Alma Mater Studiorum
Università di Bologna campus
di Cesena

Tesi di Laurea in ingegneria e scienze informatiche

Applicazione di Secure Dynamic Messaging a dispositivi RFID

Relatore	**Presentata da:**
Prof.re Margara Luciano	Rettaroli Andrea
Correlatore	
Prof.re Calderoni Luca	

Anno accademico 2019/2020

Indice

Tabella delle abbreviazioni

Acronimo	Descrizione
AES	Advanced Encryption Standard
APDU	Application Protocol Data Unit
API	Application Programming Interface
AWS	Amazon Web Services
C-APDU	Command APDU
CMAC	MAC according to NIST Special Publication 800-38B
DF-name	ISO7816 Dedicated File Name
DES	Data Encription Standard
ECC	Elliptic Curve Cryptography
EEPROM	Electrically Erasable Programmable Read-Only Memory
FIPS	Federal Information Processing Standards
HF	High Frequency
IBM	International Business Machines Corporation
IC	Integrated Circuit
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
IFF	Identification Friend or Foe
IV	Initialization Vector
LF	Low Frequency
LLCP	Logical Link Control Protocol
LRP	Leakage Resilient Primitive
LSB	Lowest Significant Byte
MAC	Message Authentication Code
MF Level	Master File Level
NDEF	NFC Data Exchange Format
NIST	National Institute of Standards and Technology

Acronimo	Descrizione
NFC	Near Field Communication
NSA	Nation Security Agency
OSI	Open Systems Interconnection
P2P	Peer to Peer
PCD	Proximity Coupling Device
PICC	Proximity Integrated Circuit Card
POS	Point of Sale
R-APDU	Response APDU(ricevuta dal PICC)
RFID	Radio-Frequency identification technology
RFU	Reserved for Future Use
SDM	Secure Dynamic Messaging
SDMReadCtr	Secure Dynamic Messaging Read Counter
SUN	Secure Unique NFC Messaging
UHF	Ultra High Frequency
UID	Unique IDentifier
URI	Uniform Resource IDentifier
URL	Uniform Resource Locator

Introduzione

Nata durante la seconda guerra mondiale e sviluppatasi a partire dagli anni sessanta per fini militari e civili, diffusasi poi negli anni novanta, la Radio-Frequency identification technology, nota anche come RFID, è oggi molto comune. Mira all'identificazione automatica di soggetti, oggetti o animali e allo scambio di informazioni, servendosi dei Tag che vengono interrogati attraverso un canale a radiofrequenza dai lettori tramite la comunicazione wireless. Ognuno di noi, ogni giorno, ha a che fare con l'RFID technology e ne sfrutta le potenzialità senza nemmeno rendersene conto, ad esempio quando si utilizza la carta di credito per pagare in maniera contactless o si striscia il proprio badge vicino al lettore. Attualmente è considerata una tecnologia a scopo generale, viste le sue possibilità di applicazione in una vastissima gamma di settori: si pensi che viene ritenuta un elemento alla base del processo tecnologico di una smart city capace di creare un'infrastruttura finalizzata alla raccolta dati di persone e oggetti e di rendere l'intera rete interconnessa ad altre tecnologie per scambiare e integrare le informazioni. Intorno ad essa ruotano applicazioni di tipo logistico, di identificazione, di pagamento, di tracciamento, di monitoraggio, di gestione delle risorse e di controllo degli accessi. La proprietà fondamentale di questa tecnologia è la sua forte pervasività all'interno dei settori produttivi e delle relative imprese coinvolti nella realizzazione di una determinata produzione. Spesso si hanno più necessità legate ad un prodotto e ciò fa si che vi siano diversi costi: vi sarà un costo per il tracciamento, uno per il monitoraggio, uno per il conteggio, uno per la sicurezza; grazie a questa tecnologia, tali molteplicità vengono incanalate in un Tag capace di rispondere a queste esigenze riducendo drasticamente i costi. Un altro aspetto chiave di questa tecnologia è la sicurezza, che implica segretezza dell'informazione, anticontraffazione, garanzia, aspetti che nel mercato odierno sono fondamentali: questa proprietà è resa possibile grazie dalla crittografia e dalle applicazioni crittografiche che essa stessa supporta, e oggi la sicurezza sui dati è uno dei temi più caldi a livello informatico. Un altro concetto chiave che emerge è la confidenzialità. Tramite l'utilizzo di Secure Dynamic Messagging, SDM, un processo che cifra una parte dell'informazione nel chip, la comunicazione tra Tag e Reader avviene in chiaro garantendo solo parzialmente la confidenzialità. Benché questo riduca leggermente le garanzie in termini di sicurezza di un'applicazione RFID-based, ne aumenta esponenzialmente la facilità di distribuzione ma soprattutto la compatibilità e l'interoperabilità con altri sistemi preesistenti. Si abbatte la problematica di condivisione delle chiavi crittografiche con i Reader e l'installazione di software dedicato sugli stessi. Mi pongo l'obiettivo di creare un'applicazione che configuri in maniera confidenziale dei Tag, nello specifico NTAG 424 DNA Tag Tamper, che supportano la Near Field Communication NFC e, grazie alle caratteristiche di RFID e

SDM, siano in grado di interoperare con dei sistemi già esistenti, nel mio caso di interagire con un sito web. Gli scenari in cui vengono sfruttate al massimo le funzionalità integrate nei Tag sono molto vasti e interconnessi: il mio scopo è, dunque, garantire un adeguato livello di sicurezza senza minare la caratteristica fondamentale dei sistemi complessi, l'interoperabilità.

RFID

Un po' di storia

Il concetto chiave alla base dell'RFID nasce nel 1939, tutto parte dall'invenzione inglese degli IFF, Identification Friend or Foe, un sistema militare che aveva l'obiettivo di identificare un amico o un nemico interrogando il bersaglio, che solitamente disponeva di un radar capace di emettere impulsi elettromagnetici i quali impattavano i transponder[1] presenti su altri aerei e registravano l'eco di ritorno come un'informazione in grado di fornire la distanza: militarmente parlando, se l'informazione era ritenuta valida, si identificava un amico, altrimenti un nemico. I sistemi IFF potevano già comunicare in maniera criptata. Nel 1945 ci fu il caso noto anche come 'The Thing': dei ragazzi della Yuong Pioneer Organization of the Soviet Union realizzarono una microspia inserendo in un sigillo cerimoniale degli Stati Uniti un antenna che veniva attivata dalle onde radio ed era capace di trasmettere. All'epoca dispositivi di spionaggio erano composti da inserti elettronici, batterie o cavi di alimentazione fu così che il dispositivo passò inosservato ai controlli e spiò conversazioni per 7 anni. Nel 1973, Mario Cardullo brevettò il primo transponder[1] radio passivo dotato di 16 bit di memoria, con alimentazione a segnale, destinato ad usi doganali. Nello stesso anno, Steven Depp, Alfred Koelle e Robert Freyman dimostrarono il funzionamento dei Tag RFID a potenza riflessa, ovvero passivi e attivi. Questo sistema operava a 915Mhz di frequenza e impiegava Tag a 12bit. Oggi questa tecnica è ancora impiegata nei Tag Ultra High Frequency UHF e RFID a microonde.

Il primo brevetto ufficiale nel quale veniva menzionata l'RFID venne dato a Charles Walton nel 1983: all'epoca i codici a barre erano rivali, perchè costavano circa 0.25**$**, mentre le prime carte RFID 1.75**$**, ed è per questo che negli articoli dei supermercati ancora oggi troviamo i codici a barre. L'RFID era già noto in più varianti, ma Walton creò i primi Tag dormienti che venivano attivati da una quantità minima di corrente proveniente dai ricetrasmettitori radio o da lettori RFID che garantivano energia per generare una risposta. Da allora si susseguirono tantissimi brevetti in materia, con campi di applicazione diversi che spaziano tra logistica, meccanica e automazione, settore medico, commercio, manifattura e produzioni, documenti ufficiali, domotica ecc... Inizialmente questa tecnologia, che esiste da ormai 40 anni, non ha preso molto piede, però ai nostri giorni grandi marchi di rivenditori come Tesco, River Island, Adidas,

Decathlon, Lewis hanno appurato ritorni economici legati alle vendite fino a un +5,5% e una riduzione dei costi di stoccaggio fino a -13%. [1]

Caratteristiche tecniche di un sistema RFID

Per comprendere al meglio questa tecnologia , dobbiamo andare a definire le componenti essenziali di questo sistema e a studiare i loro ruoli. Il tipico sistema RFID si basa su tre attori fondamentali:

- Tag.
- Lettore.
- Un sistema di comunicazione dei dati e di gestione dei dati.

Approfondiamo questi tre elementi singolarmente cominciando dai Tag. Sono soliti avere la forma di un'etichetta: da un lato è subito possibile vedere l'antenna tipicamente arrotolata che serve per ricevere e trasmettere informazioni, e permette l'accensione del dispositivo alla ricezione di radiofrequenze emesse dal lettore. Attaccato ad essa, troviamo un microchip di memoria usato per la gestione dei dati. L'adesivo, che tiene il tutto compatto, tecnicamente è definito come substrato e può essere di diversi materiali: i più comuni sono di plastica PVC o PET.

Abbiamo tre categorie di Tag, *passivi*, *semipassivi*, *attivi*.

1. *Tag passivi:* la loro caratteristica principale è che rimangono inattivi sino a quando non ricevono un segnale radio dal lettore; solitamente sono privi di batteria e sono alimentati dal segnale.
2. *Tag semipassivi:* sono muniti di batteria, ma non trasmettono segnali periodici. Alla ricezione del segnale viene accesa la batteria che permette un ritorno di segnale al lettore.
3. *Tag attivi:* sono in grado di emettere un segnale periodico; trovano utilità nel tracciamento proprio per questa caratteristica.

Il costo segue l'ordine crescente nell'elenco. Un'altra caratteristica è la frequenza rigurardo alla quale facciamo altre tre distinzioni: *bassa*, *alta*, *ultra alta*.

1. *Tag RFID a bassa frequenza:* anche detti LF low frequency, solitamente lavorano tra i 30KHz e i 300KHz; la loro peculiarità è la resistenza alle interferenze causate da liquidi o metalli, questo perchè dotati di una lunghezza d'onda maggiore. Di contro hanno una minor velocità in lettura.

2. *Tag RFID ad alta frequenza:* anche detti HF, high frequency, la loro specificità è un 'ampia capacità di memoria e un ampio raggio di lettura. Tra essi troviamo i Tag NFC, Near Field Communication, che approfondiremo in seguito: per ora ci basti sapere che sono un sottoinsieme dei Tag HF.
3. *Tag RFID ad altissima frequenza:* anche detti UHF, ultra high frequency, tipicamente usati in applicazioni dove è necessario un monitoraggio a livello di singolo articolo, sono il giusto compromesso tra costo e velocità in lettura.

La tabella di seguito mostra un riassunto di questi aspetti:

	LF RFID	**HF RFID**	**UHF RFID**
Frequenza	30-300KHz	3-30MHz	300MHz-3GHz
Frequenza tipica	125KHz o 134KHz	13.56MHz (NFC)	860-960MHz (UHF generazione 2)
Costo	$$	–$	$
Distanza di lettura	≤10cm	≤30cm	≤100m
Pro	resiste alle interferenze	ampia capacità di memoria	basso costo e vasto raggio d'azione
Applicazioni comuni	tracciamento, inventari automatizzati	packaging, pagamenti contactless, biblioteche	supporto catene di rifornimento, tracciamento di singoli prodotti

Dopo aver identificato le principali caratteristiche dei Tag, passiamo oltre.

I lettori sono solitamente configurabili sotto diversi aspetti e possono avere accesso a Internet; si distinguono anche essi per la capacità di lettura a basse, alte, ultra alte frequenze. Il lettore più comune e diffuso è il cellulare: oggi tutti i telefoni con sistema operativo Android sono capaci di leggere Tag RFID perchè escono dalla fabbrica con lettore incorporato. Altri lettori comuni sono i POS, o i lettori badge, ma ne esistono di tantissimi tipi. Ciò che accomuna tutti questi dispositivi è la capacità di trasmettere un campo elettromagnetico che, tramite induzione, va ad alimentare i Tag. I sistemi di gestione dei dati sono le memorie, che variano, come detto sopra, a seconda della categoria di Tag e degli usi. Per quanto riguarda i sistemi di comunicazione tra Tag e lettore, in materia esistono diversi protocolli standard definiti da ISO[2]: per una visione più amplia si consiglia di consultare ISO Standard for RFID. Si citano i riferimenti normativi per completezza:

- ISO/IEC 18000-63, *Information technology — Radio frequency identification for item management — Part 63: Parameters for air interface communications at 860 MHz to 960 MHz Type C*

- ISO/IEC 19762, *Information technology — Automatic identification and data capture (AIDC) techniques — Harmonized vocabulary*
- ISO 20910, *Coding for radio frequency identification (RFID) tyre tags*
- ISO 20911 1, *Radio frequency identification (RFID) tyre tags — Tyre attachment classification*
- ISO 20912 2, *Conformance test methods for RFID enabled tyres*
- EPC GS1, Radio-Frequency Identity Protocols Class-1 Generation 2 UHF RFID Protocol for communications at 860 MHz – 960 MHz

In seguito verranno approfonditi alcuni standard che ci interessano più da vicino. Al momento è sufficiente sapere che anche essi differiscono a seconda della frequenza.

NFC

Near Field Communication

La Near Field Communication, nota come NFC, scritta che gli appassionati di telefonia hanno sicuramente visto svariate volte, sta ad indicare la capacità di comunicazione in prossimità del dispositivo e che esso è dotato di lettore RFID. Come affermato sopra, la NFC è un sottoinsieme del RFID alta frequenza: essa nacque nel 2002, inventata da Sony e NXP Semiconductors. In soli due anni, nel 2004, venne integrata dalle principali aziende di telefonia del tempo come Nokia e Philips: il primo telefono in commercio ad integrare questa tecnologia fu il Nokia 6131. Nel 2006 iniziarono ad essere definite le prime specifiche per i ricevitori, nacquero i primi Tag NFC con i quali era possibile interagire in maniera bidirezionale, ed è proprio questa la caratteristica fondamentale che ne determinò il passo evolutivo rispetto all'RFID technology, l'applicazione del Peer to Peer ad NFC, che avvenne nel 2009. Ciò fu permesso dalla nascita del NFC Logical Link Control Protocol LLCP, basato sullo standard di settore IEEE 802.2, che definisce un protocollo OSI di livello 2 capace di supportare P2P, fornendo una specifica di implementazione in aggiunta agli standard ISO/IEC 18092 e ISO/IEC 14443. L'anno successivo Samsung presentò il primo telefono NFC con sistema operativo Android: si trattava del Nexus S. Si parla molto di telefonia perchè questa tecnologia nasce per essere alla portata di tutti e permette di scambiare contenuti come immagini, video, file musicali, o condividere URL, in maniera molto facile passando il lettore fino a 10 cm di distanza dal ricevitore. Sempre nel 2004, venne istituito un Forum che stabilì gli aspetti fondamentali che avrebbe dovuto avere un sistema NFC:

- consentire la comunicazione a breve distanza, come già detto, fino ad un massimo di 10cm;
- essere integrato in dispositivi attivi con una funzionalità duale, ovvero che siano capaci di essere Tag e Reader. I cellulari sono il perfetto esempio;
- rientrare nelle seguenti specifiche tecniche: operare ad una frequenza di 13.56MHz, bandwidth di 2MHz, connettersi con un bit rate ponderato; solitamente il trasferimento dati avviene a 106, 212, 424, 848 Kbit/s;
- trasferire i dati secondo la codifica di Miller e con la codifica di Manchester. Per approfondimenti consultare Codifica dei dati nel mondo RFID;
- rispettare il protocollo ISO/IEC 14443 relativo alle card.

Funzionalità e innovazioni NFC

La tecnologia RFID è solita lavorare in modalità Reader o Writer: NFC supporta la suddetta modalità di lavoro, ma è dotata di altre due modalità lavorative, Peer to Peer e Emulation Card. Approfondiamone il funzionamento e gli usi tipici:

- *Reader/Writer:* Quando si lavora in modalità lettore/scrittore, la maggior parte dei dispositivi NFC agisce come lettore. Il dispositivo NFC funziona in modalità attiva per leggere il contenuto di un Tag. Per scrivere in un Tag è necessaria un'applicazione che permetta la scrittura, spesso è richiesta anche un'autenticazione messa a protezione dei dati inseriti. L'applicazione che interagisce con i dati NFC interpreta i dati dei Tag e reagisce di conseguenza. Ad esempio, durante la lettura di un Tag contenente un URI (Uniform Resource Identifier), il dispositivo apre un browser e il browser apre il contenuto puntato dall'URI. I casi d'uso più frequenti sono: l'utilizzo di un cellulare come lettore di carte, Smart Posters che contengono NFC Tags, apertura diretta di URL, componenti attive contenenti indicazioni, pubblicità ecc...
- *Peer to Peer:* Prevede la comunicazione bidirezionale che avviene solitamente tra due cellulari o dispositivi NFC; i dispositivi saranno Reader e Tag in maniera alternata, permettendo così lo scambio di dati. Solitamente sono utilizzati per condividere file di piccole dimensioni, immagini, canzoni, documenti di testo.
- *Card Emulation:* In questa modalità di comunicazione il dispositivo NFC è passivo ed agisce esattamente come una smart card; è spesso necessario che l'applicazione garantisca la protezione dei dati. Gli usi più comuni sono: l'emulazione di carte di pagamento, di carte per il controllo accessi come badge, carte di accesso alle camere d'hotel, carte fedeltà o carte socio.

Un' altra data importante per la tecnologia NFC è il 10-11 maggio 2011, quando all'evento Google I/O, tenutosi a San Francisco presso il Moscone Center, Jeff Hamilton e Nick Pelly, alla conferenza "How to NFC"[3], ne definirono gli aspetti innovativi e le caratteristiche. Si consiglia la visione completa dell'evento. I due cercarono di rispondere a "What is NFC" e lo fecero definendo quattro semplici punti:

- tecnologia wireless a corto raggio. Infatti questa tecnologia richiede un raggio nettamente inferiore rispetto alle comunicazioni WIFI e Bluetooth, teorica 10 cm ma pratica tra 1-4 cm.
- bassa velocità. Come abbiamo già detto il range di velocità di trasmissione dati varia tra i 106 e i 848 Kbps.
- low friction set-up ovvero non prevede una fase di configurazione, non necessita di connessioni, non richiede l'identificazione o configurazioni con il ricevente, lo

scambio in modalità P2P avviene in semplicità e naturalezza avvicinando i due dispositivi.
- target passivi ovvero capaci di attivarsi solo tramite un lettore.

Avevamo già definito queste caratteristiche, ma volevo rimarcarle, in quanto per Hamilton e Pelly sono proprio i punti di forza di questa tecnologia nonché le ragioni che dovrebbero spingerne l'uso e la diffusione: “Things just work: any time you bring two devices that support NFC within their air range they will set up a connection and talk to each other. No discovery like Bluetooth. No pairing. No password.”

Queste furono le parole usate a sostegno della tecnologia NFC. Nello stesso anno venne concessa la prima certificazione MasterCard per le funzionalità di PayPass ai BlackBerry, l'anno successivo, nel 2012, anche Visa certificò BlackBerry e Android per il servizio di pagamento Visa PayWave. Per Apple dobbiamo attendere l'uscita dell'Iphone 6, primo dispositivo ad integrare la tecnologia NFC per i pagamenti, nel 2014.[2]

RFID e NFC a confronto

Nella seguente tabella riassumo gli aspetti che accomunano e differenziano queste due tecnologie ampiamente presentate.

	NFC	**RFID**
Operatività	scambio dati in maniera passiva per i Tag, attivo passiva per i Reader.	scambio dati in maniera attiva e passiva
Modalità di comunicazione e trasmissione dati	Accoppiamento induttivo in campi vicini [4]	campi elettromagnetici
Frequenze	13,56MHz	LF, HF, UHF
Distanze	max 10cm	varia a seconda del Tag.

Attacchi a RFID e NFC

L'affidabilità di questa tecnologia è legata alla protezione dei dati al loro interno e allo scambio di essi in maniera confidenziale. In seguito presentiamo i principali attacchi che vengono fatti contro RFID e NFC.

Attacco Man in the Middle o sniffing

Man in the middle sta a significare un intercettazione, ovvero tra mittente e destinatario abbiamo una terza figura che intercetta, ascolta e modifica le informazioni in modo da rendere contraffatto il messaggio che arriva al destinatario. In questi sistemi si produce questo tipo di attacco deviando il segnale in modo da riceverlo e inviando poi dati contraffatti fingendosi un componente del sistema. Attacchi tipici di questo tipo sono Relay/Replay attack [5]

Cloning and Spoofing the reader

Tipicamente vengono eseguiti in coppia, uno dopo l'altro. La clonazione permette di duplicare i dati di un Tag e lo spoofing utilizza il Tag clonato avente i dati che permettono l'accesso a ciò che era protetto. Questo attacco viene fatto solitamente nelle operazione di accesso e di gestione delle risorse. Sfrutta le vulnerabilità legate all'accesso al dispositivo: spesso molte decryption keys sono presenti sul web o non vengono adottate le misure di protezione messe a disposizione dai Tag in maniera corretta.

Denial of service

Racchiude una categoria vasta di attacchi che prevedono la manomissione del sistema RFID, si simula un guasto, si genera dell'interferenza o del rumore che blocca i segnali radio, oppure si rimuove il dispositivo o lo si disabilita. Ad esempio un Signal o Frequency jamming [6] potrebbe causare questo tipo di attacchi.

Tag

Come abbiamo già illustrato un Tag tipico si compone di un chip, che contiene la memoria e il circuito integrato capace di eseguire le operazioni, ad esso è collegata una bobina ovvero un'antenna filiforme e tutto ciò è inserito in un substrato tipicamente prodotto da materiali plastici. La forma generalmente è rotonda o quadrata: da un lato è possibile vedere l'antenna e il chip, lato che presenta la parte applicabile per incollare il Tag; dall'altro lato, solitamente, vi è l'etichetta stampata che identifica il produttore e il tipo di Tag. La produzione avviene costruendo dei piani circolari di finissimo spessore, con un diametro fino a 30 cm, denominati "wafer": in essi sono presenti migliaia di chip che vengono contraddistinti dall'Unique Identifier, UID, inserito in fase di preconfigurazione, e testati sulle funzionalità integrate per esser poi tagliati. Ogni circuito integrato viene inserito in un substrato metallico di rame, o alluminio, o argento, che funge da conduttore, collegando l'antenna e il chip. Infine vengono accorpate la parte adesiva e l'etichetta; seguono dei test per controllare la conformità agli standard, le varie funzionalità, le performance e l'interoperabilità coi Reader.[4] Come abbiamo già introdotto nel capitolo ***Caratteristiche tecniche di un sistema RFID***, esistono tre tipologie di Tag:

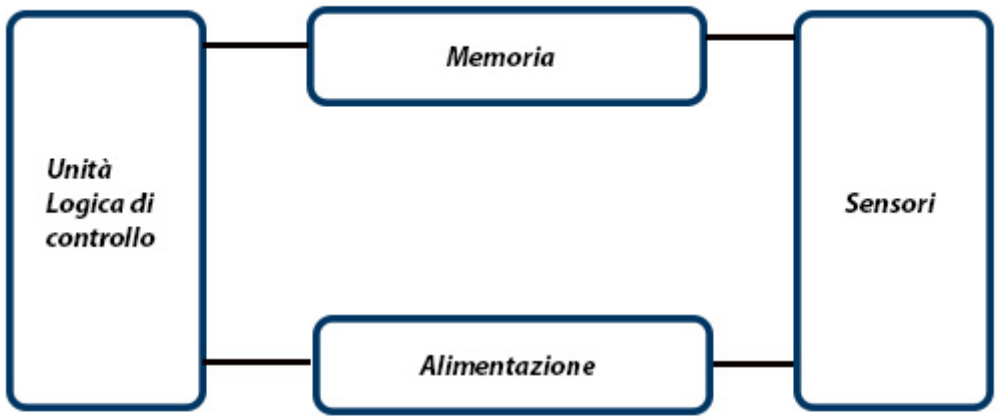
- Tag passivi;
- Tag semipassivi;
- Tag attivi

che verranno approfondite nei seguenti sottoparagrafi.

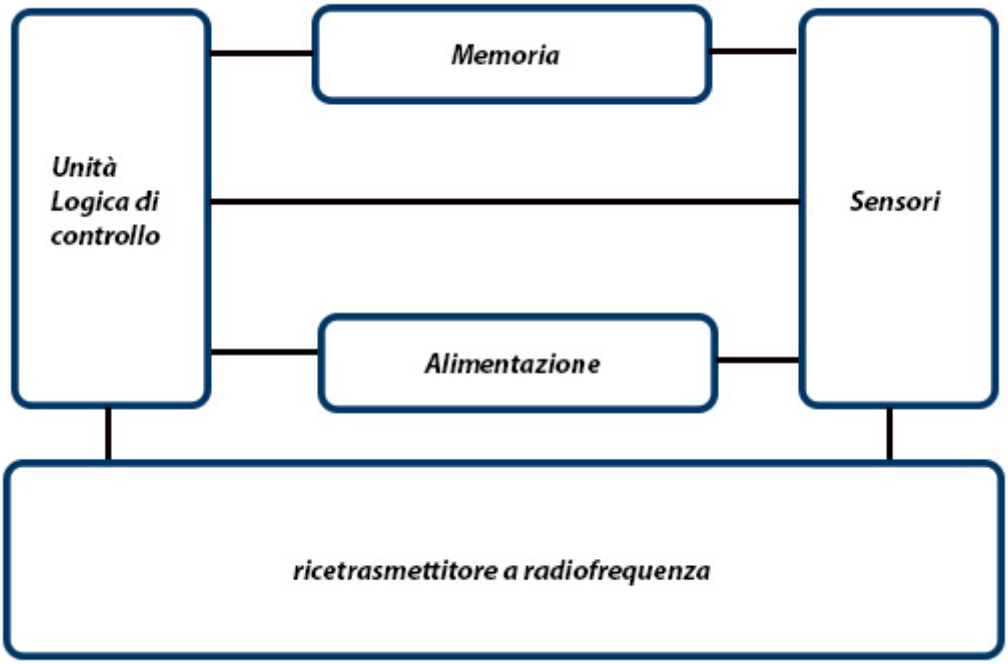
Tag passivi

Lo schema a blocchi mostra la struttura di un Tag passivo. Il blocco di trasmissione e alimentazione si compone di un'antenna in rame che sfrutta la differenza di potenziale che il campo magnetico genera in presenza di un lettore. La corrente viene immagazzinata in un condensatore che funge da alimentatore per il Tag. Questa carica limita lo spettro d'azione di questa tipologia di Tag consentendo loro di svolgere operazioni e calcoli non troppo complessi. I circuiti logici consentono le operazione di lettura, ricezione di informazioni e trasmissione. La memoria di questi Tag è tipicamente non riscrivibile e di piccole dimensioni per via della mancanza di alimentazione continua; solitamente si compone di una memoria EEPROM, non volatile, nella quale è contenuto l'UID.

Tag semipassivi



Lo schema mostra la struttura tipica di un Tag semipassivo. La caratteristica principale sta nell'alimentazione; questi Tag sono dotati di una batteria che viene comunque attivata e caricata quando il dispositivo viene interrogato dal lettore. Questo sistema di alimentazione consente l'integrazione di sensori di temperatura, pressione o movimento e supporta l'uso di memorie più capienti rispetto a quelle dei Tag passivi consentendo operazioni più complesse ai circuiti logici: ad esempio, è possibile avviare comunicazioni cifrate o è possibile trovare di default dei servizi integrati nei circuiti.

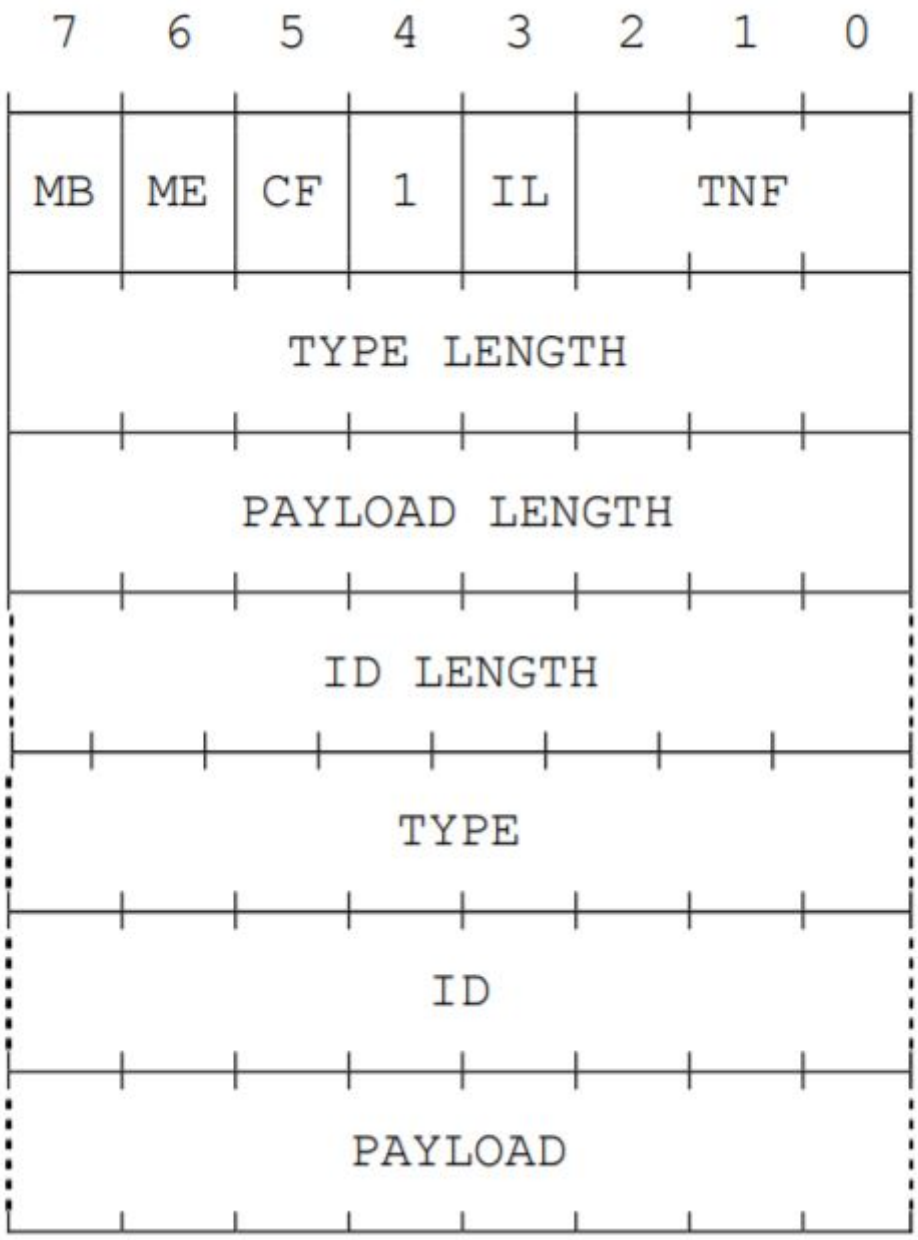


Lo schema mostra e riassume la struttura tipica di un Tag attivo. Si nota subito che lo schema è quasi identico a quello dei Tag semipassivi, infatti essi differiscono per il metodo di ricezione e trasmissione che, in questo caso, avviene per mezzo di un ricetrasmettitore a radiofrequenza. La memoria supporta operazione di lettura e scrittura. La caratteristica principale di questi Tag è che vengono rilevati anche a distanza di qualche km, il che permette comunicazioni a lungo raggio con un costo che si avvicina ai dieci euro.

NDEF

Uno degli aspetti di fondamentale importanza quando si parla dello scambio dei dati in modalità read e peer-to-peer è la struttura con cui i pacchetti vengono scambiati che è definita nell'NFC Forum. Il formato NFC Data Exchange Format, NDEF, specifica e descrive la struttura dei messaggi nelle comunicazioni. NDEF viene utilizzato per archiviare e scambiare informazioni come URI, MIME [7] media type constructs, NFC-specific record type, testo nomale, ecc... I messaggi NDEF possono essere utilizzati anche per scambiare dati tra due dispositivi NFC attivi in modalità "peer-to-peer". L'adesione a questo formato è un'altra caratteristica che genera un linguaggio comune permettendo la condivisione dei dati in modo organizzato e comprensibile: ciò

garantisce interoperabilità. I messaggi NDEF sono il protocollo a livello di trasporto di base per i record NDEF e ogni messaggio può contenere uno o più record. I record NDEF sono strutturati come mostrato nell'immagine sotto.



Riassunto nella tabella seguente:

bit	Campo	Descrizione
7	[MB]	Message Begin
6	[ME]	Message End
5	[CF]	Flag Chunk
4	[SR]	Short Record Bit
3	[IL]	ID Length

bit	Campo	Descrizione
2	[TNF]	Type name format
1	[TNF]	Type name format
0	[TNF]	Type name format
1byte		Type length
1byte - 4 byte		Payload length
1 byte		ID length
1 byte		Record type
1 byte		ID
1 byte		Payload

Descriviamo brevemente la struttura in modo da comprendere meglio il significato di ogni campo:

- **TNF type name format:** il byte 0 viene considerato come intestazione, contiene il TNF un campo a 3 bit che descrive il tipo di record e predispone la struttura al contenuto del record e definisce le aspettative sul contenuto. Leggendo questo campo saremo in grado di classificare il record e ridirigerlo verso la corretta applicazione che deve leggerlo. Type può essere specificato solo nel primo record, i seguenti adotteranno lo stesso type. La tabella seguente descrive i principali valori di type.

Valore	Type	Significato
0x00	Record vuoto	Nessun tipo o id utile è associato a questo record
0x01	Well-known record	Indica che il tipo memorizzato è definito da RTD [8], può essere RTD testo, RTD URI ecc...
0x02	MIME Media Record	Indica che il carico è un blocco intermedio di una sequenza di record NDEF
0x03	URI record	Indica che il campo contiene un percorso assoluto, URI.
0x04	Record esterno	Indica ciò che il campo che segue contiene e specifica il nome
0x05	Record sconosciuto	Indica che il tipo di payload è sconosciuto
0x06	Record invariato	Indica che il record di riferimento è quello relativo al record precedente.

- **IL ID Length:** se impostato a 0, indica che il campo length ID viene omesso dal record.
- **SR Short Record Bit:** è impostato a 1 se il campo Payload lenght è di 1 byte o inferiore.
- **CF Flag Chunk:** indica se è il primo blocco di un record o se è un blocco intermedio.
- **ME Message End:** se 1, indica che questo record segna la fine del messaggio.
- **MB Message Begin:** se 1, indica che questo record segna l'inizio del messaggio. Se MB e ME sono entrambi a 1, significa che il messaggio si compone di un solo record.
- **Type length:** indica la lunghezza in byte del campo type del record. Questo valore è sempre 0 nei type presentati sopra definiti con il campo TNF.
- **Payload length:** indica la lunghezza in byte del payload del record. Se SR = 1, allora questo campo ha lunghezza 1 byte; se SR = 0, allora questo campo avrà lunghezza 4 byte.
- **ID length:** indica la lunghezza in byte del campo ID. Questo campo è presente solo se IL = 1 nell'intestazione.
- **Record type:** descrive il valore type del record. E' importante che questo valore corrisponda al valore presente nei bit TNF.
- **ID:** è presente se il bit IL = 1 e ID Length è presente; indica un ID, utile per identificare i record.
- **Payload:** è il numero di byte descritto in payload length.

Per evidenziare l'interoperabilità fornita anche dallo standard, specifichiamo quelli che rientrano nella tipologia 0x01, ovvero gli well-known record: questi type stanno ad indicare che il tipo è ben noto nel Forum NFC, ovvero che aderisce all'RTD[8]. Tra i più noti c'è il type 0x55 che può essere utilizzato per memorizzare una serie di informazioni utili come numeri di telefono (tel :), indirizzi di siti web, link a percorsi di file, FTP ecc... I record URI sono definiti nel documento "Definizione del tipo di record URI" dal Forum NFC e hanno la seguente struttura:

Nome	Offset	grandezza	descrizione
codice identificativo	0	1 byte	Vedere la tabella seguente
campo URI	1	N bytes	Il resto dell'URI

La seguente tabella mostra i codici identificativi riconosciuti.

valore	protocollo
0x00	nessuna predisposizione, è contenuto un URI

valore	protocollo
0x01	http://www.
0x02	https://www.
0x03	http://
0x04	https://
0x05	tel:
0x06	mailto:
0x07	ftp://anonymous: anonymous @
0x08	ftp:ftp.
0x09	ftps://
0x0A	sftp://
0x0B	smb://
0x0C	nfs://
0x0D	ftp://
0x0E	dav://
0x0F	news:
0x10	telnet://
0x11	imap:
0x12	rtsp://
0x13	urn
0x14	pop
0x15	sip:
0x16	sorsi:
0x17	tftp:
0x18	btspp://
0x19	btl2cap://
0x1A	btgoep://
0x1B	tcpobex://
0x1C	irdaobex://
0x1D	file://

valore	protocollo
0x1E	urn : epc : id :
0x1F	urn : epc : tag :
0x20	urn : epc : pat :
0x21	urn : epc : raw :
0x22	urn : epc :
0x23	urn : nfc:

Il formato NDEF è stato ufficialmente definito nel 2004, però i formati precedenti vengono comunque integrati, ad esempio Android dispone di apposite API. Sempre nella conferenza "How to NFC" viene detto che l'NDEF può essere applicato per i seguenti usi:

- comunicazioni P2P tra dispositivi che seguono le specifiche NFC definite nel Forum;
- il reading di Proprietary NFC Tags;
- per la lettura dei 4 type di Tag definiti nell'NFC Forum:
  1. Topaz TM identificato come Tag type 1.
  2. MIFARE Ultralight TM identificato come Tag type 2.
  3. Felica TM identificato come Tag type 3.
  4. MIFARE Desfire TM identificato come Tag type 4.

Per ulteriori approfondimenti si consiglia di consultare l'NFC Forum.[3]

NTAG 424 DNA Tag Tamper (NT4H2421Tx)

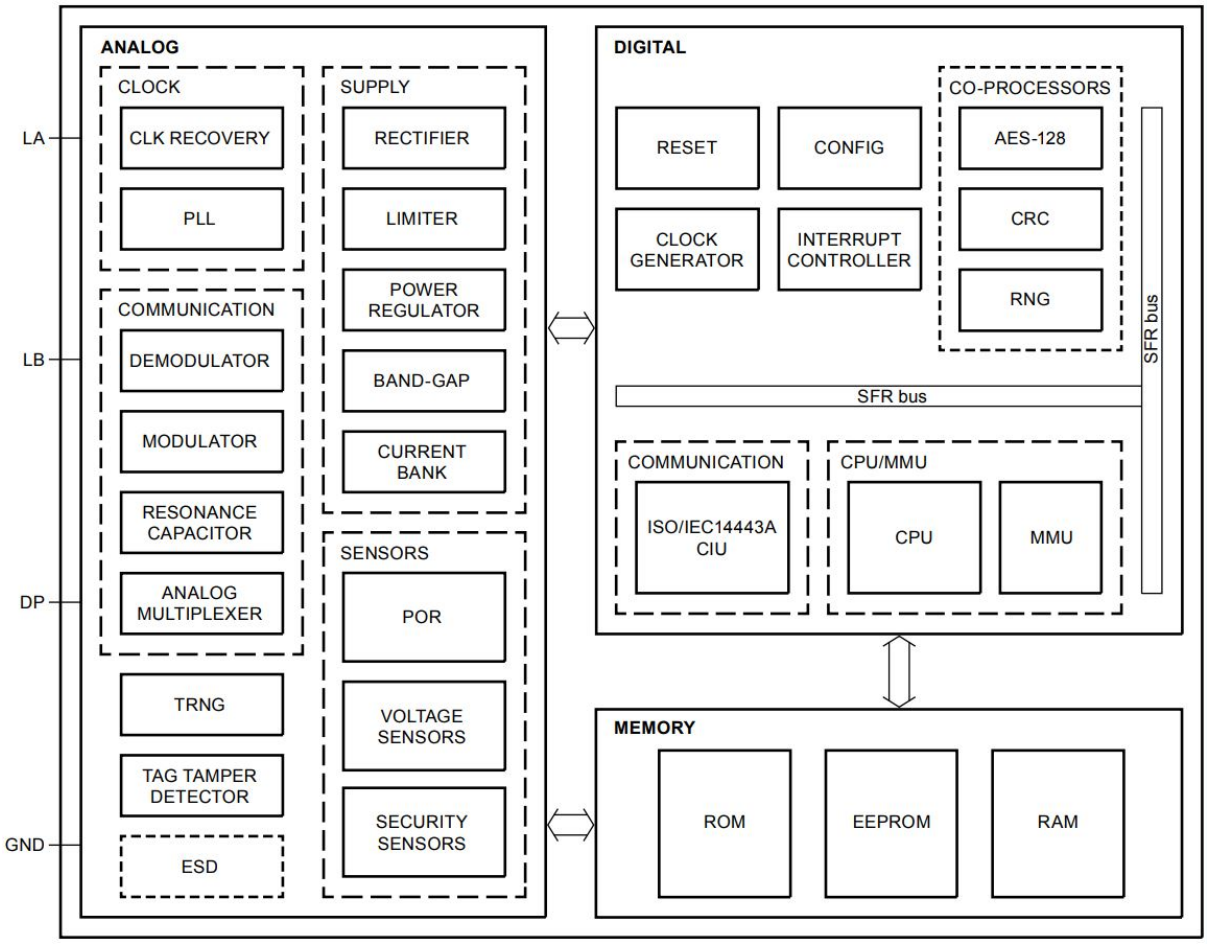
NTAG 424 DNA Tag Tamper (NT4H2421Tx) stabilisce nuovi standard di sicurezza per le applicazioni NFC e IoT, introducendo una nuova generazione di chip DNA NTAG con caratteristiche all'avanguardia per la sicurezza e la protezione della privacy. Integra operazioni crittografiche AES-128 e una nuova funzione di messaggistica Secure Unique NFC Messaging (SUN). NTAG 424 DNA TT è completamente conforme alla specifica IC Tag NFC Forum Type 4 (ID certificazione: 58562), con protocollo di prossimità senza contatto secondo ISO/IEC14443-4, e il file system è basato su ISO/IEC 7816-4 con frame di comando, per garantire la massima interoperabilità all'interno dell'infrastruttura NFC. Le sue prestazioni NFC supportano una maggiore interazione con l’utente e distanze di lettura fino a 10 cm. NTAG 424 DNA TT offre, con la lettura dei Tag, garanzie dei dati sull'autenticità, integrità e persino riservatezza, assicurando

anche la presenza di Tag fisici. Il chip ha una struttura di memoria basata su file di 416 byte totali (conforme a NFC Forum Type 4 Tag e ISO/IEC 7816-4) con un file Capability Container (CC) che specifica l'operazione di NFC Forum Tag, un file NDEF e un file di dati aggiuntivo che permette di proteggere i contenuti sensibili. I diritti di accesso sono configurabili per file, così da supportare diversi casi d'uso, molteplici marchi produttori e fornitori di servizi per soddisfare specifiche di sicurezza e requisiti operativi. Con 5 chiavi AES definite dal cliente, NTAG 424 DNA TT offre avanzate funzionalità crittografiche per CMAC, opzionalmente combinate con dati crittografati, per SUN, autenticazione reciproca (host protetto o autenticazione del lettore) e per assicurare l'accesso al file NDEF e al file di dati extra. NTAG 424 DNA TT contiene funzionalità configurabili come la crittografia facoltativa di parte del File NDEF e modalità di comunicazione completamente crittografata per gestire applicazioni sensibili alla privacy. L'ID casuale opzionale insieme all'UID/dati del chip crittografato che può essere rispecchiato, mirroring, nel file NDEF, consente la conformità con la più recente regolamentazione sulla protezione dei dati dell'utente. Oltre all'implementazione standard AES-128, può offrire anche un file di protocollo alternativo basato su AES per l'autenticazione e la messaggistica sicura utilizzando un Leakage Resilient Primitive, o LRP, un "involucro" attorno alla crittografia AES per migliorare la resistenza agli attacchi del canale laterale. Riassumiamo le caratteristiche e i vantaggi:

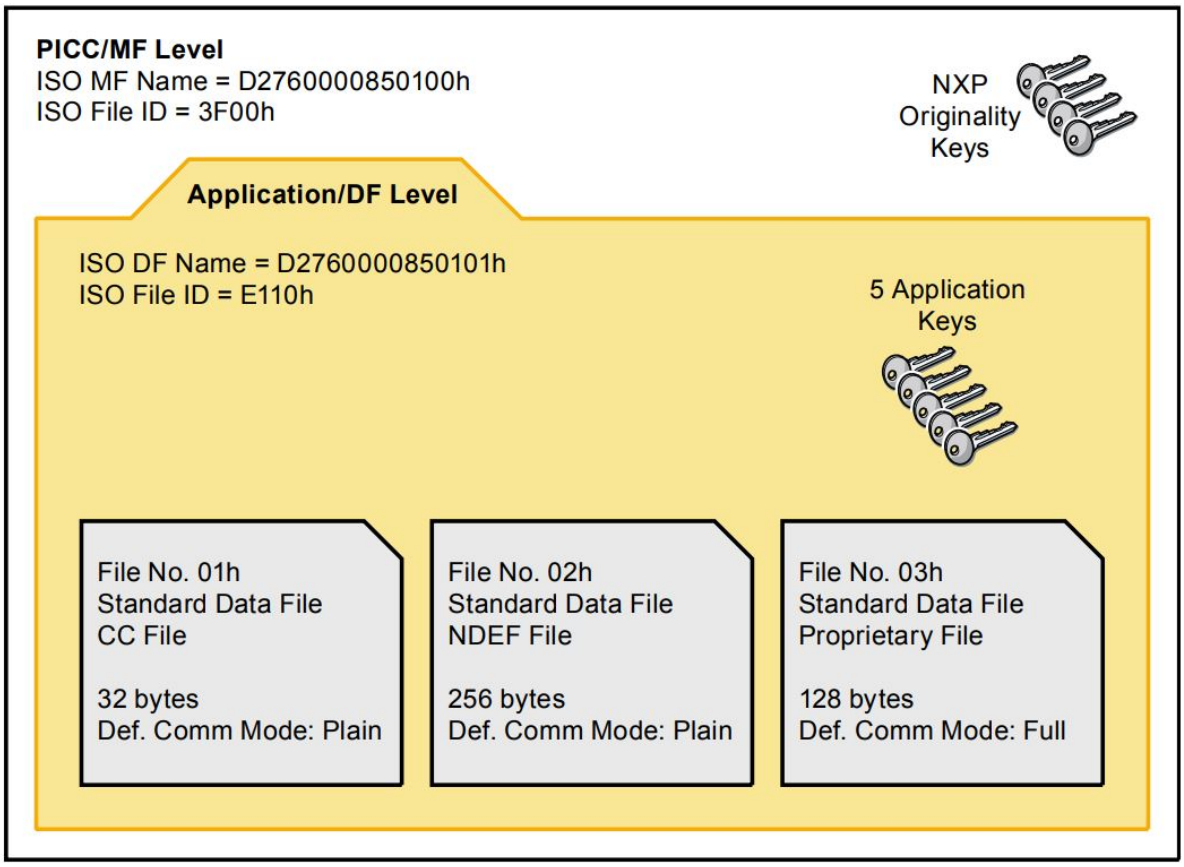
1. Interfaccia e protocollo di comunicazione:
    - pienamente conforme alla specifica tecnica NFC Forum Tag 4 Type;
    - pienamente conforme alle configurazioni della struttura dati NDEF;
    - interfaccia senza contatto conforme a ISO/IEC 14443A-2 / -3 / -4;
    - supporto di frame di comunicazione ISO/IEC 7816-4 per la massima interoperabilità con cellulari e dispositivi tascabili;
    - basso consumo energetico (Hmin) che consente distanze operative fino a 10 cm;
    - supporta velocità di trasferimento dati elevate: 106 kbit/s, 212 kbit/s, 424 kbit/ s e 848 kbit/s;
    - supporta identificatori univoci (UID) a doppia dimensione (7 byte) e ID casuale facoltativamente (RID) secondo ISO 14443-3;
    - supporta frame di comunicazione fino a dimensioni di 128 byte;
    - supporta comandi inclusi in ISO 7816-4.
2. Organizzazione della memoria:
    - 416 byte di memoria utente;
    - conservazione dei dati per 50 anni e durata in scrittura di minimo 200.000 cicli;
    - file system conforme a ISO/IEC 7816-4 con un file di directory predefinito (DF) e un set di file elementari (EF);
    - tre file di dati standard, uno con 32 byte per il file di capacità, uno con 256 byte per l'archiviazione NDEF e uno con 128 byte per i dati protetti;
    - file system conforme a ISO 7816.

3. Sicurezza e privacy:
    - certificazione Common Criteria: EAL4 sia per hardware che per software;
    - messaggio Secure Unique NFC (SUN) con protezione dell'integrità, autenticità e riservatezza;
    - la funzione SUN è abilitata dal Secure Dynamic Messaging (SDM) che si rispecchia come testo (codificato ASCII) nel messaggio NDEF;
    - contatore NFC incrementale, che conta ogni tocco;
    - messaggistica sicura conforme allo standard AES secondo la pubblicazione speciale NIST 800-38A e 800-38;.
    - protezione avanzata opzionale dagli attacchi del canale laterale utilizzando l'operazione AES con wrapping LRP secondo;
    - cinque chiavi AES a 128 bit definite dal cliente, comprese le versioni delle chiavi;
    - ID casuale opzionale per una maggiore privacy.
    - autenticazione reciproca a 3 passaggi.
    - controllo di accesso flessibile configurabile per file (EF);
    - configurazione chiave individuale per Lettura (R) /Scrittura (W) /ReadWrite (RW) /Configurazione;
    - modalità di comunicazione di messaggistica sicura configurabile come:
        - comunicazione semplice;
        - protezione CMAC per la protezione dell'integrità dei messaggi;
        - cifratura completa plus CMAC per la crittografia completa dei dati trasferiti tramite interfaccia contactless;
    - firma di originalità NXP basata su ECC.
    - chiavi di originalità basate su AES che sfruttano l'autenticazione AES con wrapping LRP.

La figura mostra lo schema del Tag da cui è possibile avere una visione completa della struttura fisica.



Nella figura sotto è, invece, possibile vedere la struttura del file system nella memoria in accordo con gli standard ISO/IEC 7816-4. Esistono due livelli MF Level, anche chiamato PICC, e DF Level detto Application all'interno del quale esistono 3 file EFs e 5 chiavi.



NT4H2421Tx memorizza i dati utente in EF (file) di tipi specifici. Tutti i file sono statici, quando vengono creati non possono essere elimintati. È possibile utilizzare un comando per modificare la configurazione dei diritti di accesso. I diritti di accesso ai file sono protetti dalle chiavi dell'applicazione. La tabella riassume la gestione dei file:

EF(file) Type	File type codificato	Nome file	File ID	Grandezza file	Casi d'uso
stabdardData file	00h	01	E103h	32 bytes	CC file
		02h	E104h	256 bytes	messaggi NDEF, supporta SDM e mirroring
		03h	E105h	128 bytes	File proprietari e storage

Il CC file, Capability Container file contiene il file di gestione dei diritti di accesso e di gestione dei dati. Esistono tre tipi di accesso riassunti nella seguente tabella:

Valore	Descrizione
0h...4h	Numero dell'AppKey (Application key)
Eh	Accesso libero
Fh	Accesso vietato o RFU

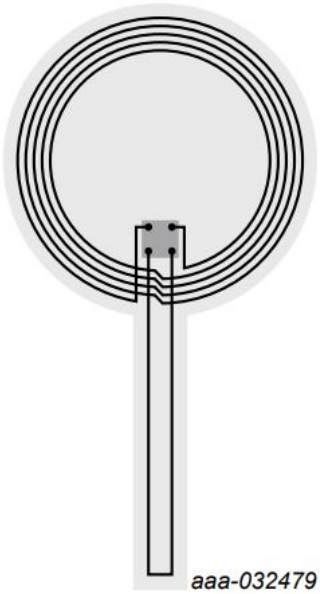
I diritti di accesso di default del Tag vengono mostrati nella seguente tabella:

Nome file	Tipo di File	Lettura	Scrittura	Lettura e scrittura	modifica
01h	StandardData File	Eh	0h	0h	0h
02h	StandardData File	Eh	Eh	Eh	0h
03h	StandardData File	2h	3h	3h	0h

Esistono, inoltre, tre tipi di comunicazione con il chip descritti dalla tabella seguente dal meno protettivo al più protettivo:

Modalità di comunicazione	Rappresentazione bit	Spiegazione
CommMode.Plain	X0b	Nessuna protezione: il messaggio è trasmesso in chiaro.
CommMode.MAC	01b	Protezione MAC per integrità e autenticità
CommMode.Full	11b	Protezione completa per integrità, autenticità e riservatezza, denominata anche "Modalità protezione completa"

Una particolarità di questo Tag è la funzionalità Tag Tamper, è un meccanismo che permette di capire se il chip ha subito delle manomissioni o attacchi di tipo fisico. Una volta applicato il Tag su un oggetto, se qualcuno tenterà di rimuoverlo, andrà ad intaccare questo meccanismo rompendo la linguetta interna al chip. Il Reader attraverso il comando GetTTStatus sarà in grado di sapere se il Tag è stato manomesso; questa funzionalità fornisce un'ulteriore criterio di sicurezza. L'immagine sotto mostra la struttura del Tag Tumper, che risulta essere come una piccola linguetta contenuta nel chip.



Per ulteriori approfondimenti sul Tag si rimanda alla documentazione di riferimento.[5]

Protocolli e standard

ISO/IEC

ISO (the International Organization for Standardization) e IEC (the International Electrotechnical Commission) costituiscono il sistema specializzato per la standardizzazione mondiale. Gli organismi nazionali che sono membri di ISO o IEC partecipano allo sviluppo di standard internazionali attraverso comitati tecnici istituiti dalla rispettiva organizzazione per occuparsi di particolari campi di attività tecnica. I comitati tecnici ISO e IEC collaborano in settori di reciproco interesse. Ai lavori prendono parte anche altre organizzazioni internazionali, governative e non governative, in collegamento con ISO e IEC. Di nostro particolare interesse è l'ISO/IEC 7816 (tutte le parti), una serie di standard che regolamentano le schede a circuiti integrati e l'uso di tali schede per l'interscambio. Queste carte sono carte di identificazione destinate allo scambio di informazioni negoziato tra il mondo esterno e il circuito integrato nella carta. A seguito di uno scambio di dati, la carta fornisce informazioni (risultato del calcolo, dati memorizzati) e/o ne modifica il contenuto (archiviazione dati, memorizzazione eventi). Cinque parti sono specifiche per schede con contatti galvanici e tre di esse specificano interfacce elettriche.

- ISO/IEC 7816-1 specifica le caratteristiche fisiche per le schede con contatti.
- ISO/IEC 7816-2 specifica le dimensioni e la posizione dei contatti.
- ISO/IEC 7816-3 specifica l'interfaccia elettrica e i protocolli di trasmissione per le schede asincrone.
- ISO/IEC 7816-10 specifica l'interfaccia elettrica e la risposta al ripristino per le schede sincrone.
- ISO/IEC 7816-12 specifica l'interfaccia elettrica e le procedure operative per le schede USB.

Tutte le altre parti sono indipendenti dalla tecnologia dell'interfaccia fisica. Si applicano alle schede a cui si accede tramite contatti e / o radiofrequenza.

- ISO/IEC 7816-4 specifica organizzazione, sicurezza e comandi per l'interscambio.
- ISO/IEC 7816-5 specifica la registrazione dei fornitori di applicazioni.
- ISO/IEC 7816-6 specifica gli elementi di dati interindustriali per l'interscambio.
- ISO/IEC 7816-7 specifica i comandi per il linguaggio di query delle carte strutturate.

- ISO/IEC 7816-8 specifica i comandi per le operazioni di sicurezza.
- ISO/IEC 7816-9 specifica i comandi per la gestione delle carte.
- ISO/IEC 7816-11 specifica la verifica personale tramite metodi biometrici.
- ISO/IEC 7816-13 specifica i comandi per la gestione del ciclo di vita delle applicazioni.
- ISO/IEC 7816-15 specifica l'applicazione delle informazioni crittografiche.

Questo documento è destinato ad essere utilizzato in qualsiasi settore di attività. Precisa:

- contenuto delle coppie comando-risposta scambiate sull'interfaccia;
- mezzi per recuperare gli elementi di dati e gli oggetti di dati nella carta;
- strutture e contenuti dei byte storici per descrivere le caratteristiche di funzionamento della carta;
- strutture per applicazioni e dati nella carta, come si vede nell'interfaccia durante l'elaborazione dei comandi;
- metodi di accesso ai file e ai dati nella carta;
- un'architettura di sicurezza che definisce i diritti di accesso ai file e ai dati nella carta;
- mezzi e meccanismi per identificare e indirizzare le applicazioni nella carta;
- metodi per la messaggistica sicura;
- modalità di accesso agli algoritmi elaborati dalla carta. Non descrive questi algoritmi. Non copre l'implementazione interna all'interno della carta o il mondo esterno.

Questo documento è indipendente dalla tecnologia dell'interfaccia fisica. Si applica alle carte a cui si accede con uno o più dei seguenti metodi: contatti, accoppiamento ravvicinato e radiofrequenza. Se la scheda supporta l'uso simultaneo di più di un'interfaccia fisica, la relazione tra ciò che accade su interfacce fisiche differenti è fuori dallo scopo di questo documento.[ISO/IEC 7816]

ISO/IEC 10536 (tutte le parti) specifica l'accesso mediante accoppiamento ravvicinato.

Di nostro interesse sono anche ISO/IEC 14443 (tutte le parti) e ISO/IEC 15693 (tutte le parti) che specificano l'accesso tramite radiofrequenza. Tali carte sono note anche come carte contactless. In particolare la serie di standard ISO/IEC 14443 descrive i parametri per le carte di identificazione o gli oggetti per l'interscambio. Questo documento descrive il polling per le carte di prossimità che entrano nel campo di un dispositivo di accoppiamento di prossimità, il formato dei byte e il framing, il contenuto del comando iniziale di richiesta e risposta alla richiesta, metodi per rilevare e comunicare con una carta di prossimità tra più carte di prossimità (anticollisione) e altre parametri richiesti per inizializzare le comunicazioni tra una tessera di prossimità e un dispositivo di accoppiamento di prossimità. In particolare, ISO/IEC 1444-2 specifica le caratteristiche dei campi da fornire per l'alimentazione e la comunicazione bidirezionale tra dispositivi di accoppiamento di prossimità (PCD) e carte o oggetti di prossimità (PICC). Questo documento non specifica i mezzi per generare campi di accoppiamento, né i mezzi di

conformità alle normative sulle radiazioni elettromagnetiche e sull'esposizione umana, che possono variare a seconda del paese. I protocolli e i comandi utilizzati dai livelli superiori e dalle applicazioni, che vengono utilizzati dopo la fase iniziale, sono descritti in ISO/IEC 14443-3. Nello specifico, questo documento descrive quanto segue:

- polling per tessere o oggetti di prossimità (PICC) che entrano nel campo di un dispositivo di accoppiamento di prossimità (PCD);
- il formato byte, i frame e la temporizzazione utilizzati durante la fase iniziale di comunicazione tra PCD e PICC;
- il contenuto del comando iniziale di richiesta e risposta alla richiesta;
- metodi per rilevare e comunicare con un PICC tra più PICC (anticollisione);
- altri parametri richiesti per inizializzare le comunicazioni tra un PICC e un PCD;
- mezzi opzionali per facilitare e velocizzare la selezione di un PICC tra diversi PICC in base a criteri di applicazione;
- capacità facoltativa di consentire a un dispositivo di alternare le funzioni di un PICC e di un PCD per comunicare rispettivamente con un PCD o un PICC. Un dispositivo che implementa questa capacità è chiamato PXD.

Il protocollo e i comandi utilizzati dai livelli superiori e dalle applicazioni e che vengono utilizzati dopo la fase iniziale sono descritti in ISO/IEC 14443-4. Questo documento è applicabile ai PICC di Tipo A e di Tipo B (come descritto nella ISO/IEC 14443-2), ai PCD (come descritto nella ISO/IEC 14443-2) e ai PXD. ISO/IEC 14443-4 specifica un protocollo di trasmissione a blocchi half-duplex che presenta le esigenze speciali di un ambiente contactless e definisce la sequenza di attivazione e disattivazione del protocollo.
Questo documento è inteso per essere utilizzato insieme ad altre parti della ISO/IEC 14443 ed è applicabile a carte di prossimità o oggetti di Tipo A e Tipo B.[5]

MIFARE

MIFARE è il marchio di proprietà di NXP Semiconductors di una serie di chip a circuito integrato (IC) utilizzati nelle smart card contactless e nelle schede di prossimità. Il nome del marchio copre soluzioni proprietarie basate su vari livelli dello standard per smart card contactless ISO/IEC 14443 Tipo A 13,56 MHz. Utilizza gli standard di crittografia AES e DES/Triple-DES, nonché un algoritmo di crittografia proprietario precedente, Crypto-1. Differisce dagli standard ISO/IEC 14443 per gli standard comunicativi contenuti in ISO/IEC 14443-4 che sono sostituiti da un protocollo proprietario. Per ulteriori informazioni consultare il sito Mifare.

SDM

Secure Dynamic Messaging

Il Secure Dynamic Messaging, SDM, è un servizio integrato in alcuni Tag, una funzionalità che viene messa a disposizione per scambiare dati riservati e protetti nell'integrità senza richiedere un'autenticazione. Prima di entrare nello specifico, vorrei risponde alla domanda: perchè usare SDM? Questo servizio nasce per risolvere esigenze di interoperabilità. L'autenticazione richiede una fase di generazione e scambio delle chiavi che nella letteratura crittografica viene considerata una branca a sé stante, dunque operazioni molto costose e delicate. SDM rilassa i vincoli di sicurezza prevedendo due fasi:

- configurazione dei Tag;
- utilizzo.

Nella prima fase, un provider che conosce le chiavi e che è in grado di autenticarsi con i Tag, abilità l'uso di SDM, configurando le opzioni di codifica interne al chip, e inserisce il contenuto. Nella fase di utilizzo, un qualsiasi lettore non autenticato può leggere il contenuto del Tag. Si immagini, quindi, uno scenario in cui siano presenti molti Tag che necessitano di interconnettersi e scambiarsi i dati, ad esempio un grande magazzino o una smart city: si pensi a quante chiavi dovrebbero essere generate e scambiate per far interagire tutti i componenti. Oltre che in termini economici, vi sarebbe un forte costo anche in termini di operazioni e computazionali, che porterebbero a sovraccarichi di lavoro producendo ritardi; inoltre occorrerebbe poi l'inserimento di un'applicazione propria nel Tag e il funzionamento sarebbe al di fuori di un' operazione di lettura NDEF standard. In termini di sicurezza, è sbagliato pensare che, essendo prevista una fase di autenticazione si sia più protetti dagli attacchi; è vero il contrario, si è più soggetti ad attacchi di tipo Man in the Middle o di sniffing sulle operazioni più sensibili come il passaggio delle chiavi. Quando si utilizza SDM, è necessario tenere conto dei rischi, poiché SDM consente la lettura gratuita del messaggio protetto, ovvero senza alcuna autenticazione del lettore iniziale, chiunque può leggere il messaggio. Ciò significa che anche un potenziale aggressore è in grado di leggere e memorizzare uno o più messaggi e riprodurli in un secondo momento al verificatore. Altre mitigazioni del rischio possono essere applicate per SDM per limitare e attenuare il rischio residuo:

- tracciamento di letture; SDMReadCtr è un'opzione che abilita un contatore di letture; i valori di SDMReadCtr già letti o che vengono letti fuori ordine vengono rifiutati;
- limite di letture in una finestra temporale; in combinazione con la mitigazione precedente, è possibile settare un limite di letture in un arco di tempo, ad esempio si può configurare un Tag per essere letto x volte al giorno.

Approfondiamo gli aspetti tecnici necessari per il provider. Inizialmente è necessaria un autenticazione con il Tag per avere accesso alle proprietà da configurare, questa avviene secondo AES descritto in seguito; è dunque necessario che l'applicazione atta alla configurazione contenga le chiavi per l'autenticazione. L'ultima operazione che tutte le applicazioni di configurazione dei Tag attuano è quella del cambio/modifica delle chiavi. Lo scenario tipico prevede che l'applicazione conosca la chiave di fabbrica, la cui lunghezza varia a seconda del protocollo utilizzato, ma tipicamente formata da tutti zero, che nella fase finale viene cambiata con una chiave che l'applicativo conosce per impedire che enti terzi modifichino le impostazioni e il contenuto del Tag. Esistono due opzioni, SDMMetaRead access right e SDMFileRead access right.

- SDMMetaRead access right: prevede mirroring, ovvero crea una copia del file cifrata e il lettore va a leggere la copia. Viene cifrato con la SDMMetaRead key.
- SDMFileRead access right: non prevede che il file venga cifrato all'interno del chip, viene cifrato al momento della lettura. Utilizza la SDMFileRead key.

E' possibile attivarli entrambi. Nella mia implementazione utilizzerò SDMMetaRead access right. Il caso d'uso tipico è un NDEF che contiene un URI e alcuni metadati, dove SDM consente a questi metadati di essere comunicati con riservatezza e integrità protetta verso un server di backend. Un esempio è : https://ntag.nxp.com/424?e=EF963FF7828658A599F3041510671E88&c=94EED9EE65337086 che mostra come viene applicato l'algoritmo:

Algorithm:

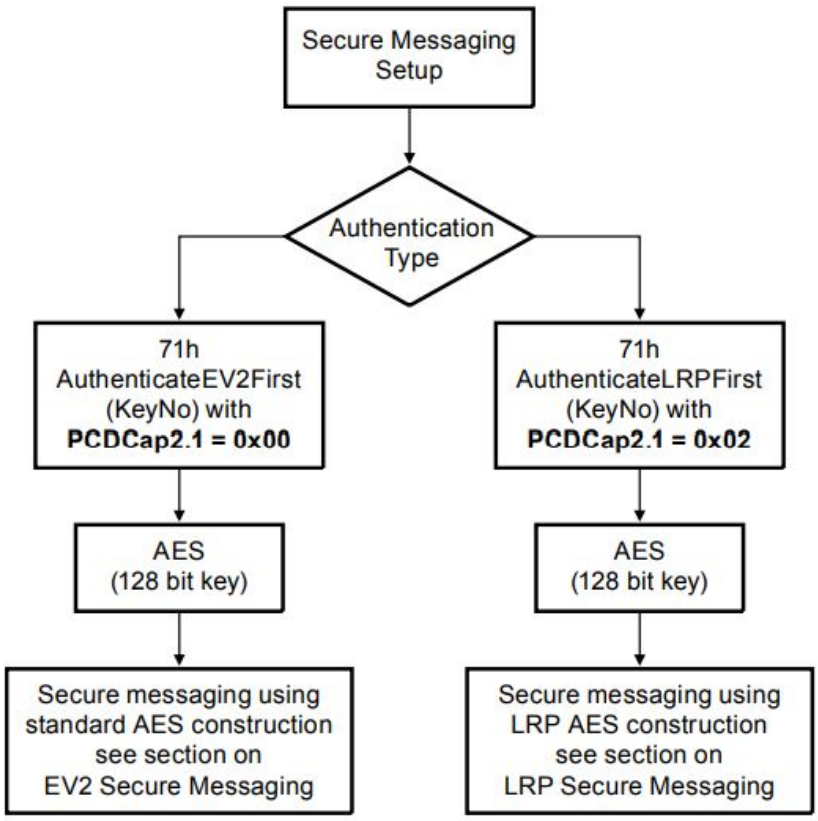
PICCENCData=E(KSDMMetaRead;PICCDataTag[||UID]
[||SDMReadCtr]||RandomPadding)

Il risultato è che un qualsiasi lettore è in grado di effettuare la fase due e non appena avvicinerà il suo device al Tag, vedrà aprirsi su di esso una pagina web relativa al link contenuto nel Tag alla quale passerà delle informazioni cifrate ovvero l'UID e l'SDMReadCounter cifrati, e=EF963FF7828658A599F3041510671E88, e il dato che si desidera comunicare cifrato, c=94EED9EE65337086. L'interoperabilità e la sicurezza sono garantite con questo sistema. Approfondiremo più accuratamente alcuni aspetti tecnici nell'implementazione, visto che esistono diverse sfaccettature e opzioni di SDM.

Secure Messaging

Secure Messaging è una modalità di comunzione. Prima della trasmissione dei dati, è possibile eseguire un'autenticazione reciproca in tre passaggi tra PICC e PCD che si traduce nella generazione di chiavi di sessione utilizzate nella messaggistica sicura, Secure Messaging. Ci sono tre tipi di messaggistica sicura disponibili in NT4H2421Tx, incluso SDM. Uno utilizza AES-128 ed è indicato come modalità AES. L'altro utilizza AES-128 con un wrapper LRP (Leakage Resilient Primitive), noto anche come modalità LRP. La modalità LRP può essere abilitata in modo permanente utilizzando il comando SetConfiguration. Dopo questo passaggio, non è più possibile tornare alla modalità AES. Rispetto alla modalità AES, la modalità LRP ha il vantaggio di fornire forte resistenza agli attacchi side-channel e fault attacks. Serve come sostituto di AES in quanto utilizza solo costruzioni crittografiche standard basate su di esso, senza l'utilizzo di crittografia proprietaria. Pertanto, LRP può essere visto come un'alternativa per ad AES che si basa su di esso e garantisce lo stesso livello di sicurezza, ma ha proprietà migliori. Per migliorare la resistenza agli attacchi side-channel e card attack per la modalità AES, NT4H2421Tx fornisce un limite di tentativi per l'autenticazione non riuscita. Ogni fallimento viene conteggiato nel TotFailCtr. I parametri TotFailCtrLimit TotFailCtrDecr possono essere configurati utilizzando il "Failed authentication counter configuration". Ogni autenticazione non riuscita viene conteggiata internamente nel TotFailCtr counter. Dopo aver raggiunto TotFailCtrLimit, la relativa chiave non può più essere utilizzata per l'autenticazione. Inoltre, dopo aver raggiunto un limite di autenticazioni consecutive non riuscite, NT4H2421Tx inizia a rallentare l'elaborazione di questa funzionalità per ostacolare gli attacchi. Questo viene fatto rifiutando qualsiasi comando di autenticazione con una risposta AUTHENTICATION_DELAY. Il tempo di risposta di un singolo AUTHENTICATION_DELAY è circa il 65% del tempo di risposta massimo specificato. Il tempo di ritardo dell'autenticazione totale aumenta con ogni operazione: tentativo di autenticazione non riuscita; fino a un valore massimo specificato. Solo un'autenticazione riuscita ripristina la piena velocità operativa. La modifica di una chiave AES bloccata è possibile solo con il comando ChangeKey dopo essersi autenticati con AppMasterKey. Se il file AppMasterKey è bloccato, non è possibile alcun recupero. Le autenticazioni AES e LRP vengono avviate da comandi che condividono lo stesso codice. I protocolli di autenticazione sono entrambi basati su AES, ma differiscono per quanto riguarda

l'effettivo
protocollo applicato e la successiva modalità di messaggistica sicura avviata. Una panoramica
delle diverse modalità è riportata nella figura sotto.



Una prima autenticazione può essere eseguita in qualsiasi momento. La Non-First Authentication può essere eseguita solo mentre il Tag è nello stato autenticato ovvero, dopo una prima autenticazione riuscita. La corretta applicazione della prima autenticazione e della Non-First Authentication consente di
associare crittograficamente tutti i messaggi all'interno di una comunicazione grazie al Transaction Identifier stabilito nella prima autenticazione. Ci concentriamo solo sul ramo AES; la AuthenticateEV2First avvia un'autenticazione AES standard, e Secure Messaging. La decisione tra AES e LRP viene presa a seconda del parametro PCDCap2.1 che vedremo in maniera più dettagliata nella sezione Autenticazione.

Transaction Identifier

Al fine di evitare l'interleaving di transazioni da più PCD verso un PICC, il Transaction Identifier (TI) è incluso in ogni MAC, calcolato sui comandi o sulle risposte. Il TI viene generato dal PICC e comunicato al PCD con un file dopo la corretta esecuzione di un comando AuthenticateEV2First. La dimensione è di 4 byte che possono contenere qualsiasi valore.

Command Counter

Un Command Counter è incluso nel calcolo MAC per comandi e risposte, al fine di evitare, ad esempio: raplay attack. Viene anche utilizzato per costruire il vettore di inizializzazione (IV), per cifrare e decifrare. Ogni comando, a parte poche eccezioni, viene conteggiato dal contatore dei comandi CmdCtr che è un numero intero senza segno a 16 bit. Entrambe le parti contano i comandi, quindi l'attuale valore del CmdCtr non viene mai trasmesso. Il CmdCtr viene reimpostato su 0000h in PCD e PICC dopo un'autenticazione AuthenticateEV2First riuscita ed è mantenuto tale fintanto che il PICC rimane autenticato. Nei calcoli crittografici, il CmdCtr è rappresentato in formato LSB. Le successive autenticazioni che utilizzano AuthenticateEV2NonFirst non influenzano il CmdCtr. Il CmdCtr viene aumentato tra il comando e la risposta, per tutte le modalità di comunicazione. Quando un MAC su un comando viene calcolato sul lato PCD che include CmdCtr, utilizza l'attuale CmdCtr. Il CmdCtr viene successivamente incrementato di 1. Sul lato PICC, un MAC aggiunto ai comandi ricevuti viene verificato utilizzando il valore corrente di CmdCtr. Se il MAC corrisponde, CmdCtr viene incrementato di 1 dopo la corretta ricezione del comando, e prima di inviare una risposta.

MAC Calculation

I MAC vengono calcolati secondo lo standard CMAC descritto da NIST in NIST Special Publication 800-38B, Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication. Il padding è applicato secondo lo standard. Il MAC utilizzato in NT4H2421Tx viene troncato utilizzando solo gli 8 byte pari dei 16 byte di output come descritto nella pubblicazione sopracitata.

Cifratura

La cifratura e la decifratura vengono effettuate utilizzando AES-128 in base alla modalità CBC di
NIST SP800-38A. Il riempimento viene applicato secondo Padding Method 2 di ISO/IEC 9797-1, ovvero mediante aggiunta di 80h in seguito, se richiesto, al byte zero fino a quando non si ottiene una stringa di lunghezza multipla di 16 byte. L'unica eccezione è durante l'autenticazione stessa
(AuthenticateEV2First e AuthenticateEV2NonFirst), dove non viene applicato alcun riempimento.
La notazione E (chiave, messaggio) viene utilizzata per denotare la crittografia e D(chiave, messaggio) per decrittazione.

Initialization Vector for Encryption IV

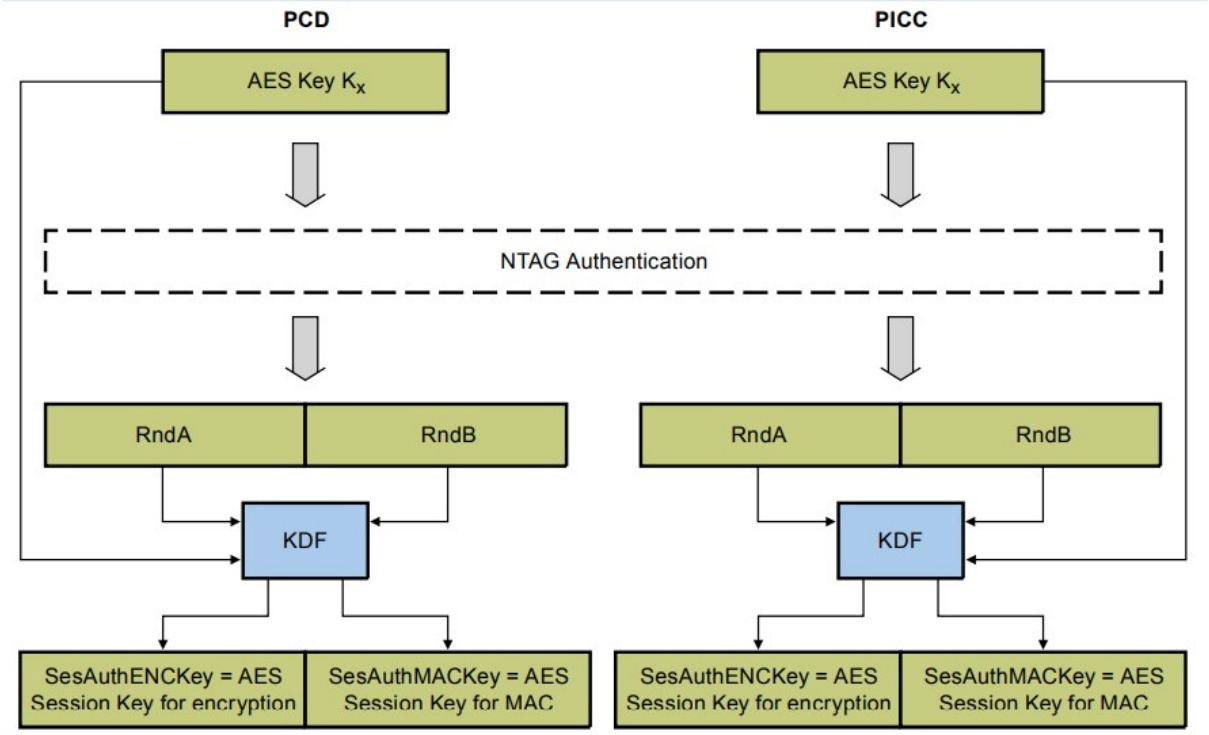
Quando la crittografia viene applicata ai dati inviati tra il PCD e il PICC, il file Initialization Vector (IV) viene costruito crittografando con SesAuthENCKey secondo la modalità ECB del NIST SP800-38A con la concatenazione di:

- un'etichetta di 2 byte, che distingue lo scopo dell'IV: A55Ah per i comandi e 5AA5h per le risposte;
- Transaction Identifier TI;
- Command Counter CmdCtr (LSB prima);
- Riempimento di zeri;

Ciò si traduce nei seguenti IV:
IV per CmdData = E (SesAuthENCKey; A5h || 5Ah || TI || CmdCtr || 0000000000000000h)
IV per RespData = E (SesAuthENCKey; 5Ah || A5h || TI || CmdCtr || 0000000000000000h)
Quando viene calcolata una cifratura o una decifratura, i CmdCtr da utilizzare nell'IV sono i valori correnti. Nota che questo significa che se CmdCtr = n prima della ricezione di un comando, dopo la convalida del comando CmdCtr = n + 1 e quel valore sarà utilizzato nel IV per la crittografia della risposta. Per la crittografia durante l'autenticazione (sia AuthenticateEV2First che
AuthenticateEV2NonFirst), l'IV sarà 128 bit a 0.

Session Key Generation

Alla fine di un'autenticazione valida con AuthenticateEV2First o AuthenticateEV2NonFirst, sia il PICC che il PCD generano due chiavi di sessione per Secure Messaging, come mostrato nella Figura sotto.



- SesAuthMACKey per il MAC dei messaggi;
- SesAuthENCKey per la cifratura e la decifratura dei messaggi;

Si noti che questi identificatori vengono utilizzati anche nel contesto del protocollo LRP, sebbene l'effettivo calcolo delle chiavi di sessione è diverso. La Pseudo Random Function PRF(key; message) applicata durante la generazione della chiave è l'algoritmo CMAC descritto nella pubblicazione NIST 800-38B. La chiave di derivazione è la chiave Kx, che è stata applicata durante l'autenticazione. I dati di input sono costruiti utilizzando i seguenti campi:

- un'etichetta di 2 byte, che distingue lo scopo della chiave: 5AA5h per MACing e A55Ah per crittografata;
- un contatore a 2 byte, fissato a 0001h poiché vengono generate solo chiavi a 128 bit;
- una lunghezza di 2 byte, fissata a 0080h poiché vengono generate solo chiavi a 128 bit;
- un contesto a 26 byte, costruito utilizzando i due numeri casuali scambiati, RndA e RndB.

I vettori di sessione di input a 32 byte SVx sono derivati come segue:
SV1 = A5h || 5Ah || 00h || 01h || 00h || 80h || RndA [15..14] ||

(RndA [13..8] # RndB [15..10]) || RndB [9..0] || RndA [7..0]

SV2 = 5Ah || A5h || 00h || 01h || 00h || 80h || RndA [15..14] ||

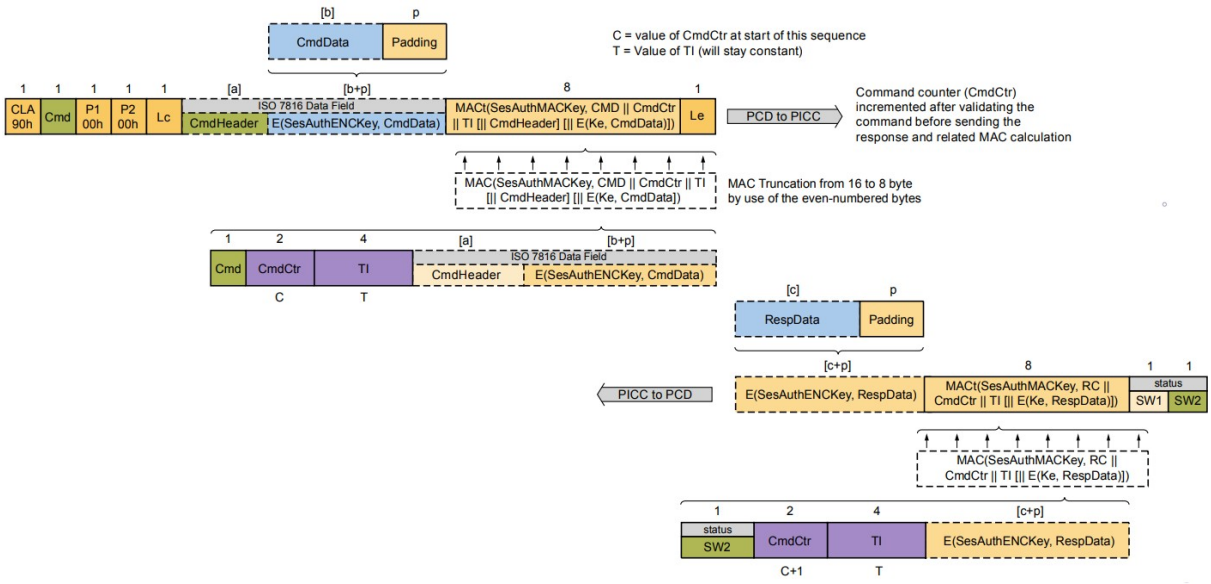
(RndA [13..8] # RndB [15..10]) || RndB [9..0] || RndA [7..0]
con # che è l'operatore XOR.

Quindi, le chiavi di sessione a 16 byte vengono costruite come:

- SesAuthENCKey = PRF (Kx, SV1)
- SesAuthMACKey = PRF (Kx, SV2)

Full Communication Mode

Secure Messaging applica la crittografia (CommMode.Full) a molti dei comandi utilizzabili per interagire con NTAG 424DNA TT. Nel caso in cui si debba applicare CommMode.Full, vale quanto segue: la cifratura/decifratura viene calcolata utilizzando la chiave di sessione corrente SesAuthENCKey. Dopo la crittografia, i frame di comando e risposta vengono gestiti come MAC. Ciò significa che viene calcolato un MAC e aggiunto per la trasmissione, utilizzando la chiave di sessione corrente SesAuthMACKey. Questo viene fatto esattamente come per MAC ovvero, sostituendo il CmdData o RespData con il campo crittografato: E(SesAuthENCKey; CmdData) o E(SesAuthENCKey; RespData). Il calcolo completo è illustrato nella figura sotto.



Vista la complessità del tema si consiglia di consultare la documentazione[5].

DES, 3DES, AES Evoluzione della crittografia a chiave simmetrica

Cifrari

Al fine di comprendere la crittografia a chiave simmetrica, dobbiamo andare alle radici di questo concetto. Basti pensare che fino a pochi decenni fa i procedimenti di cifratura e decifrazione venivano fatti con carta e penna. Un cifrario è un algoritmo che definisce la procedura utilizzata per rendere oscuro, ovvero semanticamente non leggibile, il testo di un messaggio, anche detto plain text, ma che identifica anche la serie di passaggi volti a rendere chiaro un messaggio oscurato con quello stesso algoritmo. Il cifrario è ciò che ambo le parti, mittente e destinatario, devono conoscere per poter comunicare nella stessa "lingua". I più antichi cifrari risalgono ai tempi dei Greci e Romani, per completezza ne cito alcuni con i relativi riferimenti.

- Cifrario di Cesare
- Cifrario di Alberti
- Cifrario di Vigenère
- Cifrari a griglia
- Sistema Enigma

Questi cifrari, oggi, vengono definiti "semplici" perchè basano la loro logica su algoritmi facili da violare o manipolare, e ciò li rende molto sensibili ad ogni tipo di attacco. Da questi cifrari si possono, però, dedurre le due proprietà fondamentali che un cifrario deve avere:

- *diffusione:* una caratteristica che riguarda la dissipazione della struttura statistica del plain text sul cyphertext;
- *confusione:* una caratteristica che intende rendere la relazione tra pain text e ciphertext più complessa e legata possibile.

Le definizione riportate furono introdotte da Cloude Shannon nell'articolo A Mathematical Theory of Cryptography scritto nel 1945 e riprese anche in Communication Theory of Secrecy Systems, del 1949, che parla dell'inviolabilità del cifrario perfetto One-Time Pad.

Crittografia simmetrica

Con "crittografia simmetrica", altresì detta "crittografia a chiave privata", intendiamo un metodo per cifrare e decifrare il plain text con una sola chiave. Si suppone che il mittente e il destinatario siano già in possesso della stessa chiave, essendo impossibile con questa tecnica uno scambio di chiavi in maniera sicura. Solitamente questa operazione avviene con algoritmi a chiave asimmetrica o pubblica; in alternativa, come in passato, la chiave viene condivisa in maniera fisica tra i due diretti interessati. Metaforicamente parlando, introduciamo anche il concetto di chiave e riportiamo un esempio di crittografia simmetrica; immaginiamo di avere un forziere e di riporvi all'interno il nostro messaggio da recapitare al destinatario senza che nessuno sia in grado di aprirlo. Per farlo, inseriamo il messaggio nel forziere e lo chiudiamo con un lucchetto, che sia a chiave o a combinazione; il concetto di chiave rappresenta proprio quell'elemento univoco capace di aprire e chiudere la serratura. La simmetria è data dal fatto che la chiave o la combinazione è in possesso del mittente e del destinatario, ovvero si assume che il mittente prima di inviare il forziere abbia già incontrato il destinatario e gli abbia dato una copia della chiave o gli abbia detto la combinazione per aprire la serratura. Nei capitoli seguenti riprenderemo questi aspetti con un approccio più matematico.

Il cifrario simmetrico standard: la storia

Nel 1972 la National Bureau of Standards, NBS, attuale NIST, National Institute for Security and Technology, si rese conto della necessità di proteggere le comunicazioni commerciali o tra privati. All'epoca erano già presenti software capaci di proteggere le comunicazioni tra privati, ma non erano regolamentati, spesso erano incompatibili tra loro nelle varie comunicazioni. Nel 1973, al fine di definire uno standard, venne pubblicato un bando in cui veniva richiesto che:

- la sicurezza dipendesse dalla segretezza della chiave e non dai processi crittografici;
- l'algoritmo dovesse essere facilmente eseguibile dall'hardware.

Il primo bando non riscosse proposte significative, perciò venne riproposto un altro bando nel 1974. IBM, International Business Machines Corporation rispose al bando proponendo DEA, Data Encryption Algorithm una versione derivante dal software Lucifer[9]. In seguito ad un confronto con NSA, Nation Security Agency, venne certificato il DES, Data Encryption Standard. Definito e pubblicato come standard dalla

Federal Information Processing Standard, FIPS, nel 1977, fu riconfermato nel 1988 e nel 1993. Nel luglio del 1998 venne violata per la prima volta una chiave, in 56 ore, dal DES cracker [10]; l'anno successivo DES cracker [10] e distributed.net in congiunzione violarono una chiave DES in 22 ore e 15 minuti. Il 25 ottobre del 1999 FIPS confermò per la quarta volta DES raccomandando l'uso di Triple DES che approfondiremo brevemente in seguito. Il 26 novembre del 2001 venne pubblicato l'Advanced Encryption Standard, AES, come FIPS 197.

DES

Come abbiamo visto DES è un pilastro della crittografia a chiave simmetrica. Approfondiamo la sua struttura al fine di comprenderne meglio il funzionamento. Tipicamente DES ha una chiave di lunghezza k = 56 bit e un blocco lunghezza n = 64 bit. Il messaggio viene suddiviso in blocchi che vengono cifrati e decifrati in maniera indipendente gli uni dagli altri. Le operazioni di cifratura e decifrazione consistono in r fasi successive, dette round di quella che viene chiamata "Feistel network", ideata dal tedesco Horst Feistelt. DES ha r = 16.

Algoritmo DES:

function $DES_K(M)$ // $|K| = 56$ and $|M| = 64$
$(K_1,...,K_{16}) \leftarrow$ KeySchedule(K) // $|K_i| = 48$ for $1 \leq i \leq 16$
$M \leftarrow IP(M)$
Parse M as $L_0 \,||\, R_0$ // $|L_0| = |R_0| = 32$
for r = 1 to 16 **do**
  $L_r \leftarrow R_{r-1}$; $R_r \leftarrow f(K_r, R_{r-1}) \oplus L_{r-1}$
$C \leftarrow IP^{-1}(L_{16} || R_{16})$
return C

IP								**IP^{-1}**							
58	50	42	34	26	18	10	2	40	8	48	16	56	24	64	32
60	52	44	36	28	20	12	4	39	7	47	15	55	23	63	31
62	54	46	38	30	22	14	6	38	6	46	14	54	22	62	30
64	56	48	40	32	24	16	8	37	5	45	13	53	21	61	29
57	49	41	33	25	17	9	1	36	4	44	12	52	20	60	28
59	51	43	35	27	19	11	3	35	3	43	11	51	19	59	27

IP								IP^{-1}							
61	53	45	37	29	21	13	5	34	2	42	10	50	18	58	26
63	55	47	39	31	23	15	7	33	1	41	9	49	17	57	25

La tabella descrive la permutazione iniziale IP di DES e la sua inversa.

L'algoritmo DES illustrato prende in ingresso una chiave K a 56 bit e un plain text a 64 bit M. Il KeySchedule produce dalla chiave K a 56 bit una sequenza di 16 sottochiavi, una per ciascuno dei round. Ogni sottochiave è lunga 48 bit. La permutazione iniziale IP permuta semplicemente i bit di M, come descritto dalla tabella sopra. Si noti che il bit 1 dell'uscita è il bit 58 dell'ingresso, il bit 2 dell'uscita è il bit 50 dell'ingresso...il bit 64 dell'uscita è il bit 7 dell'ingresso. Abbiamo così definito le permutazioni iniziali. Inoltre, in questa fase è evidente che la chiave non influenza il processo di permutazione, questo ciclo sembra non interessare il processo crittografico dell'algoritmo. Il plain text viene immesso in un ciclo che opera in 16 round, ognuno dei quali prende in input 64bit, visti come un composto di 32 bit di sinistra e 32 bit di destra; questi vengono influenzati dalla K_r sottochiave, producendo 64bit di output. Gli input vengono rispettivamente arrotondati da $L_{r-1} \| R_{r-1}$, l'uscita dal singolo round è L_r e R_r calcolati in funzione di $L_{r-1} \| R_{r-1}$ calcolati da f dipendente dalla K_r esima sottochiave associata al r-esimo giro. L'importanza di utilizzare una struttura circolare è che garantisce reversibilità, essenziale per far sì che DES_k sia una permutazione per ogni chiave K. Dati L_r, R_r e K_r, è possibile recuperare $L_{r-1} \| R_{r-1}$ con $R_{r-1} \leftarrow L_r$ and $L_{r-1} \leftarrow f(K - r, L_r) \oplus R_r$. Eseguiti i 16 round, si applica IP^{-1} all'output a 64bit e si ottiene il cyphertext.

round Fiestel Function Algorithm:

function f(J, R) // |J| = 48 and |R| = 32
$R \leftarrow E(R)$; $R \leftarrow R \oplus J$
Parse R as $R_1 \| R_2 \| R_3 \| R_4 \| R_5 \| R_6 \| R_7 \| R_8$ // $|R_i| = 6$ for $1 \leq i \leq 8$
for i = 1,..., 8 **do**
$R_i \leftarrow S_i(R_i)$ // Each S-box returns 4 bits
$R \leftarrow R_1 \| R_2 \| R_3 \| R_4 \| R_5 \| R_6 \| R_7 \| R_8$ // |R| = 32 bits
$R \leftarrow P(R)$
return R

E						P			
32	1	2	3	4	5	16	7	20	21
4	5	6	7	8	9	29	12	28	27
8	9	10	11	12	13	1	15	23	26

E						P			
12	13	14	15	16	17	5	18	31	10
16	17	18	19	20	21	2	8	24	14
20	21	22	23	24	25	32	27	3	9
24	25	26	27	28	29	19	13	30	6
28	29	30	31	32	1	22	11	4	25

La tabella descrive la funzione di espansione E e la permutazione finale P della funzione f di DES.

Una sequenza di round Fiestel va vista come una funzione f(. , .) che richiede in input una sottochiave J di 48 bit e R a 32 bit. R viene prima espanso dalla funzione E descritta nella tabella in 48 bit. Il bit 1 dell'uscita è il bit 32 del file ingresso, il bit 2 dell'uscita è il bit 1 dell'ingresso...il bit 48 dell'uscita è il bit 1 dell'ingresso. E' possibile notare che la funzione E è molto strutturata, oltre a 1 e 32 che sono scambiati, risulta essere sequenziale. Viene poi eseguito lo XOR tra l'output di E e J che produce un risultato a 48 bit che nell'algoritmo sopra citato viene denotato con R. I 48 bit vengono suddivisi in 8 blocchi da 6 bit, ad ognuno viene applicata la funzione S che prende 6 bit e ne ritorna 4. Il risultato ottenuto è una compressione da 48 bit a 32 bit permutati secondo la funzione P descritta nella tabella. Le seguenti tabelle mostrano i risultati delle S-box.

S1		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
1	0	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
1	1	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S2		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
0	1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
1	0	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
1	1	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S3		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
0	1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
1	0	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	1	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S4		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
0	1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
1	0	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
1	1	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S5		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
0	1	14	11	2	12	4	7	13	1	5	0	15	10	2	9	8	6
1	0	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
1	1	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S6		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
0	1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
1	0	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
1	1	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S7		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
0	1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	0	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
1	1	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S8		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
0	1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
1	0	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
1	1	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Le tabelle sopra elencate rappresentano le DES S-box.

Ogni S-box prevede un ingresso a 6 bit che fa da ingresso alle singole sottofunzioni di pari indice. Ogni sottofunzione S_i sarà definita dividendo $B_j = b_1b_2b_3b_4b_5b_6$ in due gruppi come b_1b_2 nelle righe e $b_3b_4b_5b_6$ nelle colonne, andando a delineare delle tabelle che definiscono una coppia di numeri x,y tali per cui

$$0 \leq x \leq 3$$

e

$$0 \leq y \leq 15$$

Le S-box sono il cuore dell'algoritmo e, per progettarle, sono stati fatti molti sforzi al fine di raggiungere vari obiettivi di sicurezza e resistenza a determinati attacchi. La chiave K è formata da 56 bit ma le chiavi K_r sono da 48 bit, e per ottenerle si eseguono i seguenti passaggi:

Algorithm KeySchedule (k): // |K| = 56
K ← PC-1(K)
Parse K as $C_0 \,||\, D_0$
for r = 1,..., 16 **do**
  if r ∈ {1, 2, 9, 16} **then** j ← 1 else j ← 2 **fi**
  $C_r \leftarrow \text{leftshift}_j (C_{r-1})$; $D_r \leftarrow \text{leftshift}_j (D_{r-1})$
  $K_r \leftarrow \text{PC-2}(C_r \,||\, D_r)$
return(K_1,...,K_{16})

Inizialmente viene fatta la permutazione PC-1 mostrata nella tabella sotto. In seguito, si divide K in due metà da 28 bit $C_0||D_0$. L'algoritmo fa 16 giri, all'r-esimo giro prende in input $C_{r-1}||D_{r-1}$ e applica la funzione PC-2 che estrae 48 bit definendo così K_r, la sottochiave del r-esimo giro. Per ottenere C_r, si prendono i bit di C_{r-1} e si ruotano nelle posizione j di sinistra. D_r è ottenuto alla stessa maniera da D_{r-1}. j è 1 o 2 a seconda di r . PC-1 contiene 56 numeri interi, compresi tra 1 e 64, esclusi i multipli di 8. Sia K una stringa di 56 bit, diremo che K=K[1]...K[56] in input, avremo in output dalla funzione una stringa a 56 bit L tale che L=L[1]...L[56] calcolata come segue. Data i, i-esima entrata in tabella, dove $1 \leq i \leq 56$. Si determina:

$$a = 8q + r$$

Dove r è $1 \leq r \leq 7$. Quindi L[i]=K[a-q]. Esempio: determiniamo il primo bit L[1] dell'uscita della funzione sull'ingresso K. Osserviamo la prima voce in tabella, è 57. Dividiamo per 8 ottenendo 57=8*(7) + 1. Avremo che L[1]=K[57-7]=K ovvero il primo bit dell'uscita è il 50esimo bit in ingresso. PC-2 viene letto come la solita mappa che prende in input 56 bit e ha un' uscita a 48 bit: il bit 1 dell'uscita è il bit 14 dell'input,il bit 2 dell'uscita è il bit 17 dell'ingresso...il bit 56 dell'uscita è il bit 32 dell'ingresso. Giunti a questo punto, siamo a conoscenza di come lavori DES.

PC-1							**PC-2**					
57	49	41	33	25	17	9	14	17	11	24	1	5
1	58	50	42	34	26	18	3	28	15	6	21	10
10	2	59	51	43	35	27	23	19	12	4	26	8
19	11	3	60	52	44	36	16	7	27	20	13	2

PC-1							PC-2					
63	55	47	39	31	23	15	41	52	31	37	47	55
7	62	54	46	38	30	22	30	40	51	45	33	48
14	6	61	53	45	37	29	44	49	39	56	34	53
21	13	5	28	20	12	4	46	42	50	36	29	32

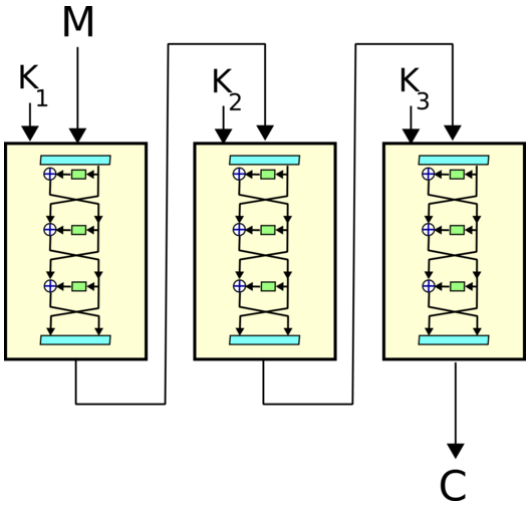
Le tabelle mostrano le funzioni PC-1, PC-2 usate da DES nell'algoritmo KeySchedule.

Triple DES, 3DES

Introduciamo, per completezza anche il triple DES, detto pure 3DES. Possiamo riassumere questo algoritmo con la funzione:

$$cyphertext = DES(K3, DES(K2, DES(K1, M))) = DES(K3, DES^{-1}((K2, DES(K1, M)))$$

Dove M è il messaggio e k1, k2, k3 sono le chiavi. E' usuale, per motivi di interoperabilità, trovare la versione che nel passaggio centrale esegua una decriptazione, in questo caso possiamo eseguire tutto con una sola chiave K tale che K1 = K2 = K3 = K. E' evidente che l'algoritmo prevede una tripla criptazione del messaggio, mediante DES. Si noti che 3DES è più sicuro di DES in quanto DES non è un gruppo [11]. Esistono più varianti e per ulteriori approfondimenti si consiglia di consultare Introduction to Modern Cryptography capitolo 3.4.



L'immagine mostra in maniera sintetica e schematica il 3DES, dove ogni blocco giallo rappresenta DES.

AES

Con l'evoluzione della tecnologia, le chiavi DES risultavano troppo corte e facilmente calcolabili. Nel 1998, NIST annunciò un nuovo bando per definire un successore di DES. Vi furono quindici algoritmi in risposta al bando; al secondo round rimasero in cinque. Nell'estate del 2001 NIST scelse un algoritmo chiamato Rijndeal, nel novembre venne pubblicato come FIPS 187. Il progetto, ideato Joan Daemen e Vincent Rijmen è il discendente di un algoritmo chiamato Square[12]. Nello standard, questo algoritmo prese il nome di AES, Advanced Encryption Standard: per i puristi, AES è un caso speciale di Rijndeal legato alla lunghezza del blocco in input. AES prende in ingresso un blocco n di 128 bit e una chiave k tipicamente di 128, 192, 256 bit. Gli standard variano leggermente a seconda della lunghezza della chiave: ci concentriamo su AES128 che è quello utilizzato dal nostro Tag in fase di autenticazione.

Algorithm AES 128:

function $AES_K(M)$
$(K_0,...,K_{10}) \leftarrow$ expand(K)
$s \leftarrow M \oplus K_0$
for r = 1 to 10 **do**
  $s \leftarrow S(s)$
  $s \leftarrow$ shift-rows(s)
  if $r \leq 9$ **then** $s \leftarrow$ mix-cols(s) **fi**
  $s \leftarrow s \oplus K_r$
endfor
return s

L'algoritmo sopra descritto prevede che | K | = 128 e | M | = | C | = 128. Suddividiamo l'algoritmo nelle quattro operazioni più significative: expand, S, shift-row e mix-cols al fine di analizzarne meglio il comportamento.

Expand prende in ingresso i 128 bit della chiave e produce un vettore di undici chiavi $(K_0,...,K_{10})$. Le altre tre sono funzioni biiettive utilizzate per mappare i bit. s viene chiamato stato; in fase di inizializzazione viene definito uno stato in M, C il cyphertext è lo stato finale di M a seguito di una codifica. Diremo round tutto ciò che accade ad ogni iterazione del ciclo for: in AES abbiamo 9 round identici che utilizzano la $K_{i\text{-esima}}$ chiave, il decimo giro non effettua la mix-cols. Le operazioni di S e mix-cols sfruttano le proprietà algebriche di un campo finito, noto come Galois Field. In particolare GF(28) composto da 256 punti, in cui è possibile trovare l'inverso di un ognuno di essi escluso

lo zero. In questo campo la somma equivale a uno XOR bit a bit, mentre la moltiplicazione a un and. Diremo che inv (a) = a^{-1}, da cui deriva la mappatura S : $\{0, 1\}^8 \rightarrow \{0, 1\}^8$. Per calcolare S(a) sostituiamo a con inv(a), numeriamo i valori di a = $a_7a_6a_5a_4a_3a_2a_1a_0$ e ci aspettiamo come risultato un a' dove a' = $a'_7a'_6a'_5a'_4a'_3a'_2a'_1a'_0$.

$$\begin{pmatrix} a'_7 \\ a'_6 \\ a'_5 \\ a'_4 \\ a'_3 \\ a'_2 \\ a'_1 \\ a'_0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} a_7 \\ a_6 \\ a_5 \\ a_4 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

Nella sua complessità la mappatura di S è mostrata qui di seguito ed elenca tutti i valori da S(0) a S(255).

63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	8a
51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
cd	0c	13	ec	5f	97	44	17	c4	a7	73	3d	64	5d	19	73
60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Avremo che S(00)=63,S(01)=7c,...,S(ff)=16. Ora che abbiamo definito S, estendiamo il suo dominio a $\{\{0, 1\}^8\}^+$. Vale a dire, data una stringa di m byte A=A[1]....A[m], allora applicare S(A) equivale a fare S(A[1])...S(A[m]), ovvero eseguire S per byte. La prima mappa di S(s) prevede che venga preso lo stato di 16 byte e si applichi la tabella di ricerca a 8 bit su ciascuno dei sui byte per ottenere lo stato modificato di s. L'operazione di shift-row funziona così: si immagini di ricoprire i 16 byte di $s = s_0s_1...s_{15}$ dall'alto verso il basso, da sinistra verso destra per creare un tabellina 4x4 come mostrato qui sotto.

$$\begin{matrix} s_0 & s_4 & s_8 & s_{12} \\ s_1 & s_5 & s_9 & s_{13} \\ s_2 & s_6 & s_{10} & s_{14} \\ s_3 & s_7 & s_{11} & s_{15} \end{matrix}$$

Lo shift-rows sposta in maniera circolare a sinistra, la seconda riga di una posizione, la terza di due posizioni, la quarta di tre posizioni. La prima riga non viene mai spostata.

shift-rows($s_0s_1...s_{15}$)=$s_0s_5s_{10}s_{15}s_4s_9s_{14}s_3s_8s_{13}s_2s_7s_{12}s_1s_6s_{11}$

Mix-cols prende ciascuna delle quattro colonne della matrice 4x4 ed applica mix-cols(s) su parole a 4 byte estendibili fino a quantità di 16 byte a livello di parola. Il valore di mix-cols($a_0a_1a_2a_3$)= $a'_0a'_1a'_2a'_3$ è definito come segue:

$$\begin{pmatrix} a'_0 \\ a'_1 \\ a'_2 \\ a'_3 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 02 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix}$$

Un modo equivalente per descrivere questo passaggio è dire che stiamo moltiplicando $a(x) = a_3x^3+a_2x^2+a_1x+a_0$ con il polinomio fisso $c(x) = \{03\}x^3+\{01\}x^2+\{01\}x+\{02\}$ e prendiamo il resto in modulo x^4+1. Arrivati a questo punto non ci rimane che descrivere la funzione expand(K), perciò riassumiamo i passaggi principali.

function expand(K)
$K_0 \leftarrow K$
for i ← 1 to 10 **do**
  $K_i[0] \leftarrow K_{i-1}[0] \oplus S(K_{i-1}[3] \langle\langle\langle 8) \oplus C_i$
  $K_i[1] \leftarrow K_{i-1}[1] \oplus K_i[0]$
  $K_i[2] \leftarrow K_{i-1}[2] \oplus K_i[1]$
  $K_i[3] \leftarrow K_{i-1}[3] \oplus K_i[2]$
od
return (K_0,...,K_{10})

L'algoritmo mappa una chiave a 128 bit in undici sottochiavi da 128 bit, K0,...,K10. Le costanti C1,...,C10 sono ({02000000}, {04000000}, {08000000}, {10000000}, {20000000}, {40000000}, {80000000}, {1B000000}, {36000000}, {6C000000 }). Abbiamo terminato la descrizione dell'algoritmo AES, vista la complessità, per ulteriori approfondimenti si consiglia la visione del materiale bibliografico di riferimento.[7] Concludiamo dicendo che una delle proprietà di AES è che la mappa è invertibile, e ciò deriva dall'invertibilità di ogni round. Un round è invertibile perchè ognuno dei suoi steps è invertibile; tutto questo è una conseguenza diretta del fatto che S è una permutazione e la matrice usata in mix-cols ha un inverso.

Implementazione

Analisi

Ai fini dello sviluppo analizziamo i requisiti che l'applicazione vuole soddisfare.

Obiettivo

Dopo aver presentato i Tag e aver descritto le loro funzionalità principali, mi pongo l'obiettivo di creare un'applicazione che permetta l'autenticazione, la configurazione, l'attivazione di SDM e la modifica delle chiavi di un Tag. Nello specifico, si vuole configurare il servizio di Secure Dynamic Messaging andando ad inserire in ogni Tag l'indirizzo URL di un server, configurando il Tag per cifrare in mirroring UID e SDMReadCtr ed un'informazione da passare al server. Ogni lettore che si interfaccerà con il Tag sarà in grado di interagire con il server grazie al formato NDEF integrato nei sistemi Android che, attivando il Tag, vedrà subito aprirsi la pagina del browser predefinito e si troverà reindirizzato all'URL contenuto nel Tag. Il fine è quello di sfruttare queste funzionalità integrate nel chip favorendo l'interoperabilità nei sistemi complessi in cui vengono adottati questi device, ma sempre tenendo alto il livello di sicurezza.

Requisiti funzionali

Si richiede che l'applicazione permetta ad un utente di configurare in maniera smart un Tag, abilitando il Secure Dynamic Messaging al fine di permettere al Reader di interfacciarsi con un server web e far interagire le componenti. E' necessario che l'applicazione preveda una funzione per il cambio delle chiavi.

Requisiti non funzionali

L'implementazione e l'uso di questa tecnologia deve garantire efficienza e sicurezza all'interno del sistema. I Tag non devono essere messi sotto stress per svolgere le funzionalità di interazione. Devono essere previsti dei criteri di sicurezza sufficienti a garantire una comunicazione sicura e la protezione dei dati da vari tipi di attacchi.

Analisi e modello del dominio

L'applicazione necessita di autenticarsi al Tag per poter andare ad agire sulle impostazioni di configurazione: inizialmente le 5 application keys (compresa la Master Key) sono tutte azzerate, dunque possiamo usare una qualunque chiave e considerare che il suo valore corrisponde a 128 bit (32 caratteri esadecimali) tutti a 0. La configurazione prevede che venga attivato SDM con mirroring: dovremo quindi dire che vogliamo specchiare UID, SDMReadCtr (dopo averlo attivato) e prevedere che la codifica venga fatta in AES mode utilizzando l'SDMMetaReadKey. Una volta configurato il Tag, dobbiamo calcolare in maniera opportuna gli offset, per poter inserire le informazioni corrette nel chip; calcolati e settati gli offset andremo a scrivere il nostro URL e i dati nella posizione opportuna. Sappiamo, inoltre, che il Tag NT4H2421Tx prevede il servizio di SDM sul File No. 2 ed è proprio lì che dobbiamo andare a scrivere. L'applicazione, oltre a contenere le chiavi private per autenticarsi, possiede già l'URL da inserire nel chip, questo è un fattore che garantisce sicurezza. Ritengo che il requisito funzionale di cambio delle chiavi, azione che è sempre doverosa alla fine di qualsiasi configurazione per far sì che altri utenti non possano collegarsi al chip in modalità autenticata, la modifica dei permessi, e le garanzie apportate dall'utilizzo di SDM descritte anche nell'apposito capitolo, siano sufficienti a soddisfare i requisiti non funzionali sulla sicurezza e a garantire interoperatività. Non dovendo installare software esterni, ma andando a sfruttare al meglio le funzionalità integrate nei Tag, credo possa ritenersi soddisfatto anche il requisito non funzionale che prevede fluidità e dinamicità dell'uso di queste tecnologie all'interno di sistemi complessi.

Layout

Viene data rilevanza a questo a questo aspetto in quanto utile per un approccio “divide et impera", il processo per la realizzazione dell'obiettivo designato è complesso, si preferisce scomporre il problema di configurazione in più parti: scrittura, autenticazione e configurazione. Inizialmente per testare le singole funzionalità sono stati utilizzati tre bottoni diversi in modo da rendere più chiari i risultati. La figura qui sotto mostra il layout iniziale.

In seguito, è stato realizzato un layout con un unico bottone che racchiude questi tre passaggi, evitando all'utente in fase di utilizzo la possibilità che scada la sessione di connessione tra un comando e l'altro e il dover effettuare più tap per raggiungere un unico risultato. L'immagine qui sotto mostra il layout finale.

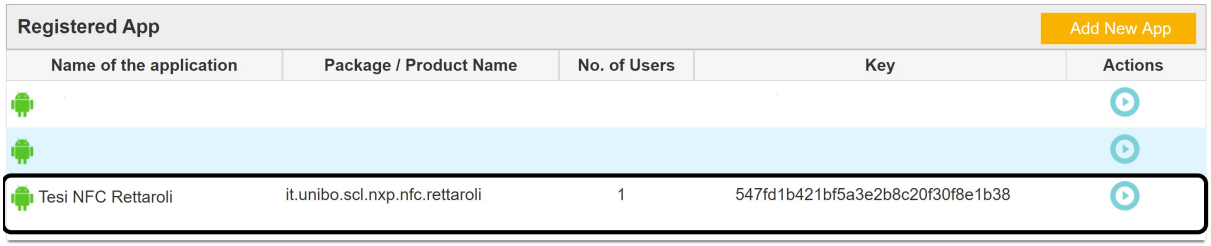
L'applicazione si compone di tre scroll view nelle quali vengono mostrati: i metadati NFC, i Log delle APDU e il contenuto della memoria. Il layout è molto semplice e funzionale.

Sviluppo

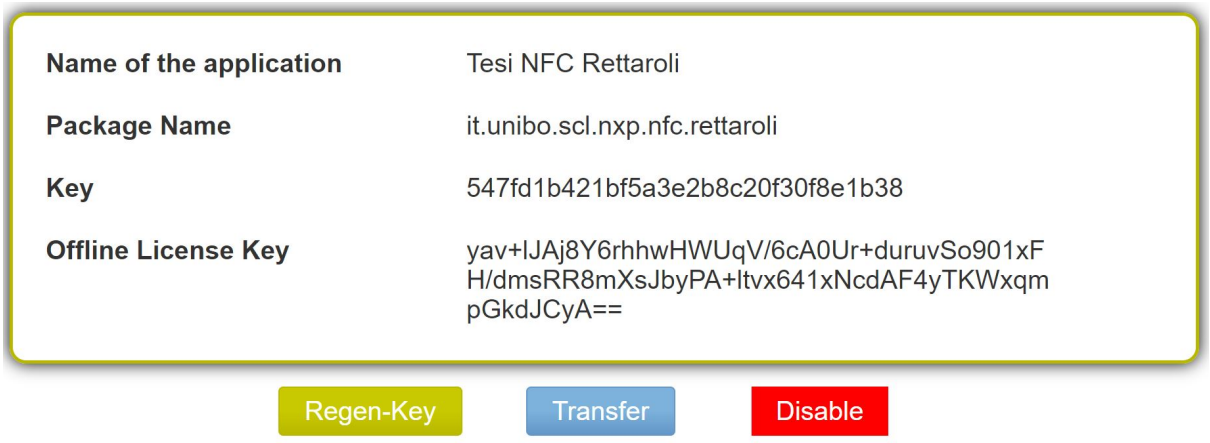
Ai fini dello sviluppo analizziamo gli aspetti principali distinguendoli in: scrittura del Tag, autenticazione, configurazione delle impostazioni e cambio delle chiavi. Le sezioni seguenti si occupano di descrivere questi aspetti tramite due approcci: alto livello e basso livello. Si ritiene importante analizzare la logica di basso livello al fine di comprendere al meglio le operazioni che si andranno ad eseguire sul Tag. In seguito viene presentata la libreria utilizzata per le funzioni ad alto livello.

TapLinx

MIFARE mette a disposizione delle librerie che consentono di interagire con i Tag lavorando ad alto livello. Una volta registrati sul sito ufficiale MIFARE è possibile accedere alla sezione TapLinx dove si può scegliere tra TapLinx Android e TapLinx Java, inoltre, vengono messe a disposizione: note di rilascio, un'applicazione di esempio, note di applicazione e la JavaDoc. Una volta scaricato il file, è necessario aprire le note di applicazione che illustrano i passaggi da eseguire per inserire correttamente le librerie nel nostro progetto. Il primo passo è effettuare la registrazione dell'applicazione nel portale TapLinx sul sito, come mostra la figura sotto.



Una volta registrata l'applicazione abbiamo a disposizione delle chiavi di licenza che servono per istanziare la libreria.



La figura sopra mostra le chiavi di licenza registrate, io ho scelto di non utilizzare la libreria seguendo l'approccio Maven, ma, ho preferito inserirla all'interno dell'applicazione al fine di evitare problemi derivanti dagli aggiornamenti automatici della libreria. Ho preferito attivare anche la chiave di licenza offline. Viene inoltre richiesto di modificare il Manifest inserendo il seguente codice:

```
<uses-feature android:name="android.hardware.nfc" android:required="true" />
<uses-permission android:name="android.permission.NFC" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

Una volta effettuati correttamente tutti i passaggi descritti nelle note di applicazione, nel metodo onCreate viene chiamato il metodo initializeLibrary così come descritto sotto.

```
    private void initializeLibrary(){
        try {
            m_libInstance = NxpNfcLib.getInstance();
            m_libInstance.registerActivity(this, m_strKey,m_m_strKey_offline);
        }catch (Exception e){
            Log.d(TAG,e.getMessage());
        }
    }
```

Le API sono disponibili nella sezione JavaDoc.

Scrittura del Tag

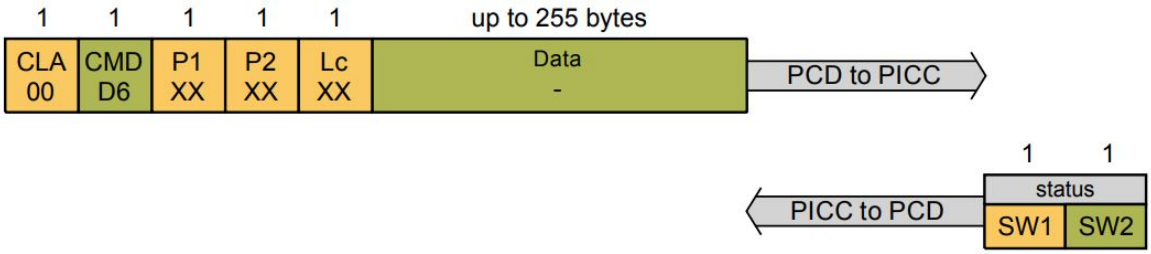
Come descritto nella documentazione del NTAG 424 DNA Tag Tamper, è possibile scrivere nel Tag con un approccio a basso livello tramite C-APDU. La tabella sotto descrive i due comandi a basso livello.

Comando	C-APDU(hex)							R-APDU		Modalità di comunicazione
ISOUpdateBinary	00	D6	XX	XX	XX	Dati	-	-	9000	CommMode.Plain
WriteData	90	8D	00	00	XX	Dati	00	-	9100	A seconda del file scelto

Entriamo nello specifico approfondendo tutti i campi presenti in entrambi i comandi.

ISOUpdateBinary

Il comando ISOUpdateBinary è implementato in conformità con ISO/IEC 7816-4, è possibile solo con CommMode.Plain per un file di dati standard. NT4H2421Tx supporta la tearing protection per i dati inviati all'interno di un frame di comunicazione al file. La figura sotto riassume lo scambio di dati che avviene quando si esegue questo comando.



La tabella seguente riporta la descrizione di ogni parametro.

Nome	Lunghezza	Valore	Descrizione
CLA	1	00h	

Nome	Lunghezza	Valore	Descrizione
INS	1	D6h	
P1	1		ShortFile ID/Offset
	bit 7		RFU
		1b	P1[Bit 6..5] are RFU. P1[Bit 4..0] encode a short ISO FileID. P2[Bit 7..0] encode an offset from zero to 255.
		0b	P1 - P2 (15 bits) encode an offset from zero to 32767.
	bit 6-5	00b	[if P1[7] == 1b] RFU
	bit 4-0		[if P1[7] == 1b] short ISO FileID
		00h	Targeting currently selected file.
		01h .. 1Eh	Targeting and selecting file referenced by the given short ISO FileID.
		1F	RFU
	bit 6-0	vedi P2	[if P1[7] == 0b] Most significant bits of Offset
P2	1	000000h .. (FileSize - 1)	Offset
Lc	1	01h .. (FileSize - Offset)	Length of subsequent data field
Data	fino a 255byte	XX	Data to be written

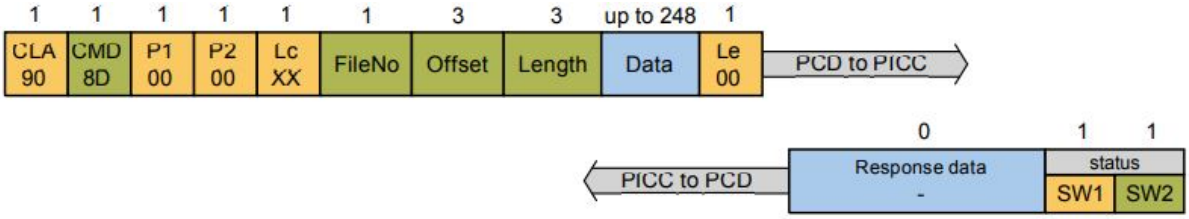
La risposta è strutturata come mostrato sotto.

Nome	Lunghezza	Valore	Descrizione
SW1SW2	2	9000h XXXXh	Successful execution Error

Per avere la piena visione dei possibili errori si consiglia di consultare la documentazione.[5]

WriteData

Il comando WriteData consente di scrivere dati su file StandardData. NT4H2421Tx supporta la tearing protection per i dati inviati all'interno di un frame di comunicazione al file. A seconda delle impostazioni della modalità di comunicazione del file dati, è necessario inviare i dati con: CommMode.Plain, CommMode.MAC o CommMode.Full. In caso di CommMode.Full, tutto crittografato, le operazioni vengono eseguite in modalità CBC. In caso di CommMode.MAC o CommMode.Full, la validità dei dati verificata dal PICC controllando il MAC. Se la verifica fallisce, il PICC interrompe l'ulteriore programmazione della memoria utente e restituisce un errore di integrità al file PCD. Come conseguenza dell'errore di integrità, qualsiasi transazione, che potrebbe essere iniziata, viene automaticamente interrotta. La figura sotto riassume lo scambio di dati che avviene quando si esegue questo comando.



La tabella seguente riporta la descrizione dei parametri.

Nome	Lunghezza	Valore	Descrizione
Cmd	1	8Dh	Command Code.
FileNo	1	-	File number of the targeted file.
	bit 7-5	000b	RFU
	bit 4-0		File number
Offset	3	000000h .. (FileSize - 1)	Starting position for the write operation.
Length	3	000001h .. (FileSize - Offset)	Number of bytes to be written.
Data	up to 248	full range	Data to be written.

La risposta è strutturata come mostrato sotto.

Nome	Lunghezza	Valore	Descrizione
Response data	0	-	No response data
SW1SW2	2	9100h 91XXh	Successful execution Error

Per avere la piena visione dei possibili errori si consiglia di consultare la documentazione.[5]

Ora che abbiamo compreso questi due comandi vediamo la procedura da seguire per ottenere lo stesso risultato con le API TapLinx. In seguito viene presentato il codice da seguire.

```
byte[] NTAG424DNATT_APP_NAME =
                {(byte) 0xD2, 0x76, 0x00, 0x00, (byte) 0x85, 0x01, 0x01};
        if(CardType.NTAG424DNATagTamper==m_libInstance.getCardType(intent)){
            INTAG424DNATT objNtag424tt=DESFireFactory.getInstance()
                            .getNTAG424DNATT(m_libInstance.getCustomModules());
            objNtag424tt.getReader().connect();
            objNtag424tt.getReader().setTimeout(2000);
            //This command is implemented in compliance with ISO/IEC 7816-4.It selects
an application by               the DFName
            objNtag424tt.isoSelectApplicationByDFName(NTAG424DNATT_APP_NAME);
            //writeData( int fileNO(02h),int offset (000000h),byte[] stringa in ASCI
esadecimale)
             objNtag424tt.writeData(2,0,UrlToWriteByte);
        }
```

La prima operazione da eseguire è proprio quella di andare a inserire il nostro URL all'interno del Tag, nel fileNo 02h. L'URL da inserire è nella seguente forma: http://192.168.178.21/a.php?e=00000000000000000000000000000000&c=0000000000000000; predisposto per la configurazione. A seguito di quanto detto nel capito NDEF e consapevoli del fatto che il File 02h supporta NDEF, costruiamo un pacchetto NDEF da inviare con il comando ISOUpdateBinary. L'URL sopra descritto diventerà:

```
String NDEF_lenght="004F"; //NDEF lenght
String NDEF_header="D1014B5503"; //NDEF header, 55 03 indica che è nell'elenco
presentato nel capitolo NDEF più precisamente significa http://
String
NDEF_data="3139322E3136382E3137382E32312F612E7068703F653D30303030303030303030303030303030303
03030303030303030303030303030303030303026633D30303030303030303030303030303030" //equivale a
192.168.178.21/a.php?e=00000000000000000000000000000000&c=0000000000000000

// il comando ISOUpdateBinary completo

 tag.transceive(UtilityManager.hex2Byte("00D6000051004FD1014B55033139322E3136382E3137382
E32312F612E7068703F653D30303030303030303030303030303030303030303030303030303030303030303030302
6633D30303030303030303030303030303030"));
```

La situazione nella memoria del Tag a seguito di questa operazione è la seguente:

```
EF104Content: //equivale al file 02h
00:4F:D1[1.1.0.1.0.003]:01:4B:55:03:31:39:32:2E:31:36:38:2E:31:37:38:2E:32:31:2F:61:2E:7
0:68:70:3F:65:3D:30:30:30:30:30:30:30:30:30:30:30:30:30:30:30:30:30:30:30:30:30:30:30:30:30:30
:30:30:30:30:30:30:30:30:26:63:3D:30:30:30:30:30:30:30:30:30:30:30:30:30:30:30:30:30:30//e poi
tutti 0
```

E' già possibile connettersi al server Web attraverso il tap del Tag. Per configurare il file e poter applicare SDM è necessario effettuare l'autenticazione.

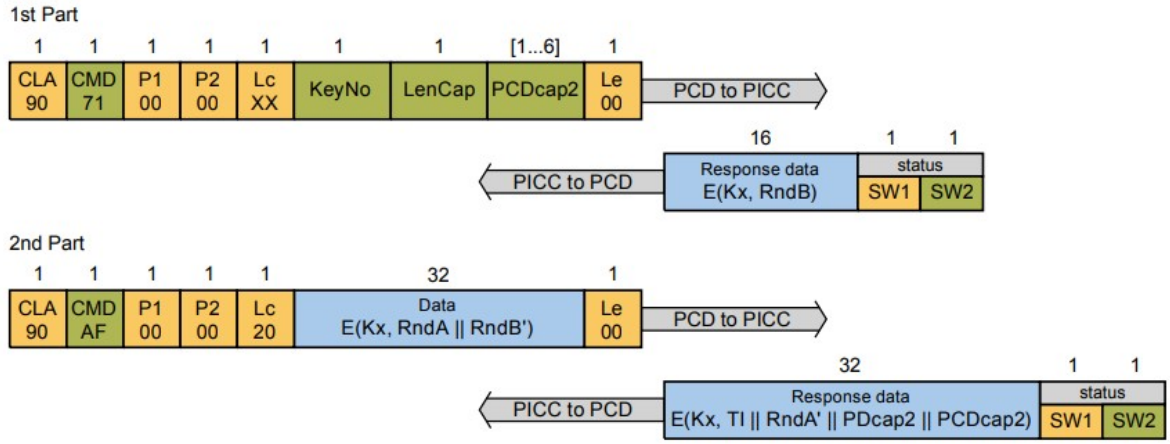
Autenticazione

Approfondiamo lo scambio di dati che avviene tra Tag e Reader in fase di autenticazione, iniziamo con un approccio ad alto livello. Il seguente codice mostra come è possibile autenticarsi al Tag utilizzando le API TapLinx.

```
private void CardLogic(final Intent intent){
    cardType=m_libInstance.getCardType(intent);
    byte[] KEY_AES128_DEFAULT = {
                (byte) 0x00, (byte) 0x00, (byte) 0x00, (byte) 0x00,(byte) 0x00, (byte)
0x00,
                (byte) 0x00, (byte) 0x00, (byte) 0x00, (byte) 0x00,(byte) 0x00,
                (byte) 0x00, (byte) 0x00, (byte) 0x00,
                (byte) 0x00, (byte) 0x00
        };
    byte[] NTAG424DNATT_APP_NAME =
                {(byte) 0xD2, 0x76, 0x00, 0x00, (byte) 0x85, 0x01, 0x01};
    try{
        if(CardType.NTAG424DNATagTamper==m_libInstance.getCardType(intent)){
            INTAG424DNATT objNtag424tt=DESFireFactory.getInstance()
                            .getNTAG424DNATT(m_libInstance.getCustomModules());
            objNtag424tt.getReader().connect();
            objNtag424tt.getReader().setTimeout(2000);
            objNtag424tt.isoSelectApplicationByDFName(NTAG424DNATT_APP_NAME);
                Log.d(TAG, "selected");
            SecretKeySpec keyDesDefault=new SecretKeySpec(KEY_AES128_DEFAULT,"AES");
            KeyData keydatadesDefault=new KeyData();
            keydatadesDefault.setKey(keyDesDefault);
                Log.d(TAG, "Authenticate with AES key 0");
            objNtag424tt.authenticateEV2First(0, keydatadesDefault, null);
                Log.d(TAG, "Authenticate ");
            }
        }catch(Exception e){
            Log.d(TAG,e.getMessage());
        }

}
```

Ora scendiamo a basso livello per comprendere meglio il meccanismo di autenticazione. L'immagine sotto mostra la struttura del comando authenticateEV2First.



La tabella seguente riporta la descrizione dettagliata dei parametri della prima parte.

Nome	Lunghezza	Valore	Descrizione
CMD	1	71h	Command code.
KeyNo	1		Targeted authentication key
	bit 7-6	00b	RFU
	bit 5-4	0h to 4h	Key number
LenCap	1	00h to 06h	Length of the PCD Capabilities. [This value should be set to 00h].
PCDcap2.1	[1]	-	Capability vector of the PCD.
	bit 7-2	Full range	RFU. can hold any value
	bit 1	0b	EV2 secure messaging
PCDcap2.2-6	bit 0	Full range	RFU, can hold any value
	[1 .. 5]	Full range	Capability vector of the PCD. All other bytes but PCDcap2.1 are optional, RFU and can hold any value. [If LenCap set to 00h, no PCDcap2 present]

La risposta è strutturata come mostrato sotto.

Nome	Lunghezza	Valore	Descrizione
E(Kx, RndB)	16	Full range	Encrypted PICC challenge The following data, encrypted with the key Kx referenced by KeyNo: - RndB: 16 byte random from PICC
SW1SW2	2	91AFh 91XXh	Successful execution Error

Per avere la piena visione dei possibili errori si consiglia di consultare la documentazione.[5]

La tabella seguente riporta la descrizione dettagliata dei parametri della seconda parte. Si precisa che || è un operatore che indica la concatenazione.

Nome	Lunghezza	Valore	Descrizione
CMD	1	AFh	Additional frame.

Nome	Lunghezza	Valore	Descrizione
E(kx, RndA || RndB')	32	Full range	Encrypted PCD challenge and response
			The following data, encrypted with the key Kx referenced by KeyNo: - RndA: 16 byte random from PCD. - RndB': 16 byte RndB rotated left by 1 byte

La risposta della seconda parte è strutturata come mostrato sotto.

Nome	Lunghezza	Valore	Descrizione
E(Kx,TI|| RndA' || PDcap2 || PCDcap2)	32	Full range	Encrypted PICC response The following data encrypted with the key referenced by KeyNo: - TI: 4 byte Transaction Identifier - RndA': 16 byte RndA rotated left by 1 byte. - PDcap2: 6 byte PD capabilities - PCDcap2: 6 byte PCD capabilities
SW1SW2	2	9100h 91XXh	Successful execution Error

Per avere la piena visione dei possibili errori si consiglia di consultare la documentazione.[5]

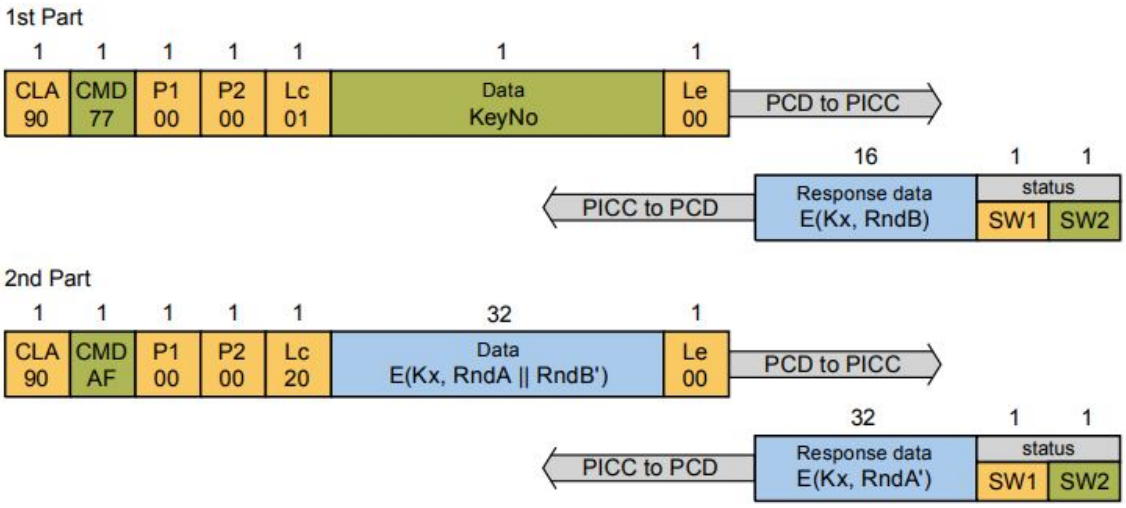
Si presenta una soluzione di C-APDU che mostra come quanto detto viene realizzato a basso livello. L'implementazione avviene utilizzando la classe Utilities TapLinx per la codifica in AES.

```
//prima parte dell'autenticazione
    //CLA: 90 (enc)
    //CMD: 71 (AuthenticateEV2firs)
    //P1: 00 (Select by DF name)
    //P2: 00
    //Lc: 02 (3 bytes data)
    //Data: keyNo, LenCap, PCDcap2
    //keyNo: 00
    //LenCap: 00
    //PCDcap2:00 non presente
    //Le: 00
    public APDU_NT4H2421Gx_response AuthenticateEv2FirstPt1(){
        byte[] response = null;
        APDU_NT4H2421Gx_response resp = null;
        try {
            response = tag.transceive(UtilityManager.hex2Byte("9071000002000000"));
            resp = new APDU_NT4H2421Gx_response(response, "AuthenticateEv2FistPt1");
```

```
            String Enc_RndB=UtilityManager.byte2Hex(resp.getData());
            KeyData aesKeyData = new KeyData();
            Key keyDefault = new SecretKeySpec(KEY_AES128_DEFAULT, "AES");
            aesKeyData.setKey(keyDefault);
            byte[] VI= KEY_AES128_DEFAULT;
            byte[] RndB=Utilities.aes(resp.getData(),
Cipher.DECRYPT_MODE,VI,KEY_AES128_DEFAULT);
            RndB=Utilities.rotateOneByteLeft(RndB);
            byte[] RndA=Utilities.generateRandom(16);
            ByteArrayOutputStream outputStream = new ByteArrayOutputStream( );
            outputStream.write(RndA);
            outputStream.write(RndB);
            byte RndA_RndB[] = outputStream.toByteArray();

 Enc_RndA_RndB=Utilities.aes(RndA_RndB,Cipher.ENCRYPT_MODE,VI,KEY_AES128_DEFAULT);
        } catch (IOException | GeneralSecurityException e) {
            e.printStackTrace();
        }
        return resp;
    }
    //seconda parte dell'autenticazione
    //CLA: 90 (enc)
    //CMD: AF (AuthenticateEV2firs)
    //P1: 00 (Select by DF name)
    //P2: 00
    //Lc: 20 (32 bytes data)
    //referenced by KeyNo:
            // RndA: 16 byte random from PCD.
            // RndB': 16 byte RndB rotated left by 1 byte
    //Le: 00
    public APDU_NT4H2421Gx_response AuthenticateEv2FirstPt2(){
        byte[] response = null;
        APDU_NT4H2421Gx_response resp = null;
        try {
            byte[] command=UtilityManager.hex2Byte("90AF000020");
            byte[] command_end=UtilityManager.hex2Byte("00");
            ByteArrayOutputStream outputStream = new ByteArrayOutputStream( );
            outputStream.write(command);
            outputStream.write(Enc_RndA_RndB);
            outputStream.write(command_end);
            byte cc[] = outputStream.toByteArray();
            response = tag.transceive(cc);
            resp = new APDU_NT4H2421Gx_response(response, "AuthenticateEv2FistPt2");
        } catch (IOException e) {
            e.printStackTrace();
        }
        return resp;
    }
```

Una volta effettuata questa operazione, è importante ricordare che viene applicato lo stato di autenticato, dopodiché si lavora in Secure Messaging come descritto nel capitolo Secure Messaging. Si consiglia, una volta effettuata l'authenticateEv2First di effettuare AuthenticateEV2FirstNonFirst che continua la transazione avviata dal precedente comando avviando una nuova sessione. Lo schema delle transazioni e delle sessioni all'interno della transazione sono state progettate per proteggere la comunicazione da possibili attacchi replay. La figura sotto riassume lo scambio di dati che avviene quando si esegue questo comando.



Non mostriamo le tabelle in quanto identiche a quelle del comando AuthenticateEV2First. Si presenta invece un'implementazione ad alto livello.

```
//è sufficiente richiamare questa funzione dopo essersi autenticati
objNtag424tt.authenticateEV2NonFirst(0, keydatadesDefault, null);
```

E a basso livello.

```
 //AUTENTICATEEV2 NON FIRST pt1
    //cla 90
    //cmd 77
    //p1 00
    //p2 00
    //lc 01
    //key no 00
    //le 00
    public APDU_NT4H2421Gx_response AuthenticateEv2NonFirstPt1(){
        byte[] response = null;
        APDU_NT4H2421Gx_response resp = null;
        try {
            response = tag.transceive(UtilityManager.hex2Byte("90770000010000"));
            resp = new APDU_NT4H2421Gx_response(response, "AuthenticateEv2NonFistPt1");
            String Enc_RndB=UtilityManager.byte2Hex(resp.getData());
            KeyData aesKeyData = new KeyData();
            Key keyDefault = new SecretKeySpec(KEY_AES128_DEFAULT, "AES");
            aesKeyData.setKey(keyDefault);
            byte[] VI= KEY_AES128_DEFAULT;
            byte[] RndB=Utilities.aes(resp.getData(),
Cipher.DECRYPT_MODE,VI,KEY_AES128_DEFAULT);
            RndB=Utilities.rotateOneByteLeft(RndB);
            byte[] RndA=Utilities.generateRandom(16);
            ByteArrayOutputStream outputStream = new ByteArrayOutputStream( );
            outputStream.write(RndA);
            outputStream.write(RndB);
            byte RndA_RndB[] = outputStream.toByteArray();
            Enc_RndA_RndB_forEV2NonFirst=Utilities

.aes(RndA_RndB,Cipher.ENCRYPT_MODE,VI,KEY_AES128_DEFAULT);
        } catch (IOException | GeneralSecurityException e) {
            e.printStackTrace();
        }
        return resp;
    }
```

```
//AuthenticateEV2NonFirstp2
//CLA: 90 (enc)
//CMD: AF (AuthenticateEV2nonfirs)
//P1: 00 (Select by DF name)
//P2: 00
//Lc: 20 (32 bytes data)
//referenced by KeyNo:
// RndA: 16 byte random from PCD.
// RndB': 16 byte RndB rotated left by 1 byte
//Le: 00
public APDU_NT4H2421Gx_response AuthenticateEv2NonFirstPt2(){
    byte[] response = null;
    APDU_NT4H2421Gx_response resp = null;
    try {
        byte[] command=UtilityManager.hex2Byte("90AF000020");
        byte[] command_end=UtilityManager.hex2Byte("00");
        ByteArrayOutputStream outputStream = new ByteArrayOutputStream( );
        outputStream.write(command);
        outputStream.write(Enc_RndA_RndB_forEV2NonFirst);
        outputStream.write(command_end);
        byte cc[] = outputStream.toByteArray();
        response = tag.transceive(cc);
        resp = new APDU_NT4H2421Gx_response(response, "AuthenticateEv2NonFistPt2");
    } catch (IOException e) {
        e.printStackTrace();
    }
    return resp;
}
```

Una volta effettuata l'autenticazione è possibile eseguire le operazioni che non avvengono in CommandMode.Plain: ChangeFileSettings e ChangeKey.

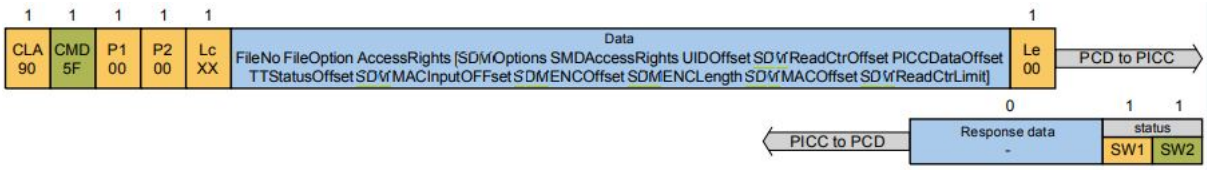
Configurazione

In questa fase si vuole configurare il FileNo 02h per abilitare il Secure Dynamic Messaging. Lavorando ad alto livello non vediamo che i messaggi vengono incapsulati da Secure Messaging e quindi crittografati. Per realizzare la configurazione ad alto livello, è sufficiente creare un oggetto NTAG424DNATTFileSettings fileSettings e chiamare il metodo changeFileSettings, qui sotto presentiamo una possibile implementazione.

```
            try {
                byte[] readAccess= UtilityManager.hex2Byte("E");
                byte[] writeAccess= UtilityManager.hex2Byte("E");
                byte[] readWriteAccess= UtilityManager.hex2Byte("E");
                byte[] change= UtilityManager.hex2Byte("0");
                NTAG424DNATTFileSettings nt= new NTAG424DNATTFileSettings
                    (MFPCard.CommunicationMode.Encrypted,readAccess[0],writeAccess[0],
                                                    readWriteAccess[0],change[0]);
                nt.setSDMEnabled(true);
                //sdmOptions
```

```
                nt.setUIDMirroringEnabled(true);
                nt.setSDMReadCounterEnabled(true);
                nt.setSDMReadCounterLimitEnabled(false);
                nt.setSDMEncryptFileDataEnabled(true);
                //sdmAccessRight 16bit 7-4 3-0 15-12 11-8
                nt.setSdmAccessRights(UtilityManager.hex2Byte("F000"));
                //PICCDataOffset =30 =>1Eh
                nt.setPiccDataOffset(UtilityManager.hex2Byte("1E0000"));
                //sdmMacInputOffset=65 =>41h
                nt.setSdmMacInputOffset(UtilityManager.hex2Byte("410000"));
                nt.setSdmMacOffset(UtilityManager.hex2Byte("410000"));
                //changeFileSettings(int FileNO,NTAG424DNATTFileSettings fileSettings);
                objNtag424tt.changeFileSettings(2,nt);
            }catch (Exception e){
                TextView tv = findViewById(R.id.nfc_memory_label);
                tv.setText("configurazione fallita: "+e.getMessage());
            }
```

Solo spostandoci a basso livello possiamo comprendere tutte le possibili opzioni di configurazione di SDM. La figura sotto riassume lo scambio di dati che avviene quando si esegue questo comando.



La tabella seguente riporta la descrizione dettagliata dei parametri.

Nome	Lunghezza	Valore	Descrizione
CMD	1	5Fh	Command code.
FileNo	1	-	File number of the targeted file.
	bit 7-5		RFU
	bit 4-0		File numeber
FileOption	1	-	Options for the targeted file.
	bit 7	0b	RFU
	bit 6		Secure Dynamic Messaging and Mirroring
		0b	disabled
		1b	enabled
	bit 5-2	0000b	RFU
	bit 1-0	11b 00b	CommMode.Full CommMode.Plain
AccessRight	2	-	Set of access conditions for the first set in the file
SDMOptions	[1]	-	[Optional, present if FileOption[Bit 6] set] SDM Options
	bit 7	-	UID(only for mirroring)
		0b	disabled

Nome	Lunghezza	Valore	Descrizione
		1b	enabled
	bit 6	-	SDMReadCtr
		0b	disabled
		1b	enabled
	bit 5	-	SDMReadCtrLimit
		0b	disabled
		1b	enabled
	bit 4	-	SDMENCFileData
		0b	disabled
		1b	enabled
	bit 3	-	TTstatus
		0b	disabled
		1b	enabled
	bit 2-1	00b	RFU
	bit 0	-	Encoding mode
		1b	ASCII
SDMAccessRights	[2]	-	[Optional, present if FileOption[Bit 6] set] SDM Access Rights
	bit 15-12	-	SDMMetaRead acces right
		0h .. 4h	Encrypted PICCData mirroring using the targeted AppKey
		Eh	Plain PICCData mirroring
		Fh	No PICCData mirroring
	bit 11-8	-	SDMFileRead access right
		0h .. 4h	Targeted AppKey
		Fh	No SDM for Reading
	bit 7-4	Fh	RFU
	bit 3-0	-	SDMCtrRet access right
		0h .. 4h	Targeted AppKey
		Eh	Free
		Fh	No Access
UIDOffset	[3]	-	[Optional, present if ((SDMOptions[Bit 7] = 1b) AND (SDMMetaRead access right = Eh)] Mirror position (LSB first) for UID
		0h .. (FileSize - UIDLength)	Offset within the file
SDMReadCtrOffset	[3]	-	[Optional, present if ((SDMOptions[Bit 6] = 1b) AND (SDMMetaRead access right = Eh)] Mirror position (LSB first) for SDMReadCtr

Nome	Lunghezza	Valore	Descrizione
		0h .. (FileSize - SDMReadCtrLength)	Offset within the file
		FFFFFFh	No SDMReadCtr mirroring
PICCDataOffset	[3]	-	[Optional, present if SDMMetaRead access right =0h..4h] Mirror position (LSB first) for encrypted PICCData
		0h .. (FileSize - PICCDataLength)	Offset within the file
TTStatusOffset	[3]	-	[Optional, present if (SDMOptions[Bit 3] = 1b)] Mirror position (LSB first) for TTStatus
		0h .. (FileSize-2)	Offset within the file
SDMMACInputOffset	[3]	-	[Optional, present if SDMFileRead access right != Fh] Offset in the file where the SDM MAC computation starts (LSB first)
		SDMMACInputOffset .. (SDMMACOffset - 32)	Offset within the file
SDMENCOffset	[3]	-	[Optional, present if ((SDMFileRead access right != Fh) AND (SDMOptions[Bit 4] = 1b))] SDMENCFileData mirror position (LSB first)
		SDMMACInputOffset .. (SDMMACOffset - 32)	Offset within the file
SDMENCLength	[3]	-	[Optional, present if ((SDMFileRead access right != Fh) AND (SDMOptions[Bit 4] = 1b))] Length of the SDMENCFileData (LSB first)
		32 .. (SDMMACOffset - SDMENCOffset)	Offset within the file, must be multiple of 32
SDMMACOffset	[3]	-	[Optional, present if SDMFileRead access right != Fh] SDMMAC mirror position (LSB first)
		SDMMACInputOffset .. (FileSize - 16)	[if (SDMFileRead access right != Fh) AND (SDMOptions[Bit 4] = 0b)] Offset within the file
		(SDMENCOffset + SDMENCLength) .. (FileSize- 16)	[if (SDMFileRead access right != Fh) AND (SDMOptions[Bit 4] = 1b)] Offset within the file
SDMReadCtrLimit	[3]	Full range	[Optional, present if SDMOptions[Bit 5] = 1b] SDMReadCtrLimit value (LSB first)

La risposta è strutturata come mostrato sotto.

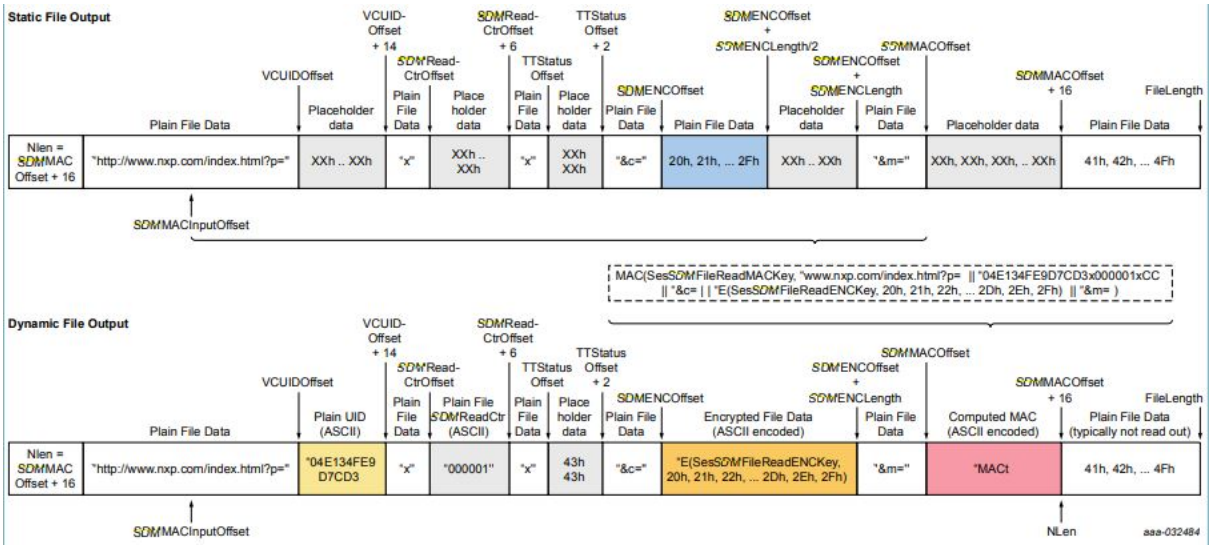
Nome	Lunghezza	Valore	Descrizione
Response data	0	-	No response data
SW1SW2	2	9100h 91XXh	Successful execution Error

Per avere la piena visione dei possibili errori si consiglia di consultare la documentazione.[5]

In seguito viene mostrata una versione delle C-APDU non incapsulata secondo Secure Messaging.

```
    //CLA: 90 (enc)
    //CMD: 5F (changeFileSettings)
    //P1: 00 (Select by DF name) o 02
    //P2: 00
    //Lc: 10 (16 bytes data)
    //FileNO: 02h  1byte
    //file option 43
    //accessright 00E0 se l'ordine è 7-4 3-0 15-12 11-8=>RW change Read Write
    //sdmOption D1 11010001
    //sdmAccessRight scritto 7-4 3-0 15-12 11-8=> F000
    //piccDataOffset serve è 30=> 1E0000
    //SDMMacInputOfsset serve è 65=> 410000
    //sdmMacOffset serve    è 65=> 410000
    //le=00
    public APDU_NT4H2421Gx_response changeFileSettingsToConfigureSDM(){
        byte[] response = null;
        APDU_NT4H2421Gx_response resp = null;
        try {
            byte[] command=UtilityManager.hex2Byte("905F000010");
            byte[]
commandData=UtilityManager.hex2Byte("024300E0D1F0001E0000410000410000");
            byte[] commandEnd=UtilityManager.hex2Byte("00");
            ByteArrayOutputStream outputStream = new ByteArrayOutputStream( );
            outputStream.write(command);
            outputStream.write(commandData);
            outputStream.write(commandEnd);
            byte cd[]=outputStream.toByteArray();
            response = tag.transceive(cd);
            resp = new APDU_NT4H2421Gx_response(response,
"changeFileSettingsToConfigureSDM");
        } catch (IOException e) {
            e.printStackTrace();
        }
        return resp;
    }
```

Le possibilità di configurazione di SDM sono diverse, l'immagine sotto ci aiuta a capirle, e a capire come calcolare i vari offset necessari a seconda della configurazione.



Nelle parti di codice si desidera attenersi alla configurazione che produce questo risultato:

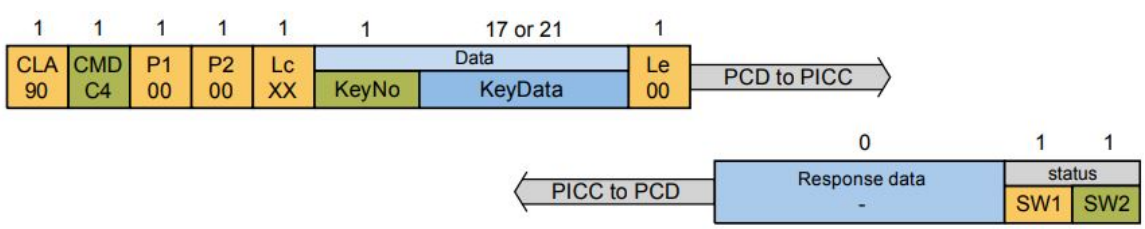
PICCENCData=E(KSDMMetaRead; PICCDataTag[||UID] [||SDMReadCtr] ||RandomPadding).

Cambio delle chiavi

Anche in questo caso lavorando ad alto livello non vediamo che i messaggi vengono incapsulati da Secure Messaging e quindi crittografati. Per realizzare la configurazione ad alto livello è sufficiente richiamare il metodo fornito da TapLinx.

```
    //This command is used to change the application keys.
    //changeKey(int keyNumber, byte[] currentKeyData, byte[] newKeyData, byte
newKeyVersion)
    try {
            byte[] newKeyVersion=UtilityManager.hex2Byte("01");
           objNtag424tt.changeKey(0,KEY_AES128_DEFAULT,KEY_AES128_PERSONALIZED,
                                                 newKeyVersion[0]);
        }catch (Exception e){
             e.getMessage();
        }
```

Anche qui scendiamo a basso livello per comprendere al meglio la struttura del comando. La figura sotto riassume lo scambio di dati che avviene quando si esegue questo comando. E' possibile effettuarlo solo in CommMode.Full.



La tabella seguente riporta la descrizione dei parametri.

Nome	Lunghezza	Valore	Descrizione
CMD	1	C4h	Command code.
KeyNo	1	-	Key number of the key to be changed.
	bit 7-6	00b	RFU
	bit 5-0		Key number
		0h .. 4h	The application key number
KeyData	17 o 21		new key data
		full range	if key 0 is to be changed NewKey || KeyVer
		full range	if key 1 to 4 are to be changed (NewKey XOR OldKey) || KeyVer || CRC32NK[13]

La risposta è strutturata come mostrato sotto.

Nome	Lunghezza	Valore	Descrizione
Response data	0	-	No response data
SW1SW2	2	9100h 91XXh	Successful execution Error

Il seguente codice mostra un implementazione a basso livello che deve essere però impastata secondo Secure Messaging.

```
//CHANGE KEY
    //cla 90
    //cmd C4
    //p1 00
    //p2 00
    //lc 12    1+17 perchè cambio la chiave 0 =18=>12h
    //keyNo 00
    //KeyData NewKey concatenato KeyVersion NewKey:01010101010101010101010101010101
KeyV: 00
    //le 00
    public APDU_NT4H2421Gx_response changeKeyZero(){
        byte[] response = null;
        APDU_NT4H2421Gx_response resp = null;

        try {
            response = tag.transceive(

UtilityManager.hex2Byte("90C400001200010101010101010101010101010101010000"));
            resp = new APDU_NT4H2421Gx_response(response, "changeKeyZero");
        } catch (IOException e) {
            e.printStackTrace();
```

```
        }

        return resp;
    }
```

Server Web

Per questa parte viene utilizzato XAMPP, tramite il quale è possibile creare una web page che sarà il contenuto del Tag. Interagiremo inserendo nella pagina PHP lo script Python che permette di decifrare UID e SDMReadCtr, in seguito viene presentato il codice.

```
#! /usr/bin/env python -2
from binascii import hexlify, unhexlify
from Crypto.Cipher import AES
import sys
if len(sys.argv)>0:
    var=sys.argv[1] #prende il parametro ovvero l'Enc_Picc_Data
# PICCData decryption
# PICCData = AES-128_DECRYPT(KSDMMetaRead; PICCDataTag[||UID]
[||SDMReadCtr]||RandomPadding)
IV = 16 * '\x00'
key = 16 * '\x00' # File02.SDMMetaRead Key
Enc_PICC_Data = sys.argv[1]
myAes=AES.new(key, AES.MODE_CBC, IV=IV)
PICCData = myaes.decrypt(unhexlify(Enc_PICC_Data))
PICCDataTag = hexlify(PICCData[0:1])
UID = hexlify(PICCData[1:8])
SDMReadCtr = hexlify(PICCData[8:11])
```

Sviluppi futuri

Immaginando uno scenario reale e di vasta portata, al fine di migliorare le prestazioni aumentando efficienza e sicurezza si prospetta di realizzare il Web server su AWS, Amazon Web Services, più precisamente si pensava di usufruire del servizio Amazon EC2. Amazon Elastic Compute Cloud (Amazon EC2) è un servizio Web che fornisce capacità di elaborazione sicura e scalabile nel cloud. È concepito per rendere più semplice il cloud computing. Mediante l'interfaccia Web service intuitiva di Amazon EC2 è possibile ottenere e configurare la capacità in modo semplice e immediato. L'utente ha il controllo completo delle proprie risorse informatiche. Amazon EC2 offre una piattaforma di calcolo vasta e diversificata con una scelta di: processore, spazio di

storage, sistema operativo e sistema di rete con Ethernet fino a 100 Gb/s; ciò permette la creazione di un Web server e simula uno scenario più reale di interoperabilità.

Guida utente

L'applicazione ha un layout molto intuitivo, ci si pone quindi nei panni dell'operatore che deve configurare i Tag. Premendo il bottone "Configure" automaticamente verranno effettuati i passaggi descritti nella parte di sviluppo. Si consiglia di ricontrollare che il contenuto inserito sia corretto e quindi di premere il tasto "Read", e visionare che tutto sia andato a buon fine. Qualora vi siano errori è possibile utilizzare il tasto "Clear" per riportare il Tag al contenuto iniziale. Infine si procede premendo il bottone "ChangeKey" che provvederà a cambiare la chiave di autenticazione garantendo così l'accesso solo a chi è a conoscenza della nuova chiave inserita. A seguito del click di ogni bottone va effettuato il tap con il Tag. Una volta realizzata questa procedura otterremo la condizione desiderata: il Tag avrà libero accesso solo in lettura, dunque, un qualsiasi Reader con lettore NFC che non dispone dell'applicazione è in grado di leggere il contenuto inserito nel Tag.

Conclusioni

In seguito ad un'attenta analisi è possibile notare che: per effettuare delle operazioni che modificano le impostazioni del Tag è necessario essere autenticati, le operazioni di configurazione e autenticazione non avvengono in chiaro, il contenuto del Tag viene crittografato secondo la configurazione scelta e il chip ha un meccanismo interno rilevazione degli attacchi fisici. Da questa analisi ritengo che i requisiti di sicurezza preposti siano pienamente soddisfatti. Tornando ad immaginare uno scenario complesso, possiamo ritenere che i Tag, se configurati e utilizzati nel modo giusto, sono un fattore capace di aumentare la dinamicità del sistema e ampliarne le capacità, permettendo una facile interazione e integrazione delle tecnologie connesse. Come abbiamo visto le funzioni integrate che si celano dietro questi chip sono molteplici e aumenta esponenzialmente la facilità di distribuzione, la compatibilità e l'interoperabilità con altri sistemi preesistenti; inoltre permettono di espandere le funzionalità del sistema attuale centralizzando le molteplici necessità legate ad un prodotto, quali: tracciamento, monitoraggio, conteggio e sicurezza. I costi legati a queste necessità vengono quindi drasticamente ridotti, come già detto, ne sono testimoni Tesco, River Island, Adidas, Decathlon, Lewis che hanno appurato ritorni economici legati alle vendite fino a un +5,5% e una riduzione dei costi di stoccaggio fino a -13%. Possiamo concludere affermando che i fattori che caratterizzano e danno potere di applicazione a questi Tag sono: interoperabilità e sicurezza.

Bibliografia, sitografia, fonti e riferimenti

[1]:

- *IFF systems*
- Wright, Peter (1987). *Spycatcher: L'Autobiografia Candid di un anziano funzionario dell'intelligence*. New York: Penguin. ISBN 0-670-82055-5 .
- Kennan, George (1967). *Memorie, 1925-1950*. Little, Brown. e *Memorie: 1950-1963* . Pantheon.
- Il Gran Sigillo Bug sul Crypto Museo Sito
- *Charlie Walton, inventor of RFID, passes away at 89*

- Standard ISO

[2]:

- Story of near field communication
- NFC forum LLCP introduction
- NFC operating modes 1
- NFC operating modes 2
- "How to NFC conference 2011"

[3]:

- NDEF

[4]:

- P. Talone e G. Russo, Rfid: tecnologie e applicazioni, Fondazione Ugo Bordoni. Disponibile online qui.

[5]:

- NT4H2421Tx NTAG 424 DNA TT – Secure NFC T4T compliant IC. Reperibile in Documentazione NTAG424 DNA Tag Tamper.

[6]

- I singoli standard sono definiti e accessibili dalla pagina ISO/IEC 7816.

[7]:

- J. Daemen and V. Rijmen, The design of Rijndael. Springer, 2002.
- Introduction to Modern Cryptography
- A. Bernasconi, P. Ferragina, F. Luccio, Elementi di crittografia.

---

1. Un componente delle telecomunicazioni; il nome deriva da Transmitter responder, un dispositivo capace di ricevere e ritrasmettere un messaggio o un segnale. ↩ ↩

2. International Organization for Standardization, un organizzazione che si occupa di definire delle norme tecniche a livello mondiale. ↩

3. Conferenza integrale dell'intervento "How to NFC" https://www.youtube.com/watch?v=49L7z3rxz4Q ↩

4. Due conduttori sono detti essere **accoppiati induttivamente** o **accoppiati magneticamente** quando sono configurati in modo tale che un cambiamento nella corrente attraverso un filo induce una tensione attraverso le estremità dell'altro filo per mezzo dell'induzione elettromagnetica. Ulteriori informazioni. ↩

5. L'attacco relay/replay è una tecnica in cui un utente malintenzionato potrebbe implementare un dispositivo per intercettare una transazione NFC e riscattarla in un secondo momento, utilizzando un altro dispositivo o anche una posizione diversa. Per ulteriori informazioni consultare Relay/Replay Attack. ↩

6. Il disturbo di frequenza è l'interruzione dei segnali radio attraverso l'uso di un segnale sovralimentato nella stessa gamma di frequenza. Quando la maggior parte delle persone pensa al disturbo delle frequenze, ciò che viene in mente è il disturbo della radio, del radar e del telefono cellulare. ↩

7. Il Multipurpose Internet Mail Extensions (MIME) è uno standard di Internet che definisce il formato delle e-mail. ↩

8. Signature Record Type Definition è un protocollo di sicurezza utilizzato per proteggere l'integrità e l'autenticità dei messaggi NDEF. ↩ ↩

9. E' un insieme di algoritmi sviluppati ad uso civile dal tedesco Horst Feistel per IBM negli anni sessanta, la prima versione era stata sviluppata per rendere sicure le transazioni bancarie. ↩

10. Dispositivo costruito dall'Electronic Frontier Foundation per recuperare la chiave segreta di un messaggio cifrato DES utilizzando un attacco Brute Force, ovvero provando tutte le chiavi fino a trovare quella giusta. Per ulteriori informazioni consultare DES cracker. ↩ ↩

11. Un gruppo è una struttura algebrica formata dall'abbinamento di un insieme non vuoto con un'operazione binaria interna, tipicamente somma o prodotto che soddisfa gli assiomi di associatività, di esistenza dell'elemento neutro e di esistenza dell'inverso di ogni elemento. ↩

12. Per approfondimenti Joan Daemen; Lars Knudsen; Vincent Rijmen (1997). *Il Block Cipher Square*. ↩

13. Il CRC32NK sono 4-byte CRC calcolati in accordo con IEEE Std 802.3-2008. ↩