protocollo è trasparente per le applicazioni (sotto TCP/UDP), quindi non è necessario modificare il software su un sistema utente o server quando IPSec è implementato nel firewall o nel router. Nella suite di protocolli IPSec, ci sono due protocolli principali, il protocollo Authentication Header (AH) che fornisce l'autenticazione della fonte e l'integrità dei dati, ma non la riservatezza e il protocollo Encapsulation Security Payload

(ESP) che fornisce l'autenticazione della fonte (opzionale), l'integrità dei dati e la riservatezza. Poiché la riservatezza è spesso fondamentale per varie applicazioni, il protocollo ESP è molto più utilizzato rispetto al protocollo AH. I datagrammi IPSec vengono inviati tra coppie di entità di rete, che creano una connessione logica a livello di rete, chiamata associazione di sicurezza (SA), che è unidirezionale dall'origine alla destinazione. Se entrambe le entità desiderano scambiarsi datagrammi sicuri, è necessario stabilire due SA (ovvero due connessioni logiche), una in ciascuna direzione. ISAKMP è un protocollo per stabilire SA e chiavi crittografiche in un ambiente Internet, per l'autenticazione e lo scambio di chiavi ed è progettato per essere indipendente dallo scambio di chiavi.

11.4 Transport-Level Security

11.4.1 SSL

Durante un acquisto, senza riservatezza, un intruso potrebbe intercettare l'ordine di Bob e ottenere i dati della sua carta di pagamento. L'intruso potrebbe quindi fare acquisti a spese di Bob. Senza integrità dei dati, un intruso potrebbe modificare l'ordine di Bob, facendogli acquistare dieci volte più flaconi di profumo di quelli desiderati. Senza autenticazione del server, un server potrebbe impersonare il sito. SSL risolve questi problemi migliorando il protocollo TCP con riservatezza, integrità dei dati, autenticazione del server e autenticazione del client. SSL fornisce una semplice API (Application Programmer Interface) con socket, simile e analoga a TCP. Sebbene SSL risieda tecnicamente nel livello dell'applicazione, dal punto di vista dello sviluppatore è un protocollo di trasporto che fornisce i servizi di TCP migliorati con i servizi di sicurezza. SSL è composto da due fasi:

- Viene utilizzato lo schema di handshake SSL per stabilire un canale sicuro, affidabile e autenticato tra client e server;
- I protocolli SSL Record encapsulano i messaggi in blocchi autenticati e cifrati.

Durante la fase di handshake, Bob deve stabilire una connessione TCP con Alice e verificare che Alice sia davvero Alice. Successivamente invia ad Alice una chiave master secret, da utilizzare per generare tutte le chiavi simmetriche necessarie per la sessione SSL.

11.4.2 TLS

SSL e TLS sono entrambi protocolli crittografici che forniscono l'autenticazione e la crittografia dei dati su una rete (ad esempio un client che si connette a un server Web). SSL è il predecessore di TLS. TLS è lo standard di crittografia utilizzato da tutti e viene spesso utilizzato insieme ad altri protocolli Internet come HTTPS, SSH, FTPS e posta elettronica sicura. Differenze tra SSL e TLS:

- Il protocollo TLS non supporta le suite di cifratura Fortezza/DMS mentre SSL supporta Fortezza.
- Il protocollo SSL aggiunge MAC dopo aver compresso ogni blocco e lo crittografa. Al contrario, il protocollo TLS utilizza il codice di autenticazione dei messaggi basato su hash
- In SSL per creare un master secret, viene utilizzato il message digest del pre-master secret. Al contrario, TLS utilizza una funzione pseudocasuale per generare il master secret.

11.4.3 DTLS

Datagram Transport Layer Security (DTLS) è un protocollo di comunicazione che fornisce sicurezza per le applicazioni basate su datagrammi. Il protocollo DTLS ha un Record Layer, per crittografare i payload, dividerli in frammenti e incapsularli in pacchetti strutturati, chiamati record, per fornire l'autenticazione del messaggio. Ciò impedisce l'intercettazione, la manomissione o la contraffazione dei messaggi. Durante l'handshake inizializzato dal client vengono negoziate la sicurezza, l'autenticazione e si stabilisce un canale sicuro. DTLS implementa la ritrasmessione utilizzando un singolo timer in ogni punto finale e ritrasmettendo l'ultimo messaggio finché non viene ricevuta una risposta. Questo comportamento è formalizzato utilizzando una macchina a stati.

11.4.4 HTTPS

Hypertext Transfer Protocol Secure (HTTPS) è un'estensione di Hypertext Transfer Protocol (HTTP) per proteggere la comunicazione su una rete di computer utilizzando TLS. Le principali motivazioni per HTTPS sono l'autenticazione del sito Web a cui si accede e la protezione della privacy e dell'integrità dei dati scambiati durante il transito. Protegge dagli attacchi man-in-the-middle e la crittografia bidirezionale delle comunicazioni tra un client e un server crea un canale sicuro su una rete non sicura per proteggere da intercettazioni e manomissioni.