Università degli Studi di Padova

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE
Corso di Laurea Magistrale in Ingegneria Elettronica

Studio del protocollo Keeloq e implementazione di un attacco con analisi di potenza

Candidato:
Nicola Barbon
Matricola 603209-IL

Relatore:
prof. Lorenzo Vangelista

Anno Accademico 2010–2011

Abstract

Keeloq cryptographic algorithm is used in many commercial applications to achieve a good level of security in wireless transmissions. It is included in a lot of gate openers and anti-theft car systems. A lot of theoretical cryptanalysis literature is available on this algorithm, however pratical applications of the literature's finding are very limited. In this study a different approach that employs a side-channel technique for analyzing Keeloq real systems is used. This technique, based on Differential Power Analysis, allows to discover secret information from the transmitter and consequently to duplicate the transmitter device, making Keeloq systems less secure. This work shows successful attacks on HCS30X commercial devices.

Indice

Capitolo 1

Introduzione

L'algoritmo di crittografia Keeloq è largamente usato per sistemi di sicurezza come ad esempio i transponder di tipo Radio Frequency Identification (RFID) e i sistemi antifurto di ultima generazione, cosiddetti immobilizer, ma soprattutto per impianti di apertura radio utilizzati nell'automazione di cancelli e controllo accessi. Per questo vasto utilizzo, la sicurezza dell'algoritmo è cruciale e perciò la crittoanalisi svolta sul Keeloq è stata molto approfondita dalla comunità di ricerca internazionale. Dopo i primi studi, svolti da Bogdanov [1], sono stati presentati molti attacchi analitici che mostravano i punti deboli dell'algoritmo. Gli studi fatti su implementazioni Identify Friend or Foe (IFF), permettono di clonare un trasmettitore dopo aver acquisito un minimo di 216 coppie chiaro-cifrato e con qualche giorno-macchina di calcolo. Solo in alcuni casi di implementazione particolarmente semplici si è inoltre in grado di ricavare la chiave madre che consente di creare nuovi trasmettitori.
Malgrado i risultati positivi degli studi sul Keeloq, l'impatto sul mondo reale di questi attacchi è stato piuttosto limitato. Infatti, come prima cosa, tutti questi attacchi non sono applicabili all'applicazione Rolling Code dell'algoritmo che è la più utilizzata nei sistemi di apertura remota (che forma la gran parte del mercato Keeloq). Infatti non è possibile ottenere le coppie chiaro-cifrato poichè il messaggio in chiaro è custodito e inaccessibile nel dispositivo trasmettitore. In secondo luogo, la chiave master che permette la creazione di nuovi disposi-

tivi è situata nel ricevitore e nelle applicazioni commerciali è molto complessa da ricavare e per questo l'attacco risulta piuttosto limitato. Per questi motivi l'analisi sulla vulnerabilità del Keeloq nelle applicazioni reali è stata portata avanti su altri versanti, che potessero mettere alla prova i sistemi di sicurezza basati sull'algoritmo in applicazioni commerciali. Questi nuovi attacchi basati su tecniche side-channel che vengono qui studiate, hanno portato ad attacchi di successo sulla famiglia di controllori Microchip HCS30X e sono in via di studio su implementazioni software del Keeloq.
In questo studio vedremo in dettaglio l'algoritmo Keeloq e una panoramica sui tipi di attacchi che sono stati fatti per testarne la robustezza. Dopo di questo, vedremo in dettaglio l'attacco basato su Differential Power Analysis (DPA), in cosa consiste e come può essere applicato. L'attacco ha avuto successo ed è stato testato su più di un dispositivo della famiglia HCS30X, risultando quindi adatto ad applicazioni Rolling Code ed egualmente IFF.

Capitolo 2

Il keeloq

2.1 Keeloq: descrizione dell'algoritmo

Il keeloq è un blocco di cifratura basato su trame da 32 bit e su una chiave di cifratura a 64 bit. La sua struttura contiene uno shift register lungo 32 bit e una funzione di feedback con cinque bit in ingresso ed uno solo in output.
L'algoritmo proprietario è stato creato a metà degli anni Ottanta dal professor Gideon Kuhn, la parte di protocollo è stata seguita dal Dr. Frederick Bruwer, CEO di Nanoteq e l'implementazione hardware dal Dr. Willem Smith, di Nanoteq. Nel 1995 Microchip acquistò l'algoritmo per 10 milioni di dollari e venne implementato su encoder e decoder a codice variabile (detti Rolling code oppure Code Hopping) come la famiglia di dispositivi NTQ105/106/115/125D/129D e HCS101/2XX/3XX/4XX/5XX. Il keeloq venne subito utilizzato in dispositivi di accesso remoto da aziende come Fiat, GM, Honda, Volvo, Volkswagen etc. e in maniera ancora più diffusa negli ultimi anni, sui dispositivi di accesso con radiocomando per abitazioni e luoghi pubblici.

Per capire il funzionamento del Keeloq definiamo una trama di 32 bit $Y^{(i)} = (y_{31}^{(i)}, ..., y_0^{(i)}) \in V_{32}$, dove $V_n = GF(2)^n$ e $y_j^{(i)} \in GF(2)$ descrive lo stato del bit j-esimo al ciclo i-esimo. Definiamo poi allo stesso modo $K^{(i)} = (k_{63}^{(i)}, ..., k_0^{(i)}) \in V_{64}$, dove $k_j^{(i)} \in GF(2)$ identifica lo stato del bit j del registro K al ciclo i. Come si può vedere in Fig. 2.1 la codifica può essere sintetizzata

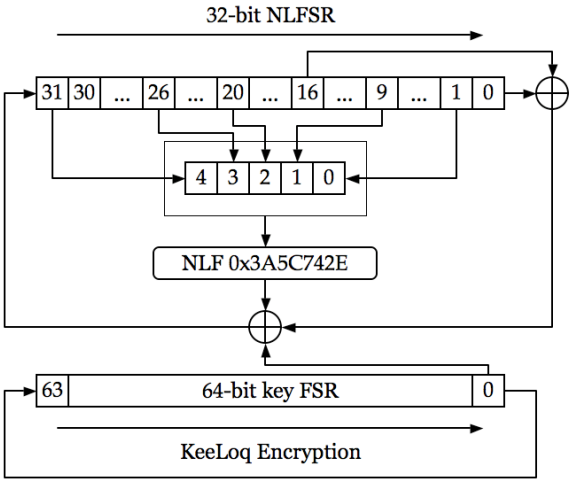

Figura 2.1: Schema di funzionamento del Keeloq nella fase di crittatura

in questo algoritmo:

- Calcolo del bit di feedback: $\varphi = NLF(y_{31}^{(i)}, y_{26}^{(i)}, y_{20}^{(i)}, y_{31}^{(9)}, y_1^{(i)}) \oplus y_{16}^{(i)} \oplus y_0^{(i)} \oplus k_0^{(i)}$
- Rotazione della trama e inserimento del feedback: $R^{(i+1)} = (\varphi, y_{31}^{(i)}, ..., y_1^{(i)})$
- Rotazione della chiave: $K^{(i+1)} = (k_0^{(i)}, k_{63}^{(i)}, ..., k_1^{(i)})$.

Il processo di codifica prevede che, al ciclo 0, il registro da 32 bit della trama contenga il testo in chiaro e che il registro della chiave contenga la chiave utilizzata dal Keeloq. L'esecuzione dell'algoritmo viene fatta 528 volte e nel registro da 32 bit, alla fine di tutti i cicli, sarà contenuto la trama cifrata. La funzione non lineare è $0x3A5C742E$ e può essere mappata su una tabella, come mostrato nel documento [2], oppure può essere scritta come $F(a, b, c, d, e) = d \oplus e \oplus ac \oplus ae \oplus bc \oplus be \oplus cd \oplus de \oplus ade \oplus ace \oplus abd \oplus abc$. I bit 1, 9, 20, 26 e 31 vengono inseriti in input nella funzione non lineare. L'output sarà poi messo in ingresso ad uno XOR assieme al bit 16 e 0 del registro della trama e al bit 0 del registro della chiave. Il risultato di queste operazioni

andranno ad aggiornare il bit 31 del registro della trama, mentre il resto del registro sarà shiftato di un bit a sinistra, allo stesso modo verrà fatto scorrere a sinistra il registro della chiave. In questo modo al ciclo successivo i bit che entreranno nella funzione non lineare saranno tutti diversi, pur mantenendo fisse le posizioni da cui la funzione estrae i bit dai registri.
Durante la decodifica il processo di calcolo è analogo, tuttavia cambiano i bit che vengono inseriti nella funzione non lineare e i versi di rotazione dei registri come si può vedere dalla Fig. 2.2. Infatti, i bit usati ad ogni ciclo di decodi-

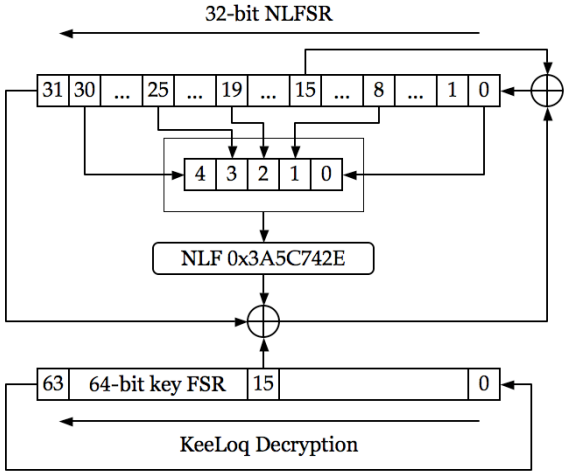

Figura 2.2: Schema di funzionamento del Keeloq nella fase di decrittatura

fica sono i bit 0, 8, 19, 25 e 30 del registro Y e nello XOR, assieme all'uscita del NLF, verranno inseriti i bit 31 e 15 del registro della trama e il bit 15 del registro della chiave. I registri vengono shiftati a destra di un bit ad ogni ciclo e il bit di feedback viene inserito nella posizione 0 del registro della trama.
La funzione non lineare è costruita in modo da massimizzare la sua immunità alla correlazione. Questo significa che, al cambiamento di pochi bit sulla trama in chiaro, il risultato della codifica cambierà di molto rispetto a prima, in modo da rendere impossibile eventuali correlazioni tra le varie trame codificate. La struttura a 528 cicli può essere scomposta in 8 blocchi identici da 64 permuta-

zioni ciascuno, che implicano un giro completo della chiave, e un blocco da 16 permutazioni, che compie solo un quarto di giro alla chiave.
Questo algoritmo bene si presta ad una facile implementazione hardware. La sua struttura richiede pochi componenti da implementare a livello fisico: un registro da 32 bit con collegamenti a sei posizioni fisse (bit in ingresso alla NLF e allo XOR), un registro da 64 bit con un collegamento ad una posizione fissa del bit (in ingresso allo XOR) e una look-up table (LUT) da $2^5 = 32$ bit. Dato che la NLF ha soltanto 5 ingressi, l'implementazione tramite LUT risulta essere semplice e poco dispendiosa in termini di hardware dedicato.

2.2 Implementazioni del Keeloq

L'algoritmo Keeloq viene integrato in diversi dispositivi, a seconda delle applicazioni, con metodologie diverse. Per questo motivo, l'algoritmo si trova implementato sia in hardware dedicato, sia in diversi microcontrollori via software. Questo permette di utilizzare il Keeloq nel modo che si ritiene più opportuno per le proprie necessità. Nel caso che andremo a studiare, trattandosi di radiocomandi avremo modo di vedere entrambe le implementazioni del Keeloq: nella parte del radiocomando troveremo un microcontrollore dedicato della famiglia Microchip HCS 1XX/2XX/3XX, mentre nella stazione ricevente la situazione si presenta diversa. Infatti la parte ricevente dovendo gestire le operazioni di apprendimento e di gestione di più radiocomandi si trova a dover avere un microcontrollore più flessibile e potente del semplice HCS e quindi utilizza un microcontrollore con all'interno implementato il Keeloq in maniera software. In alcune soluzioni, tuttavia, si trovano dispositivi che non seguono questa suddivisione e sempre più spesso i radiocomandi hanno al loro interno microcontrollori generici che, dovendo gestire funzionalità sempre più complesse, hanno integrato al loro interno una versione software del Keeloq senza avvalersi di un chip HCS separato. Inoltre Microchip fornisce altri dispositivi come ad esempio decoder per riceventi come i chip HCS 5XX, oppure come i microcontrollori con la funzione Keeloq implementata via hardware come la famiglia PIC 12F635/636/639 e altri dispositivi, come gli HCS 4XX, che hanno funzioni di

decoder ed encoder integrati.
In questo studio andremo a vedere prima di tutto i dispositivi HCS 300 e HCS 301 che sono simili e hanno caratteristiche di funzionamento pressochè uguali. Questi, essendo molto diffusi nei radiocomandi prodotti da molte case che utilizzano il Keeloq , hanno una grossa importanza e dalla loro sicurezza dipende molto la sicurezza di molti impianti dotati di Remote Keyless Entry (RKE).
L'HCS 301, e l'analogo HCS300, è un encoder a codice variabile (Rolling code) progettato per i sistemi RKE sicuri. Esso si presenta in un package Parallel Dual-in-Line Package (PDIP) o Small-outline Integrated Circuits (SOIC) a 8 piedini, i quali permettono di avere la gestione di 4 pulsanti e un led ed è equipaggiato con un'uscita Pulse-width Modulation (PWM). Come si può vedere dal diagramma dell'HCS 301 in Fig. 2.3, il chip è fornito di una Electrically Erasable Programmable Read-Only Memory (EEPROM) dove sono salvati la chiave di cifratura, il numero seriale e le informazioni di configurazione. La memoria, non accessibile da nessun collegamento esterno, è programmabile, ma protetta dalla scrittura e lettura. Questo consente di prevenire ogni possibile tentativo di leggere la chiave o altre informazioni dalla memoria.
La generazione della chiave da 64 bit, che è inserita nella memoria del trasmettitore all'atto della programmazione, può essere fatta in diversi modi. Solitamente si tende a seguire la procedura di generazione della chiave del radiocomando, consigliata dalla Microchip, che permette di creare chiavi diverse per ogni radiocomando, ma tutte ricostruibili con una sola chiave. Infatti partendo da una chiave master, chiamata chiave del produttore, e dal numero seriale del telecomando, con un algoritmo di generazione scelto inizialmente, si produce la chiave del radiocomando. In questo modo il ricevitore, con la sola chiave del produttore e con la conoscenza dell'algoritmo di generazione, sarà in grado di riconoscere tutti i radiocomandi precedentemente creati. Questo incrementa notevolmente la flessibilità delle soluzioni che si possono creare, senza togliere nè la retrocompatibilità con i ricevitori prodotti meno recentemente dalla stessa casa, nè la sicurezza del rolling code.
La creazione delle chiavi può essere fatta sostanzialmente in tre modi differenti,

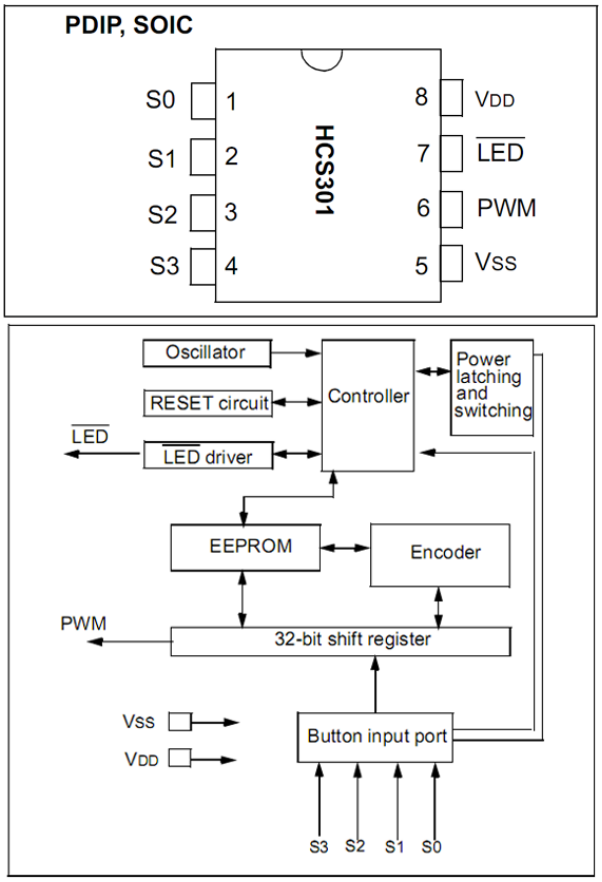

Figura 2.3: Schema a blocchi del dispositivo HCS 301

a seconda del sistema e delle misure che si intendono adottare:

- Metodo semplice: utilizza una chiave unica
- Metodo normale: utilizza il numero seriale del radiocomando per generare la chiave
- Metodo sicuro: utilizza il Seed per generare la chiave.

La soluzione del metodo semplice è la meno laboriosa ma meno sicura in quanto, se un radiocomando viene analizzato e da esso estratta la chiave, tutti i

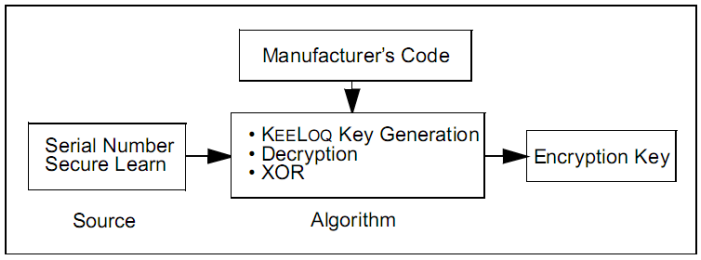

Figura 2.4: Schema di generazione delle chiavi dispositivo

radiocomandi programmati in questo modo vengono a perdere la loro chiave segreta e quindi la loro sicurezza.

Il metodo normale e il metodo sicuro prevengono questa eventualità gene-

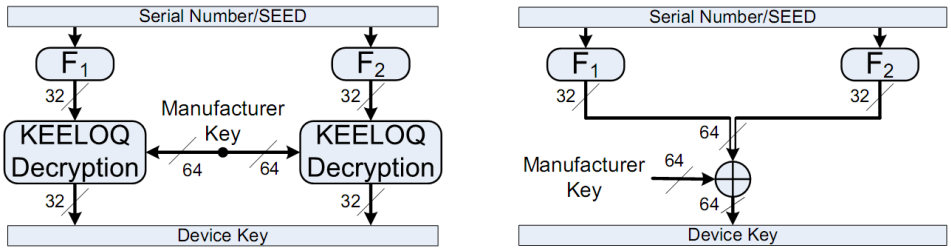

Figura 2.5: Schema di generazione chiavi dispositivo dalla chiave costruttore

rando una chiave diversa per ogni radiocomando. Con il metodo normale la chiave viene generata a partire dal numero seriale che è presente in chiaro in ogni trasmissione. In questo caso si viene a conoscenza delle informazioni da cui l'algoritmo di generazione parte per costruire la chiave. In molti casi, se l'algoritmo non è molto sicuro può essere pericoloso poichè si potrebbe ricostruire la chiave del produttore, ammesso che si sia venuti in possesso anche della chiave del dispositivo. Il metodo sicuro si differenzia per questa ultima parte, infatti il seed è una parola fissa di 32 bit che viene trasmessa al posto dei 32 bit cifrati della normale trasmissione durante la procedura di apprendimento del radiocomando nella parte ricevente. In questo modo, anche se è nota la chiave del

dispositivo, non è possibile recuperare la trasmissione del seed così facilmente se non si è in possesso del radiocomando, a differenza di prima in cui il seriale poteva essere preso in chiaro durante ogni utilizzo del radiocomando.
Durante la fase di apprendimento, la stazione ricevente si ricava la chiave dispositivo dal seed o dal numero seriale. Fatto questo, decodifica una trasmissione controllando il valore del discriminatore e verificando se è valido. In tal caso, la parte ricevente riconosce di aver calcolato la chiave correttamente e la memorizza assieme al seriale del dispositivo e al suo contatore progressivo.
Nell'utilizzo del trasmettitore, ad ogni pressione su un pulsante viene trasmesso un preambolo, un header e successivamente 66 bit, che comprendono una parte cifrata di 32 bit e una parte in chiaro di 34 bit. Come si vede dalla Fig. 2.6, la parte in chiaro prevede un bit di Repeat, un bit di Vlow, 4 bit di Button

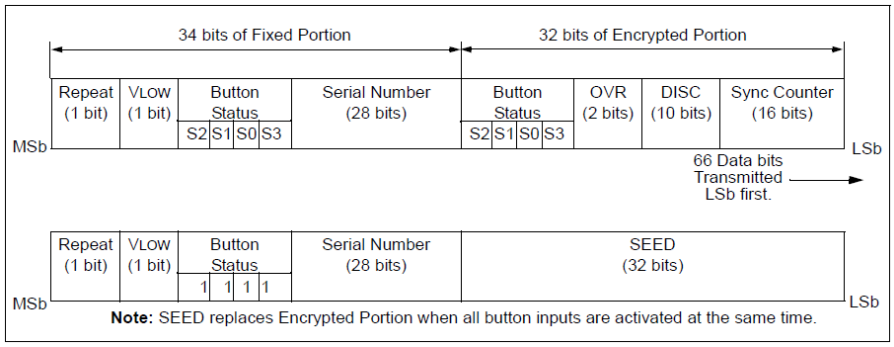

Figura 2.6: Organizzazione del pacchetto trasmesso

Status e il Serial Number di 28 bit. Il bit di Repeat viene tenuto basso solo alla prima trasmissione dello stesso codice e alto alle successive. Il bit di Vlow viene trasmesso alto quando la tensione del trasmettitore è bassa cosicchè la stazione ricevente possa informare l'utilizzatore che il radiocomando necessita di cambiare le batterie. Il Button status indica il pulsante che è stato premuto e il serial number, che è il numero seriale univoco del radiocomando. La parte cifrata contiene al suo interno lo stato dei pulsanti premuti, come nella parte in chiaro, i bit di overflow, che solitamente sono settati a 1 e possono servire ad estendere il range dei valori del contatore qualora non fossero sufficienti,

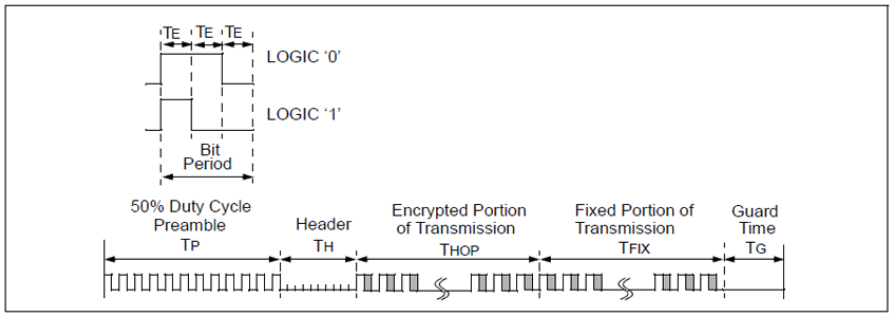

Figura 2.7: Formato di codifica di trasmissione dei bit

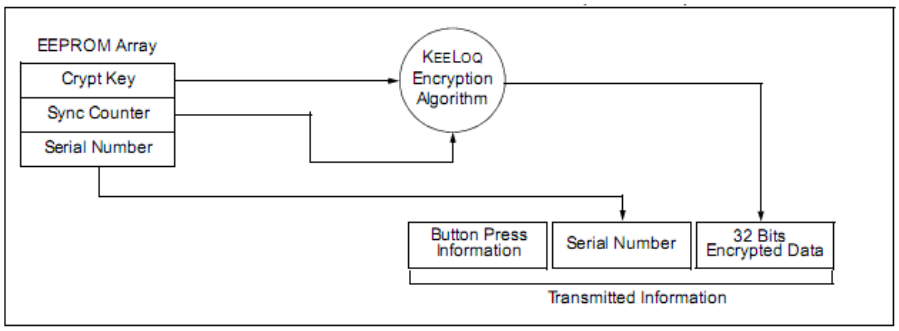

Figura 2.8: Schema di generazione del messaggio trasmesso

e il valore discriminante. Questo valore è solitamente uguale ai 10 bit meno significativi del numero seriale ed è utilizzato dal ricevitore per verificare la decodifica. Il discriminante viene solitamente confrontato con quello salvato in memoria dalla parte ricevente durante l'apprendimento, viene usato per verificare che la decodifica sia corretta e che la trasmissione ricevuta contenga informazioni valide e correttamente decifrate. Gli ultimi 16 bit della trasmissione sono quelli del contatore progressivo (Synchronization Counter o Sync Counter). Questi ultimi 32 bit sono trasmessi cifrati e solo dopo la decodifica sarà possibile ottenere le informazioni contenute in questi bit. Il Keeloq facilita il mascheramento delle informazioni in quanto, mentre da una trasmissione all'altra cambia solitamente solo qualche bit, la trasmissione dopo la cifratura

cambia per più del 50 % rispetto alla precedente, rendendo impossibile un'associazione diretta tra i bit cifrati e quelli in chiaro.
Durante il normale funzionamento di un sistema, formato per semplicità da un radiocomando e da una stazione ricevente, ad ogni pressione di un pulsante del radiocomando viene inviata la trasmissione e successivamente incrementato il contatore (Sync counter). Quando alla stazione ricevente arriva un messaggio,

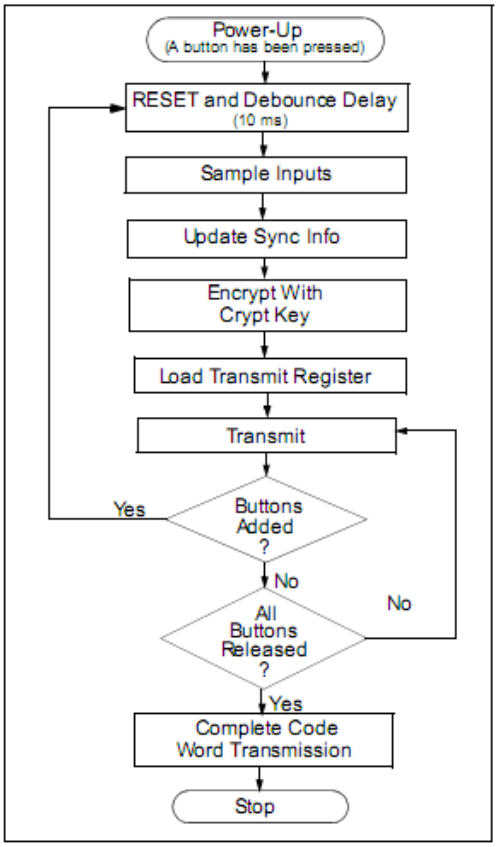

Figura 2.9: Schema del ciclo di trasmissione di un trasmettitore

essa ne decodifica la parte cifrata. Questa viene confrontata con il valore del contatore memorizzato e viene valutato in quale parte si trova della cosiddetta finestra di sincronizzazione. Quest'ultima si può vedere in Fig. 2.10 è divisa in tre aree: una parte a singola operazione, una parte a doppia operazione e

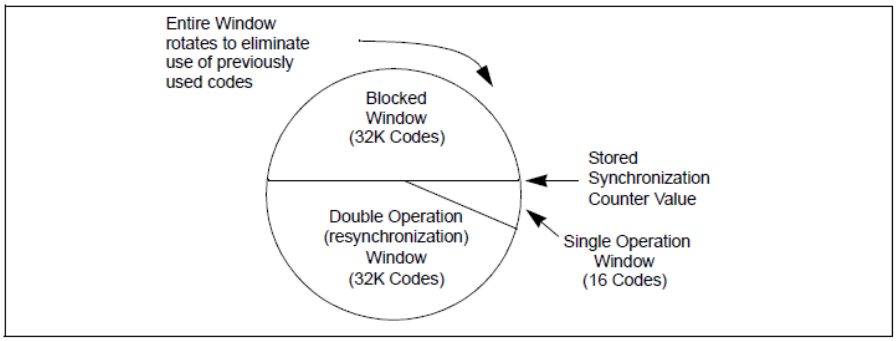

Figura 2.10: Finestra di sincronizzazione dei codici seriali

una parte bloccata. L'ampiezza di queste viene suggerita da Microchip, tuttavia può essere modificata a seconda delle esigenze del progettista. La parte a singola operazione comprende i 16 o più codici successivi a quello memorizzato nella parte ricevente. Questi codici sono quelli validi per attivare il comando inviato dal trasmettitore senza ulteriori verifiche. Può capitare infatti che il radiocomando venga premuto erroneamente facendo avanzare così il contatore. Nella finestra a doppia operazione, anche detta di risincronizzazione, parte dalla fine della parte a singola operazione e contiene circa la metà dei codici possibili (circa 32000). In questa finestra l'utente dovrà premere due volte il pulsante perchè la stazione ricevente verifichi l'identità del trasmettitore e si risincronizzi sul valore del contatore inviato dal radicomando. La restante zona di codici è una zona bloccata, ovvero comprende i restanti codici che la parte ricevente ignora. Ogni volta che viene ricevuto un codice valido la finestra di sincronizzazione ruota. In questo modo il codice appena ricevuto viene scartato, rientrando nella finestra dei codici già usati e quindi ignorati dalla parte ricevente. Perchè lo stesso codice possa venire riutilizzato la finestra di sincronizzazione dovrà ruotare compiere un intero giro (quindi 64000 volte) per riportare il codice nella finestra a singola operazione.

Capitolo 3

Le tipologie di attacco applicate al Keeloq

Benchè la progettazione dell'algoritmo Keeloq risalga ai primi anni Ottanta, la prima criptoanalisi venne pubblicata da Bogdanov solo nel febbraio 2007 [3]. Questo attacco, di tipo software, si basa su alcune proprietà del Keeloq che vedremo in seguito. Solitamente gli attacchi fatti su questi algoritmi mirano in genere a recuperare la chiave che utilizza il Keeloq per cifrare le comunicazioni. In molti casi si dispone sia del testo in chiaro, che di quello cifrato, quindi è già noto il contenuto del messaggio che non ha in sè grande valore. In queste applicazioni, in cui il messaggio cifrato è monouso, non è importante conoscerne il contenuto, ma emulare la cifratura per generare così le trasmissioni future. Per questo motivo è necessario conoscere la chiave che diventa l'obiettivo di tutti gli attacchi che descriveremo. Ottenere la chiave significa emulare perfettamente il trasmettitore e rendere inefficiente la sicurezza del sistema.
Gli attacchi fatti sul Keeloq si possono dividere in due grandi categorie, che vanno a concentrarsi su aspetti diversi del sistema Keeloq: da una parte abbiamo gli attacchi software che sfruttano le proprietà di costruzione dell'algoritmo nel tentativo di decodificarlo più o meno efficacemente. Dall'altra abbiamo un attacco di tipo hardware che lavora sull'implementazione fisica dell'algoritmo. Dallo studio del dispositivo fisico e dal suo funzionamento cerca di ricavare la

chiave.

3.1 Attacchi software

Una tabella riassuntiva degli attacchi software praticati è presente in Fig. 3.1

Attack Type	Complexity		
	Data	Time	Memory
Time-Memory Trade-Off	2 CP	$2^{42.7}$	≈ 100 TB
Slide/Algebraic	2^{16} KP	$2^{65.4}$	?
Slide/Algebraic	2^{16} KP	$2^{51.4}$	?
Slide/Guess-and-Determine	2^{32} KP	2^{52}	16 GB
Slide/Guess-and-Determine	2^{32} KP	$2^{50.6}$	16 GB
Slide/Cycle Structure	2^{32} KP	$2^{39.4}$	16.5 GB
Slide/Cycle/Guess-and-Det.[a]	2^{32} KP	(2^{37})	16.5 GB
Slide/Fixed Points	2^{32} KP	2^{27}	>16 GB
Slide/Meet-in-the-Middle	2^{16} KP	$2^{45.0}$	≈ 2 MB
Slide/Meet-in-the-Middle	2^{16} KP	$2^{44.5}$	≈ 3 MB
Slide/Meet-in-the-Middle	2^{16} CP	$2^{44.5}$	≈ 2 MB
Time-Memory-Data Trade-Off	68 CP, 34 RK	$2^{39.3}$	≈ 10 TB
Related Key	66 CP, 34 $RK^{\ggg}$	negligible	negligible
Related Key	512 CP, 2 $RK^{\ggg}$	2^{32}	negligible
Related Key/Slide/MitM	2^{17} CP, 2 $RK^{\oplus}$	$2^{41.9}$	≈ 16 MB

Time complexities are expressed in full KeeLoq encryptions (528 rounds).
KP: known plaintexts; CP: chosen plaintexts
$RK^{\ggg}$: related keys (by rotation); $RK^{\oplus}$: related keys (flip LSB)

[a] The time complexity for this attack is based on very unrealistic memory latency assumptions and hence will be much higher in practice.

Figura 3.1: Una panoramica degli attacchi software conosciuti sul Keeloq

Il primo attacco, costruito da Bogdanov, ha una complessità temporale di 2^{52} cicli di criptatura Keeloq, utilizza 16 GB di memoria e richiede l'intero codebook ovvero 2^{32} coppie chiaro-cifrato. Questo attacco si basa sulla tecnica slide [4, 5] e su un'approssimazione lineare della funzione Booleana di feedback [3].

Courtis et al. nel loro lavoro hanno applicato tecniche algebriche per fare la criptoanalisi del Keeloq: riportano nei loro studi alcuni attacchi slide-algebric andati a buon fine. Un attacco algebrico può andare a buon fine mediamente in 2.9 secondi, quando viene fornita una slid pair. Poiché però non c'è una

possibilità di identificare a priori una slid pair dall'insieme dei campioni, l'attacco viene ripetuto 2^{32} volte, una per ogni coppia generata con i 2^{16} elementi del codebook. Gli autori descrivono anche un possibile attacco che necessita dell'intero codebook (2^{32}), richiedendo una complessità temporale di 2^{27} cicli di criptatura, tuttavia con una probabilità di successo del 44 %. [6]
Una versione aggiornata dell'attacco di Bogdanov è pubblicata in [1]. In questo lavoro la complessità temporale (2^{50}) è diminuita di poco rispetto all'inizio e necessita ancora dell'intero codebook. In media gli attacchi si collocano su una fascia con complessità temporale attorno ai 2^{50} e con un utilizzo di risorse rilevanti oltre che all'utilizzo dell'intero codebook.
Un attacco innovativo, che pur avendo una complessità temporale non molto inferiore (2^{45}), ha un utilizzo minimo di memoria e utilizza soltanto metà codebook è quello di Indesteege et al. Questo attacco, come la maggior parte, si basa sempre sullo slide attack, tuttavia a differenza degli altri si basa su un nuovo approccio chiamato meet-in-the-middle che permette all'attacco di essere di facile implementazione e di prestarsi bene alla parallelizzazione. [7]
La caratteristica che più di tutte accomuna gli attacchi software finora accennati è che essi necessitano di avere il codebook per funzionare. La tabella delle coppie di parole da 32 bit in chiaro e cifrate, non è di facile reperibilità ed è causa di una forte limitazione sulle possibili applicazioni di questi attacchi.
I protocolli di autenticazione sviluppati sul Keeloq sono due: il protocollo - Keeloq Hopping Code e quello Keeloq Identifiy Friend or Foe (IFF). Entrambi utilizzano un contatore segreto di 16 bit per identificare il trasmettitore. Le differenze consistono nella modalità in cui il testo viene generato e criptato. Infatti nell'Hopping Code il testo in chiaro viene generato all'interno del trasmettitore, criptato e poi spedito alla parte ricevente per la decodifica e l'autenticazione. Il protocollo IFF invece è un semplice protocollo di interrogazione-risposta. In questa modalità la stazione ricevente spedisce il testo in chiaro al trasmettitore, questo lo codifica con la chiave e invia il testo cifrato alla parte ricevente. La decodifica che verrà fatta verificherà l'autenticità del trasmettitore e la correttezza della chiave.

La differenza sostanziale di questi due tipi di autenticazione è data dal fatto che nell'IFF il testo in chiaro viene trasmesso come pure quello cifrato. In questo modo possono essere raccolti entrambi e creare il codebook. Questo discrimina i casi in cui gli attacchi software possono essere effettuati, infatti con l'Hopping Code abbiamo a disposizione soltanto il testo cifrato, il che impedisce di avere il dizionario cifrato-in chiaro necessario per gli attacchi software.

3.2 Attacchi hardware

Quando parliamo di dispositivi crittografici intendiamo dispositivi che implementano un algoritmo di crittografia e conservano al loro interno la chiave crittografica. Riuscire ad entrare in un dispositivo crittografico significa estrarre la chiave del dispositivo. Gli attacchi a questi dispositivi vanno a valutare le caratteristiche degli stessi per estrarre la chiave utilizzando le svariate tecniche sviluppate nel tempo. Un primo criterio di classificazione degli attacchi è capire quando un attacco è attivo o passivo:

- Attacchi passivi: in un attacco passivo il dispositivo crittografico lavora in parte o totalmente nel modo normale di funzionamento. La chiave è rivelata tramite l'osservazione delle proprietà fisiche come ad esempio il tempo di esecuzione o il consumo di potenza.

- Attacchi attivi: in questo attacco gli input o il contesto di funzionamento sono manipolati per ottenere un comportamento anomalo del dispositivo o per manometterlo affinchè il funzionamento riveli informazioni sulla chiave.

Il secondo criterio per suddividere gli attacchi è in base alle interfacce che vengono sfruttate nell'attacco. I dispositivi crittografici hanno diverse interfacce fisiche e logiche. Ad alcune di queste interfacce si può aver accesso facilmente, mentre altre possono essere usate solo con attrezzature speciali. A seconda delle interfacce usate si possono distinguere attacchi invasivi e non invasivi. Tutti questi attacchi possono essere passivi o attivi.

- Attacchi invasivi: un attacco invasivo è l'attacco più forte che può essere fatto contro un dispositivo crittografico. In un attacco di questo tipo non ci sono limiti su quello che si può fare sul dispositivo per rivelare la sua chiave segreta. Un attacco invasivo inizia tipicamente con il depackaging del dispositivo. Successivamente, si accede a diversi componenti con una stazione di sondaggio. Questa parte dell'attacco è passiva solo se la sonda è usata per misurare i segnali dei dati, ad esempio in un bus. L'attacco diventa attivo se si utilizzano le sonde per alterare i valori e di conseguenza la funzionalità del chip. Questa tipologia di attacchi è molto potente, tuttavia necessita di un equipaggiamento tipicamente costoso e quindi non viene praticata molto spesso.

- Attacchi semi-invasivi: negli attacchi semi-invasivi il dispositivo viene comunque tolto dal package, tuttavia non viene effettuato nessun contatto elettrico con il chip cosicchè lo strato di passivazione rimane intatto. L'obiettivo di questo attacco è di leggere il contenuto di una cella di memoria senza usare il normale circuito di lettura. Per questo gli attacchi attivi mirano a causare un errore nel dispositivo tramite fonti elettromagnetiche o di raggi X. Le apparecchiature richieste da questi attacchi non sono molto costose, tuttavia lo sforzo per portare a termine tutto l'attacco è piuttosto alto e trovare i punti in cui effettuare l'attacco richiede tempo e una notevole esperienza.

- Attacchi non invasivi: in attacchi non invasivi il dispositivo è essenzialmente analizzato senza modificarlo, usando le interfacce che sono direttamente disponibili. Il dispositivo non è alterato permanentemente e in seguito non vi resta traccia dell'attacco. La maggior parte di questi attacchi può esser fatta con equipaggiamenti poco costosi e per questo pongono un reale problema di sicurezza su questi dispositivi. In particolare gli attacchi passivi e non invasivi hanno ricevuto molte attenzioni durante gli ultimi anni. Questi attacchi si riferiscono solitamente ad attacchi di tipo side-channel. I tre tipi più importanti sono: attacchi timing, attacchi power analysis e attacchi elettromagnetici (EM). Il concetto alla base di

questi tre attacchi è quello di determinare la chiave segreta di questo dispositivo misurando il suo tempo di esecuzione, il suo consumo di potenza o il suo campo elettromagnetico. Ci sono inoltre alcuni casi di attacchi side-channel attivi che hanno come obiettivo quello di generare un errore nel chip senza toglierlo dal packing. Questi errori possono essere causati da glitch del clock, da glitch di potenza o dal cambiamento della temperatura dell'ambiente.

Nella letteratura sono riportati molti attacchi praticati sul Keeloq, la maggior parte di tipo software. Questa tipologia di attacchi è tuttavia adatta al protocollo IFF del Keeloq e non applicabile quindi alla modalità code hopping che risulta la più diffusa nelle applicazioni commerciali e nel caso dei radiocomandi per automazioni, come trattato in questo lavoro. Per questo ci focalizzeremo su un attacco di tipo power analysis.

Gli attacchi che utilizzano la power analysis hanno ottenuto molte attenzioni poichè sono molto potenti e si possono praticare in maniera relativamente semplice. Proprio per questo pongono un problema rilevante sulla sicurezza dei dispositivi crittografici. Oggigiorno per lo sviluppo e l'implementazione dei moderni dispositivi crittografici è molto importante conoscere la power analysis per adottare le giuste contromisure. I dispositivi non protetti possono essere penetrati con il minimo sforzo.

Come già accennato, il principio base di un attacco che utilizza la power analysis si basa sul fatto che ci sono delle informazioni relative al funzionamento del dispositivo crittografico che sono a disposizione di chi pratica l'attacco (Fig. 3.2). Infatti nel consumo di potenza si possono riconoscere due tipi di informazioni: quelle dipendenti dai dati elaborati e quelle dipendenti dalle operazioni effettuate. Poichè i dispositivi sono costituiti da transistor, che agiscono come interruttori controllati in tensione, l'azionamento di questi provoca un flusso di cariche che è modulato in base alle operazioni svolte. La potenza consumata dal dispositivo può essere quindi direttamente collegata al funzionamento interno.

Negli attacchi Simple Power Analysis (SPA), l'attaccante osserva direttamente il consumo di potenza del sistema. L'ammontare dell'energia consumata

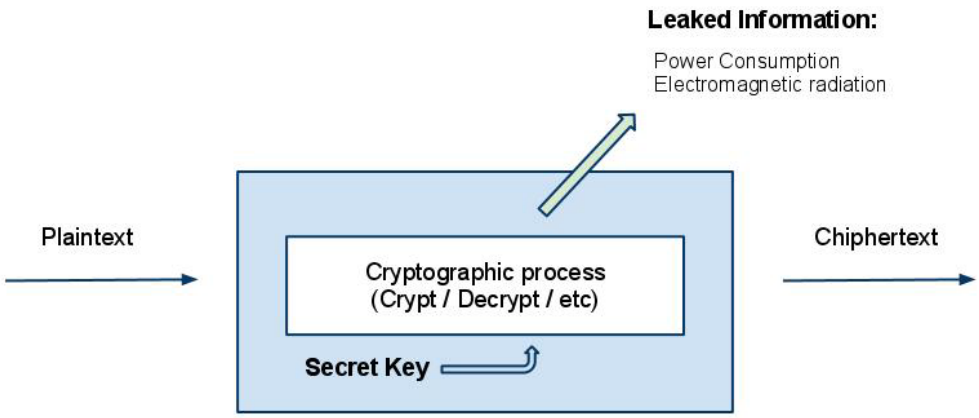

Figura 3.2: Informazioni disponibili su un dispositivo crittografico

varia in base alle istruzioni eseguite nel microprocessore. Nella traccia possono venir identificate macro aree relative ai differenti blocchi di operazioni, come i vari cicli di Data Encryption Standard (DES) o operazioni di RSA (algoritmi molto studiati e analizzati con la power analysis), poichè il funzionamento del processore cambia significativamente durante le diverse parti di elaborazione di questi algoritmi. Ingrandendo la traccia possono essere identificate le singole operazioni. Per esempio nelle analisi dell'RSA con un attacco SPA vengono lette nella traccia parti diverse che rivelano le differenti operazioni di moltiplicazione ed elevamento al quadrato. Tuttavia molti dispositivi crittografici sono stati resi immuni ad attacchi SPA senza troppe difficoltà.

La parte alta della Fig. 3.3 mostra tutti i 16 cicli di crittografica DES e la permutazione finale. La traccia in basso è una vista dettagliata del secondo e terzo ciclo di DES dove si possono distinguere le operazioni all'interno dei singoli cicli.

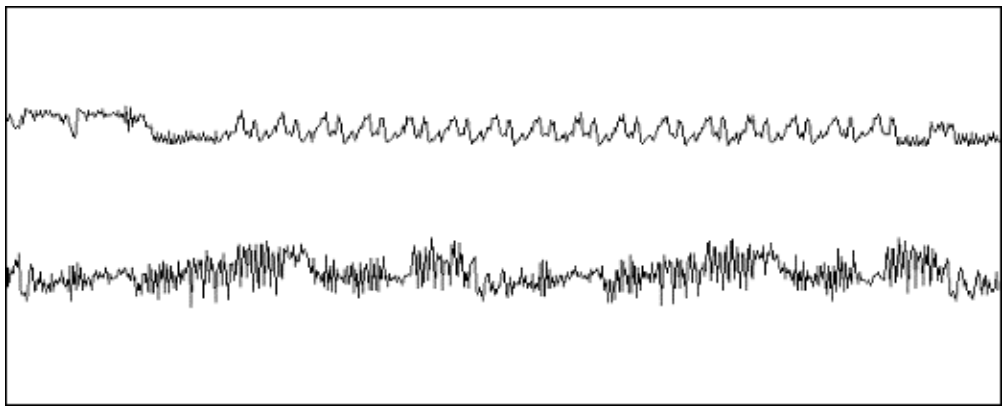

Figura 3.3: Scansione del consumo di potenza di una smart card in un ciclo di crittografia DES

3.3 Attacchi DPA: una descrizione generale

Gli attacchi che usano la Differential Power Analysis (DPA) sono molto più potenti di quelli SPA e più difficili da prevenire. Mentre nell'SPA viene effettuata un'indagine visiva sulla traccia di potenza per identificarne le differenti fluttuazioni, l'attacco DPA utilizza un'analisi statistica e una tecnica di correzione di errore per estrarre le informazioni relative alla chiave segreta. Nelle prime fasi di una DPA vengono acquisite ed allineate le tracce di potenza mentre il dispositivo sotto attacco sta svolgendo una codifica. In aggiunta bisogna registrare o il testo cifrato in uscita dal dispositivo di cifratura o il testo in chiaro in ingresso, dipendentemente da quale di questi due si può venire a conoscenza. Successivamente bisogna individuare nelle tracce dei punti intermedi fissi che siano dipendenti dalla chiave, che cambiano al variare del testo in chiaro che viene fatto elaborare al dispositivo. Questi punti intermedi devono anche poter essere calcolati combinando la parte della chiave che stiamo cercando e il testo in chiaro o cifrato che abbiamo registrato. Supponiamo infatti che, quando vengono processati questi valori, alcune informazioni riguardanti la cifratura possano essere rilevate nel consumo di potenza che viene registrato. La parte successiva dell'attacco prevede la costruzione di un modello che stimi il consu-

mo di potenza del dispositivo in base alla chiave che dobbiamo trovare. Questo modello può essere costruito in vari modi, i più utilizzati e più efficienti sono quelli che vanno a valutare la distanza di Hamming o il peso di Hamming. Il modello andrà a calcolare i valori associati ai punti intermedi delle tracce di potenza cambiando la parte della chiave che stiamo valutando. In questo modo per ogni combinazione dei bit della chiave (se valuto 8 bit avrò 256 combinazioni diverse) avrò un valore corrispondente e relativo al consumo di potenza del dispositivo. Ovviamente, quando la chiave inserita nel modello è corretta, i valori risultanti saranno maggiormente correlati alle tracce di potenza registrate. In questo modo si troveranno i valori corretti della chiave. Successivamente valuteremo in maniera più dettagliata i vari passaggi per condurre un efficiente attacco DPA.

Capitolo 4

Power analysis

4.1 Descrizione dell'attacco

Vediamo ora in dettaglio la strategia di attacco che è utilizzata per tutti i DPA. Essa consiste di cinque passi.

4.1.1 Passo 1: Scelta di un valore intermedio dell'esecuzione dell'argoritmo

Il primo passo in un attacco DPA consiste nel scegliere un valore intermedio dell'algoritmo di crittografia che è eseguito nel dispositivo. Questo valore deve essere una funzione $f(d, k)$, dove d rappresenta i dati noti che vengono elaborati e k una piccola parte della chiave. I risultati intermedi che hanno queste proprietà possono essere utilizzati per rivelare la chiave k. Nella maggior parte dei casi d è il testo in chiaro o cifrato.

4.1.2 Passo 2: Misurazione del consumo di potenza e allineamento tracce

Il secondo passo è misurare il consumo di potenza del dispositivo mentre cripta o decripta differenti D blocchi di dati, dove D è il numero di tracce acquisite. Per ognuna di queste tracce acquisite l'attaccante deve registrare il valore d che viene coinvolto nel calcolo dei punti intermedi scelti al passo 1. Scriviamo

quindi questi valori in un vettore $\mathbf{d} = (d_1, ..., d_D)^{'}$, dove d_i denota il testo cifrato o in chiaro all'i-esimo ciclo di codifica o decodifica.
Durante ognuno di questi cicli l'attaccante registra una power trace. Ogni traccia corrisponde al proprio testo e al proprio interno contiene T punti intermedi. La traccia sarà messa all'interno del vettore $\mathbf{t_i} = (d_{i,1}, ..., d_{i,T})^{'}$. Le tracce possono essere scritte in una matrice $\mathbf{T}$ di dimensioni $D \times T$ assieme ai relativi d_i. Affinchè l'attacco abbia successo vi è la necessità di allineare correttamente le tracce. L'allineamento è particolarmente importante in quanto c'è bisogno che i punti intermedi siano correttamente collocati nella matrice dei valori. Per fare questo, è necessario che ci sia un segnale di trigger che permetta di registrare sempre la stessa porzione di traccia durante un ciclo di codifica o decodifica. Se non è possibile un allineamento con un trigger ben definito, bisogna allineare le tracce tramite punti fissi all'interno delle stesse. Ad esempio, se il ciclo viene preceduto o succeduto da sequenze invarianti e ben distinguibili di operazioni, che generano un ben preciso consumo di potenza, si utilizzerà questa parte di traccia per allineare, in una post elaborazione, i punti intermedi di ogni acquisizione.

4.1.3 Passo 3: Calcolo dei valori intermedi ipotetici

Questo passo consiste nel calcolare gli ipotetici valori intermedi per ogni possibile scelta di k. Queste le scriviamo nel vettore $\mathbf{k} = (k_1, ..., k_K)$ dove K è il possibile numero di scelte della sottochiave k. In questo contesto, gli elementi di $\mathbf{k}$ li chiameremo chiavi ipotizzate. Dato il vettore $\mathbf{d}$ e il vettore $\mathbf{k}$, un attaccante può facilmente calcolare i valori intermedi $f(d, k)$ per tutti i D cicli di codifica e per tutte le K chiavi ipotizzate. Questo calcolo porta a creare una matrice $\mathbf{V}$ di dimensioni $D \times K$

$$v_{i,j} = f(d_i, k_j).$$

Le colonne j di $\mathbf{V}$ contengono i valori intermedi che sono calcolati in base alla chiave ipotizzata k_j. Una di queste chiavi ipotizzate è quella utilizzata dal dispositivo e quindi quella corretta. Lo scopo dell'attacco sarà quello di

identificare quale di queste colonne è stata processata nel ciclo di codifica o decodifica del dispositivo sotto attacco. L'indice che utilizzeremo per indicare la chiave corretta sarà k_{ck}.

4.1.4 Passo 4: Mappatura dei valori intermedi sui valori di consumo di potenza

In questo passaggio, l'attaccante mappa i valori ipotetici dei punti intermedi sui valori di potenza misurati. Per fare questo è necessario simulare il consumo di potenza mediante appositi software [8, Section 3.3] che elaborano i valori intermedi calcolati. In questo modo si ottiene una traccia simulata $h_{i,j}$ che verrà inserita nella matrice **H**. La qualità della simulazione dipende fortemente dalla conoscenza che l'attaccante ha del dispositivo analizzato. Più la simulazione è fedele alla realtà, più il DPA sarà efficiente.

4.1.5 Passo 5: Confronto tra i valori ipotetici di consumo con le tracce di potenza

Dopo aver mappato **H** in **V** si può procedere con l'ultimo passo del DPA. Qui, ogni colonna h_i della matrice **H** è confrontata con ogni colonna di t_j della matrice **T**. Questo significa che chi attacca mette a confronto i valori del consumo simulato di potenza, relativo ad ogni chiave ipotizzata, con le tracce acquisite. Il risultato della comparazione produce la matrice **R** di dimensioni $K \times T$ dove ogni elemento $r_{i,j}$ contiene la comparazione tra le colonne h_i e t_j. Gli algoritmi di confronto possono essere molteplici, noi utilizzeremo quello basato sulla correlazione. Vi è comunque una proprietà comune tra tutti gli algoritmi utilizzati: più alto è il valore $r_{i,j}$, più le colonne h_i e t_j sono simili. Da questa considerazione si può risalire alla chiave corretta.
Le differenti tracce acquisite corrispondono al consumo di potenza del dispositivo quando questo esegue un algoritmo crittografico, partendo da diversi dati in ingresso. I risultati intermedi scelti al punto 1 sono calcolati durante l'esecuzione dell'algoritmo, di conseguenza anche le tracce dipendono in alcuni punti da questi valori. Con t_{ct} ci riferiamo alla colonna della matrice **T** che contie-

ne i valori di consumo di potenza che dipendono dai valori intermedi calcolati con la chiave corretta (v_{ck}). Dalla traccia simulata h_{ck}, calcolata con la chiave ipotetica corretta k_{ck}, estraiamo i valori intermedi v_{ck}. I valori contenuti in h_{ck} e t_{ct} sono fortemente correlati poichè sono valori, da una parte simulati e dall'altra acquisiti, che dipendono dalla stessa chiave. Dal confronto di queste colonne si ottiene il valore massimo di **R**. Gli altri valori della matrice sono più bassi poichè confrontano colonne che risultano meno correlate. Nel caso in cui nessun valore si distingua tra tutti, può essere che l'attaccante non abbia raccolto sufficienti tracce per riuscire a stimare correttamente la relazione tra le matrici **H** e **T**. L'acquisizione di un maggior numero di tracce consente all'attaccante di poter misurare con più precisione la relazione tra le simulazioni e le tracce raccolte, determinando con evidenza la chiave corretta.

4.2 Modello della distanza di Hamming

Durante il calcolo dei valori intermedi è importante individuare il modello che più si avvicina al consumo di potenza che andremo successivamente a rilevare. Per questo motivo dobbiamo avere modelli che descrivono come il cambiamento dell'output all'interno dell'hardware è relazionato al consumo di potenza. Un modello generico che ben si adatta ai nostri scopi è il cosiddetto modello della distanza di Hamming. Questo modello è meno accurato di altri, ma si adatta meglio a molti dispositivi.
L'idea di base del modello di Hamming consiste nella conta delle transizioni $0 \rightarrow 1$ e $1 \rightarrow 0$ che avvengono in un circuito digitale durante un certo intervallo di tempo. Questo numero di transizioni è poi utilizzato per descrivere il consumo di potenza del circuito in questo intervallo di tempo. Dividendo l'intera simulazione in piccoli intervalli, possiamo rappresentare un'intera traccia di potenza con il numero di transizioni che si verificano in ogni intervallo di tempo. Quando si utilizza il modello di distanza di Hamming per simulare il consumo di potenza, vengono tenute in considerazione le seguenti ipotesi: la prima è che le transizioni $0 \rightarrow 1$ e $1 \rightarrow 0$ hanno lo stesso consumo di potenza. La seconda ipotesi è che le transizioni $1 \rightarrow 1$ e $0 \rightarrow 0$ contribuiscono equamente allo stesso

modo al consumo di potenza. Nel modello di Hamming, queste ultime transizioni valgono 0 poichè non cambiano il valore dei bit. Allo stesso modo, nel modello non viene considerato il consumo statico di potenza delle celle. Inoltre ogni cella contribuisce equamente al consumo di potenza, senza fare distinzioni tra le capacità parassite dei bus e quelle interne alla cella.
Per la sua semplicità, il modello della distanza di Hamming viene spesso utilizzato per le simulazioni di potenza. Queste simulazioni possono essere calcolate in maniera relativamente veloce. La distanza di Hamming di due valori v_0 e v_1 è il peso di Hamming (HW) di $v_0 \oplus v_1$. Tale peso consiste nel numero di bit che valgono 1 e quindi $HW = (v_0 \oplus v_1)$ corrisponde al numero di bit che differiscono tra v_0 e v_1.

4.3 Valutazione basata sulla correlazione

Quando si va a valutare il rapporto che intercorre tra le tracce simulate di potenza e quelle acquisite, bisogna tener conto del sistema di confronto che si va ad utilizzare. La correlazione risulta un buon modo per valutare la dipendenza tra le due campionature. Presi i vettori h_i dei consumi ipotizzati e t_i dei valori misurati, relativi allo stesso punto intermedio, abbiamo bisogno di valutare quanto le ipotesi di consumo siano congrue al consumo di potenza reale. In questo modo saremo in grado di riconoscere la correttezza della chiave. Per valutare questo legame ricorriamo alla correlazione che indica quanto due variabili cambiano allo stesso modo. Per definire la correlazione prendiamo due variabili aleatorie X e Y , i relativi valori attesi μ_X μ_Y e deviazioni standard σ_X σ_Y. La correlazione $\rho_{X,Y}$ viene definita come:

$$\rho_{X,Y} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}.$$

La correlazione risulta sempre essere compresa tra -1 e 1 ed indica: massima correlazione positiva quando è 1, massima correlazione negativa quando è -1 e incorrelazione quando è 0. Per questo bisogna valutare correttamente la correlazione con stesso modulo ma segno diverso. Infatti, nel nostro caso, la correlazione negativa portava a risultati non corretti per la valutazione della

chiave candidata. Volendo valutare soltanto i dati positivamente correlati, verranno tenuti in considerazione soltanto i risultati positivi compresi tra 0 e +1. I valori di correlazione negativa vengono ritenuti dei falsi positivi ovvero serie di dati che presentano un legame tra loro, ma che al fine pratico non sono rilevanti.
Quando non partiamo da una variabile aleatoria definita, ma da una data serie di misure come nel nostro caso, non possiamo conoscere il valore della correlazione a priori e così si utilizza uno stimatore per trovare la correlazione. Considerando X e Y una serie di n misure dove x_i e y_i sono gli elementi della serie con $i = 1, 2, 3...n$, la stima di $\rho_{X,Y}$ sarà:

$$r_{x,y} = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2 \sum_{i=1}^{N}(y_i - \bar{y})^2}}$$

dove $\bar{x}$ e $\bar{y}$ sono la media semplice delle due serie di misure.

4.4 La traccia di potenza

Un circuito digitale consuma potenza ogni qualvolta esegue un'operazione. Perciò il consumo di potenza di un dispositivo elettronico riveste un ruolo importante durante la progettazione dei circuiti stessi. Infatti, tale consumo determina la dissipazione di calore e il suo impiego in contesti dove la disponibilità di energia è limitata, come nei dispositivi a batteria. A maggior ragione nel nostro caso è fondamentale poichè su di esso basiamo il nostro attacco. La stretta relazione tra la struttura del dispositivo e il suo consumo porta ad un'analisi che rivela informazioni importanti e cruciali per la sicurezza stessa del device.
I circuiti costruiti in tecnologia CMOS hanno un consumo di potenza pari alla somma dei consumi delle singole celle di cui sono composti. Quindi, il consumo totale dipende essenzialmente dal numero di celle logiche e dalle loro interconnessioni. Questo è il risultato di decisioni di design prese a livello di sistema (architettura di sistema complessiva e algoritmi utilizzati), a livello di cella e a livello di transistor.
Quando un circuito CMOS è attivo, esso viene alimentato da una tensione

V_{DD} e le celle logiche presenti, quando elaborano i dati in ingresso, prelevano una corrente i_{DD}. Quindi avremo che il consumo di potenza istantaneo sarà $P = V_{DD} \cdot i_{DD}$ e sarà la somma di due contributi, il consumo di potenza statico P_{stat} e il consumo di potenza dinamico P_{dyn}. Il consumo di potenza statico è la parte di energia che la cella consuma quando non sono presenti commutazioni, ovvero quando la cella è alimentata, ma non sta facendo nulla. Il consumo di potenza dinamico invece è l'energia consumata dalla cella mentre svolge le operazioni. La parte statica del consumo di potenza è piuttosto piccola ed è la parte di corrente di perdita chiamata I_{leak} che fluisce da V_{DD} a massa pur avendo il transistor spento. La corrente di perdita è solitamente dell'ordine del pA, ma cresce e può diventare più rilevante con il diminuire delle lunghezze di gate dei transistor, sotto i 100 nm.

Il consumo di potenza dinamico è il fattore dominante nel consumo totale di potenza. Esso dipende dai dati che vengono processati nei circuiti CMOS ed è causato dalla variazione di stato della cella che avviene in una transazione $0 \rightarrow 1$ oppure $1 \rightarrow 0$. P_{dyn} è formata essenzialmente da due parti: la prima è l'energia necessaria a caricare le capacità di carico delle celle CMOS e la seconda l'energia di cortocircuito che fluisce, per un breve periodo di tempo, durante la commutazione della cella. Un altro contributo presente nel consumo di potenza dinamico è dato dai glitch. In una serie di stadi di celle collegate, la trasmissione del segnale di controllo da uno stadio all'altro può variare in base alle interconnessioni tra i vari stadi e le varie parti del circuito. Per questo, i segnali di controllo arrivano in tempi diversi all'ingresso della cella successiva. Questo comporta una parziale commutazione dell'uscita dello stadio finale dato da un disallineamento degli ingressi. Come si può vedere dalla Fig. 4.1 i glitches possono portare ad una temporanea ed errata variazione dello stato logico della cella ed una conseguente alterazione del consumo di potenza.

Durante l'esecuzione di un algoritmo crittografico, la commutazione delle celle logiche di un dispositivo avviene molto di frequente. Questa attività porta ad un notevole consumo che spesso non contiene tutte informazioni utili per l'attacco di potenza. Solitamente è il consumo di una piccola parte del dispositivo che

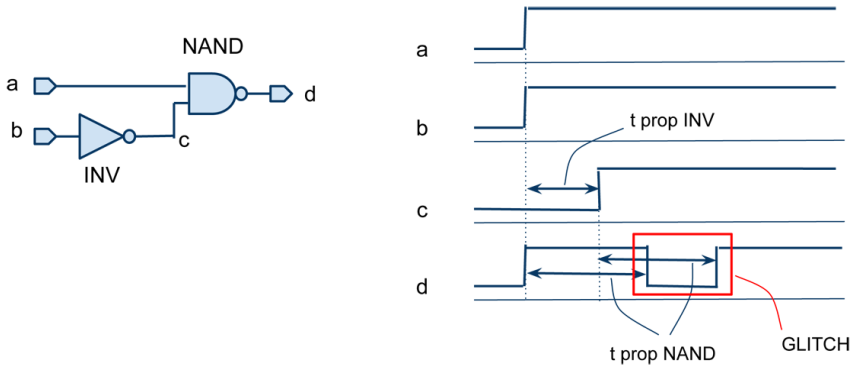

Figura 4.1: Glitch in una porta AND

permette all'attaccante di rivelare la chiave segreta. Il consumo di potenza di tutte le altre parti può costituire rumore dal punto di vista dell'attaccante. Un setup ideale per un attacco di potenza andrebbe a misurare soltanto la potenza di quella parte del dispositivo rilevante ai fini dell'attacco. In pratica questo risulta essere piuttosto difficile, ma si potrebbe portare a termine posizionando precisamente delle piccole sonde EM sulle parti rilevanti del dispositivo. Tuttavia, anche praticando questo tipo di tecnica, la traccia di potenza potrebbe contenere il contributo di celle non rilevanti ai fini dell'attacco, che si vanno a sovrapporre alla parte utile.

Quando viene raccolta una traccia di potenza, essa non è solamente formata dai contributi di potenza che abbiamo descritto, bensì vediamo come siano presenti delle fluttuazioni che possono arrivare a sovrastare il segnale utile che stiamo misurando. Analizzando diverse volte il consumo di potenza di una cella e mantenendo invariati gli ingressi, si possono notare delle variazioni nelle diverse tracce. Queste fluttuazioni sono causate dal rumore elettronico che può essere ridotto coi dovuti accorgimenti, ma non può essere rimosso completamente. Le sorgenti del rumore elettronico possono essere molteplici, alcune di queste possono essere controllate facilmente, mentre altre sono più difficili da gestire.
Di seguito viene presentato un elenco delle possibili sorgenti di rumore. Rumore

delle sorgenti di alimentazione: tutto il rumore prodotto dall'alimentazione del circuito va direttamente a deteriorare la traccia di potenza. Per questo motivo, è molto consigliato utilizzare una sorgente molto stabile per il dispositivo che si vuole studiare, evitando di alimentare direttamente dal PC o da RS232 e utilizzando alimentatori da banco o batterie. Più stabile è la sorgente di alimentazione migliore è il setup dell'esperimento.
Rumore del generatore di clock: il generatore di clock collegato al dispositivo testato deve essere stabile principalmente per due ragioni. Prima di tutto risulta essere importante per comparare le varie parti della traccie. Un disallineamento di queste può deteriorare in maniera significativa le prestazioni dell'attacco. Il secondo aspetto riguarda la stabilità dell'ampiezza del segnale di clock, poichè una fluttuazione troppo elevata si ripercuoterebbe sulla traccia di potenza.
Emissione condotta: l'emissione condotta dei componenti connessi direttamente al dispositivo sotto attacco porta del rumore sulla traccia di potenza, in modo particolare se i componenti sono montati sulla stessa Printed Circuit Board (PCB). Per questo motivo è consigliabile installare sulla stessa PCB il minor numero di componenti, possibilmente separando il dispositivo sotto test dal resto del circuito collegato al computer. E' buona norma utilizzare optoisolatori o isolatori magnetici per connettere le due parti della scheda, minimizzando così l'emissione condotta.
Emissioni irradiate: oltre alle emissioni condotte, altri contributi di rumore potrebbero arrivare dalle emissioni irradiate che possono essere ridotte con apposite schermature al setup di misura.
Rumore di quantizzazione: il rumore di quantizzazione è una conseguenza della conversione analogico-digitale effettuata dall'oscilloscopio. Più alta è la risoluzione dell'oscilloscopio, minore è il contributo del rumore di quantizzazione. Per gli attacchi di tipo power analysis sono sufficienti 8 bit di quantizzazione. Nel nostro caso, l'effetto della quantizzazione è molto inferiore a quello degli altri contributi di rumore.

4.5 Contromisure per gli attacchi DPA

Gli attacchi DPA si basano sul fatto che il consumo di potenza dei dispositivi crittografici dipende dai valori intermedi elaborati dall'algoritmo di crittografia. Appare chiaro dove bisogna porre l'attenzione per prevenire questo genere di attacchi: l'obiettivo delle contromisure sarà quello di rimuovere questa dipendenza e rendere così le tracce di potenza indipendenti da questi valori intermedi. Le contromisure che sono state pubblicate fino ad oggi possono essere divise in due grandi gruppi: contromisure di hiding e di masking. L'idea di base dell'hiding è quella di rimuovere la dipendenza dalle tracce relativa all'algoritmo. Questo richiede che l'esecuzione dell'algoritmo sia casuale oppure che venga cambiata in modo tale che chi attacca non riesca a risalire facilmente alla dipendenza dai dati processati. Per ottenere questo risultato, si può procedere in due modi diversi: progettare il device affinchè richieda approssimativamente sempre la stessa quantità di energia, oppure costruire il dispositivo in modo che il consumo sia pressoché casuale. In entrambi i casi si ottiene che la dipendenza del consumo di potenza dai dati elaborati risulta ridotta anche se non completamente. Inoltre bisogna tener conto che, nonostante l'hiding, i risultati intermedi elaborati sono gli stessi di un dispositivo non protetto, ma questa tecnica va soltanto ad alterare il consumo di potenza.
Nel caso del masking è diverso. L'idea base del masking è rendere casuali i valori intermedi elaborati dal dispositivo. In questo modo, i valori intermedi che vengono elaborati sono indipendenti dai reali valori intermedi che dovrebbero essere elaborati nello stesso istante. Il grande vantaggio del masking è che non modifica il consumo di potenza del dispositivo, che comunque può rimanere dipendente dai dati. Questa modifica viene fatta soltanto sul processo di crittografia (a livello hardware o software) e consiste nel calcolare in maniera casuale e non sequenziale i valori intermedi necessari.
A parte queste due contromisure c'è un principio generale che aiuta a prevenire gli attacchi basati sulla Power Analysis: la chiave usata nell'algoritmo crittografico deve essere cambiata più frequentemente possibile. Infatti cambiando più volte la chiave, risulta più difficile analizzare le tracce raccolte dipendenti

dalla stessa. Infatti, praticare un attacco DPA è più impegnativo se il numero di tracce acquisite è minore.

Capitolo 5

Applicazione attacco DPA

5.1 Modello per l'attacco DPA

La necessità di calcolare i valori intermedi, su cui confrontare il consumo di potenza, porta a cercare un modello adeguato per calcolarli. Consideriamo ora i vari contributi che le differenti parti dell'algoritmo portano nel consumo di potenza: la parte combinatoria, formata da XOR e dalla funzione non lineare, dà un contributo inferiore rispetto agli shift registers e per questo possiamo considerarla trascurabile nel calcolo della distanza di Hamming. Lo shift register della chiave dà anch'esso contributo nullo, poichè la sua distanza di Hamming non cambia in quanto la chiave è la stessa ma solo ruotata. In questo modo il registro della chiave porta approssimativamente ad una variazione nulla del consumo di potenza ad ogni ciclo di Keeloq. Per questo motivo ci concentreremo sullo shift register y dove è contenuta la trama. Il modello che utilizzeremo per questo attacco è:

$$P_{Hyp}^{(i)} = HD(y^i, y_{(i-1)}) = HW(y^{(i)} \oplus y^{(i-1)})$$

dove $P_{Hyp}^{(i)}$ è il consumo ipotetico di potenza al ciclo i-esimo di codifica (o decodifica), HD è la distanza di Hamming e HW è il peso di Hamming. $y^{(i)}$ indica il contenuto del registro al ciclo i-esimo di Keeloq e $\oplus$ indica la funzione XOR a 32 bit che andrà ad estrarre la differenza di bit tra il registro y allo stato

i-esimo e y al ciclo $i-1$. Il modello ovviamente funziona sia per la codifica che per la decodifica. In caso di codifica si partirà dal testo in chiaro e si valuterà il registro della trama nel cicli 1,2,3 ... mentre nella decodifica dal testo cifrato e si andranno a valutare i registri al ciclo 528-1, 528-2, ... Il modello basato sulla distanza di Hamming rappresenta molto bene il cambiamento di stato del nostro registro y. Infatti la distanza di Hamming del registro modellizza bene il consumo di potenza che si ha nel cambio di stato ad ogni ciclo di Keeloq. Infatti l'operazione di XOR che viene effettuata tra i due stati rivela soltanto le celle che vanno a cambiare il loro contenuto passando da $0 \rightarrow 1$ o $1 \rightarrow 0$. Le celle che non cambiano stato non consumano potenza e non vengono considerate.

5.2 Setup di misurazione

Per le nostre prove di attacco al ricevitore abbiamo utilizzato un Microchip HCS301 come target. Nella maggior parte delle applicazioni l'HCS301 si trova in package 8-pin SOIC, la cui configurazione standard si può vedere in Fig. 5.2.

Possiamo notare come l'integrato gestisca completamente tutte le funzionalità di un trasmettitore. Esso infatti gestisce il led di servizio, e alcuni pulsanti in ingresso e trasmette in uscita direttamente il segnale modulato per la parte radio. Ciò comporta che l'utilizzo dell'HCS301 non necessiti di altri integrati, ad esclusione della parte radio, per avere piene funzionalità in un sistema di trasmissione.

Per poter acquisire le tracce di potenza è necessario inserire una resistenza di shunt che procuri una caduta di tensione direttamente proporzionale al consumo di potenza dell'integrato. La resistenza deve avere un valore non troppo basso perchè la caduta non sia troppo piccola e quindi difficilmente rilevabile e misurabile, e non troppo grande per non alterare troppo il livello di ground a cui è connesso. Per questo il valore si aggira attorno ai 100 Ohm. Utilizzando uno shunt tutta la corrente che fluisce verso massa passa attraverso la resistenza. Ai capi di questa viene collegata la sonda dell'oscilloscopio che misurerà la traccia di potenza.

Oltre alla traccia di potenza è necessario registrare la parte cifrata in uscita

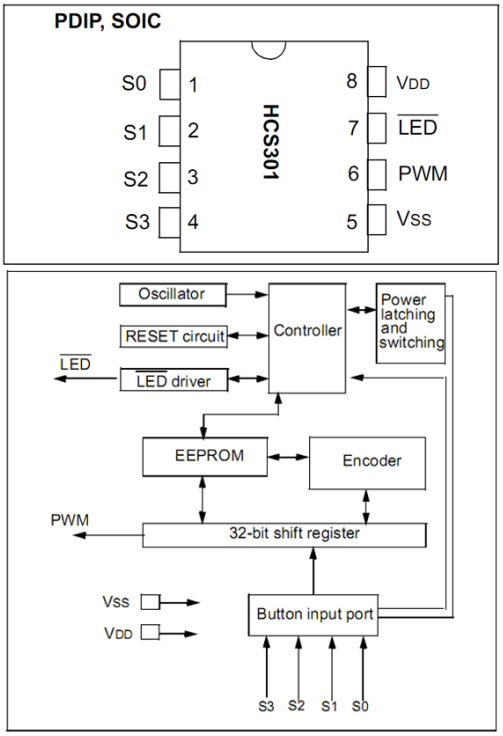

Figura 5.1: HCS301: Pinout e schema a blocchi del dispositivo

dalla codifica. Per fare ciò l'uscita PWM dell'HCS viene collegata ad un optoaccoppiatore che permette di trasmettere il segnale criptato come se venisse inviato alla parte radio, ma senza che una connessione non isolata possa deteriorare la traccia di potenza nella fase di elaborazione del Keeloq. Nel nostro setup la PCB contenente l'HCS era priva di ogni altro parte, di modo che, l'alimentazione non fosse degradata da fenomeni di emissione condotta o disturbi di altra natura. Come discusso in precedenza, l'alimentazione del circuito è di fondamentale importanza. Infatti, sia il rumore introdotto dall'alimentazione, sia le sue fluttuazioni si riflettono nella traccia di potenza e per questo abbiamo inizialmente alimentato il circuito con una batteria. Tuttavia la tensione non regolare ai capi della batteria, che dopo numerose prove non era più la stessa, ha portato ad una discrepanza nei valori misurati nelle power traces. Si è

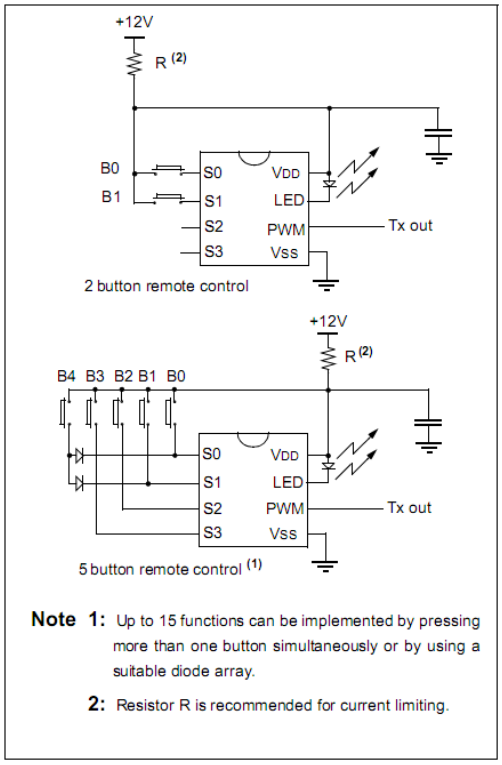

Figura 5.2: Configurazione standard HCS301

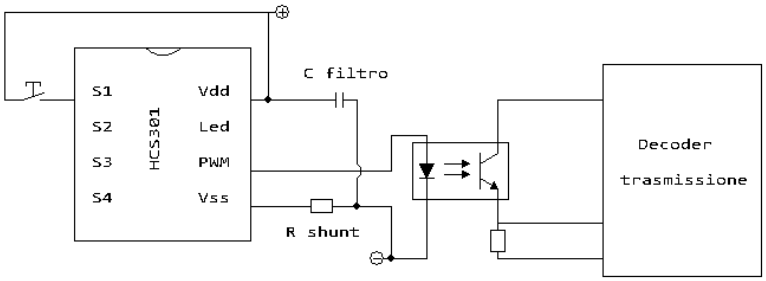

Figura 5.3: Setup di misurazione

visto come il progressivo calare della tensione di alimentazione della batteria portava ad una costante diminuzione del valore medio delle tracce. Questo pro-

duceva valori relativamente corretti all'interno della stessa traccia, ma sbagliati se confrontati tra diverse tracce. Per capire questo fenomeno abbiamo preso due tracce, ad esempio la prima e l'ultima di una serie di 30 misurazioni, e misurato i valori delle rispettive tracce in un dato istante. In entrambe le tracce abbiamo preso il valore medio e, relativamente a questo, abbiamo valutato se la misurazione risultava essere sopra o sotto la media. Dal confronto sono emersi gli errori introdotti dalla batteria. Nella prima traccia il valore misurato era inferiore alla media mentre nell'ultima traccia il valore era superiore alla media. Confrontando i valori tra loro, ovvero non rispetto al valore medio della traccia ma in valore assoluto, accadeva che il valore dell'ultima traccia era minore di quello della prima. In questo modo i valori presi allo stesso istante nelle varie misurazioni possono essere interpretati male e ottenere risultati scorretti. Questo è causato da un'offset dell'alimentazione che porta a variare la media dei valori e quindi a sbagliare la lettura dei valori assoluti. Per questo è importante avere un livello di alimentazione stabile, per non peggiorare la qualità delle prove. Un'altra soluzione consisterebbe nel valutare i valori dei punti intermedi e normalizzarli secondo la media, prima di confrontarli. In questo modo l'eventuale offset dell'alimentazione non influenzerebbe le misurazioni.

5.3 Acquisizione delle misurazioni

L'acquisizione delle tracce tramite l'oscilloscopio è un punto rilevante dell'attacco DPA. In questa fase bisogna registrare la caduta di tensione che si ha sullo shunt inserito sulla massa del dispositivo, mentre il dispositivo processa le informazioni. In questo caso siamo interessati a registrare la caduta di tensione quando il Keeloq elabora la trama e la codifica prima di trasmetterla. Una fase importante è trovare il trigger con cui avviare l'acquisizione dell'oscilloscopio. Per l'acquisizione delle tracce di potenza del trasmettitore è possibile fissare il trigger in molti punti. In prima analisi, il segnale del pulsante che avvia il processo di trasmissione è il miglior segnale di trigger utilizzabile. Tuttavia, da un'analisi più approfondita emerge che il pulsante non ha un circuito di de-

bounce hardware A causa di questo vengono introdotti molti fronti di trigger che possono cambiare l'allineamento delle tracce. Per questo si è scelto un tipo di trigger che rilevi un intervallo di tempo in cui il segnale resta sotto una certa soglia, come si vede in Fig. 5.4. In questo modo, tenendo conto che le parti macroscopiche della traccia di potenza sono uguali in tutte le misurazioni, è possibile avere un trigger stabile e un allineamento corretto su tutte le tracce.

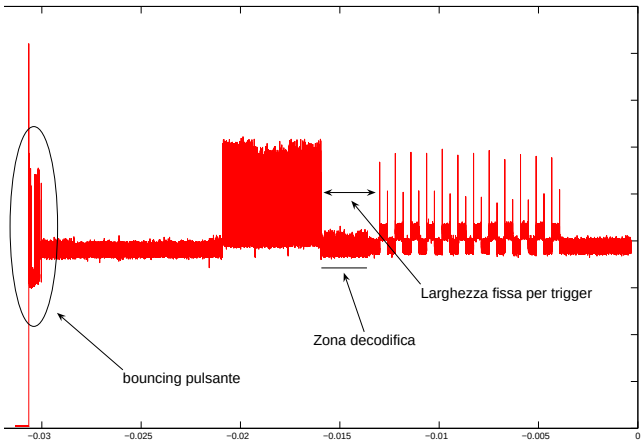

Figura 5.4: Posizionamento del trigger rispetto alla zona di decodifica. Il bouncing del pulsante non permette di avere un trigger stabile sul quel fronte di salita

Un altro parametro fondamentale è il sample rate che è condizionato da due fattori: il più importante è la dimensione di memoria dell'oscilloscopio e il secondo è il rapporto tra sample rate e numero di tracce necessarie per avere un attacco di successo. La dimensione di memoria impone un limite sul numero di campioni acquisibili dall'oscilloscopio. Questo limita la lunghezza temporale della traccia che può essere acquisita. Se il trigger è troppo lontano dal punto che dobbiamo visualizzare e abbiamo un alto sample rate, può accadere che l'oscilloscopio non visualizzi la porzione di traccia di nostro interesse poichè ha

già riempito la memoria con i punti precedenti. Per questo bisogna trovare un compromesso tra la memoria e il sample rate. Infatti, diminuendo i punti per secondo acquisiti, si può avere un intervallo più ampio e quindi visualizzare una parte più estesa della traccia di potenza. Tuttavia è importante notare che più il sample rate diminuisce, meno definita è la traccia di potenza e di conseguenza sono necessarie molte più tracce per poter avere un attacco di successo. Come si può vedere in Fig. 5.5

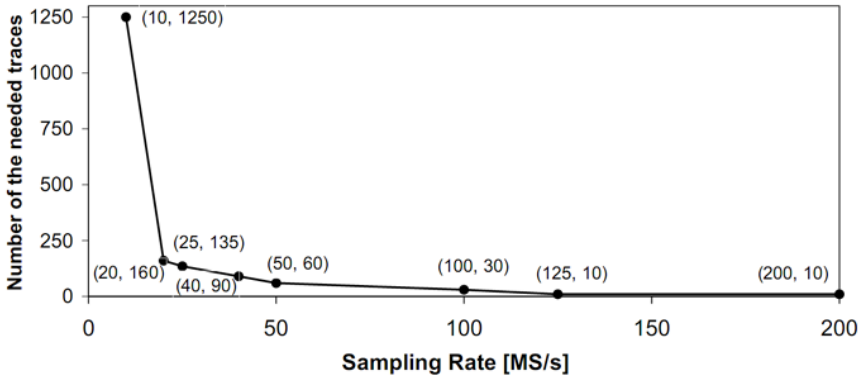

Figura 5.5: Relazione tra sample rate e tracce necessarie prima di avere successo nell'attacco [9]

le prestazioni dell'attacco calano drasticamente quando il sample rate scende sotto una certa soglia. Quindi è importante utilizzare un sample rate che permetta di acquisire un numero non troppo basso di tracce, ma allo stesso tempo non troppo alto per non incorrere nel problema di mancanza di memoria. Supponendo di avere una quantità di memoria illimitata e un oscilloscopio ideale potremmo aumentare a dismisura il sample rate. Tuttavia, le performance dell'attacco andrebbero a stabilizzarsi su un certo numero di tracce minime necessarie per portare a termine un buon attacco. Questo è dovuto al fatto che altri fattori di deterioramento del segnale, come il rumore, vanno ad incidere sulle performance più del sample rate. C'è anche da considerare che in molti casi è più conveniente non avere un sample rate troppo alto ed avere qualche

traccia in più, poichè nella post-elaborazione delle tracce avere troppi punti necessita di un'elevata capacità di calcolo e quindi di una minor efficienza.

5.4 Post-elaborazione

Nella fase di post elaborazione abbiamo raccolto un numero di tracce necessario per affrontare l'attacco, quindi ora è necessario processarle per estrarre i valori necessari. L'elaborazione delle tracce e l'estrazione dei valori intermedi necessari è stata fatta con Matlab tramite diversi script. La traccia viene importata in Matlab come una matrice di due colonne in cui una è il tempo e l'altra rappresenta il valore di tensione rilevato dall'oscilloscopio. In questa fase è importante capire come le tracce possano essere allineate ed elaborate in maniera automatica. La struttura delle tracce, come si vede in Fig. 5.6, permette di capire come lo script per l'estrazione dei dati debba agganciarsi

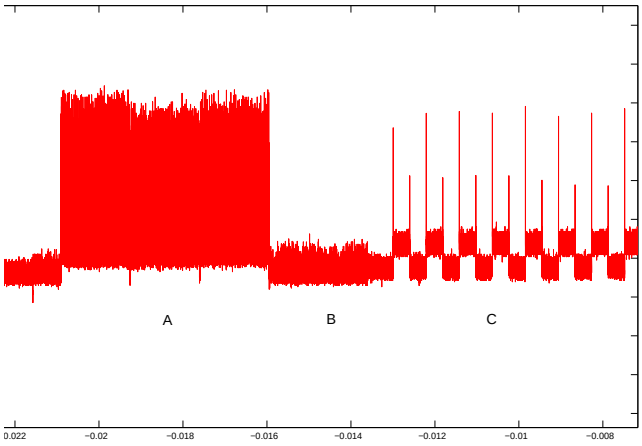

Figura 5.6: Zone della traccia di potenza acquisita. Zona A: lettura della eeprom, zona B: codifica del Keeloq, zona C: inizio preambolo trasmissione

all'inizio della parte centrale, zona B, per poi terminare l'estrazione prima che inizi la zona C.

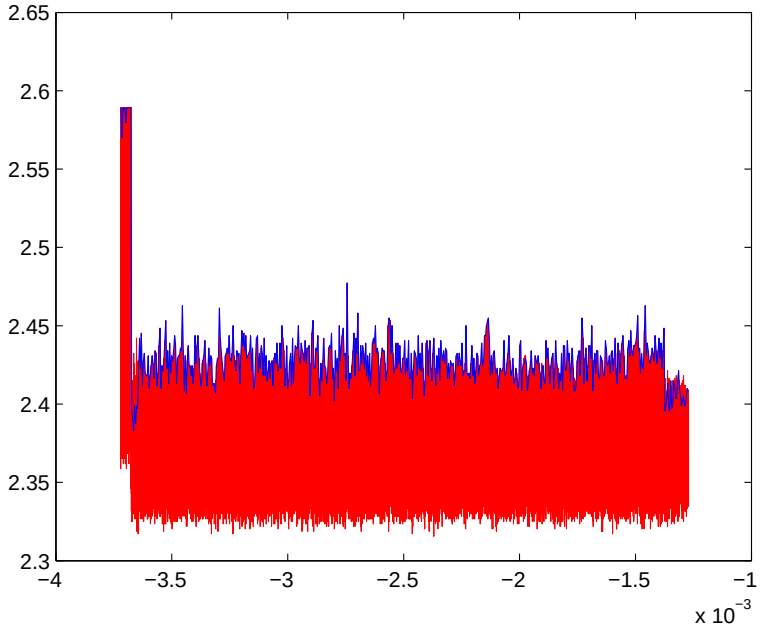

Figura 5.7: Traccia di potenza (colore rosso) e i picchi estratti dagli script automatici(colore blu). Lo script si aggancia correttamente alla traccia

5.5 Analisi tracce

Per poter estrarre le informazioni utili per l'attacco bisogna analizzare le tracce ed estrarre i valori dei punti intermedi. Come abbiamo visto in precedenza, i punti intermedi sono dei valori che dipendono dai dati conosciuti, dalla chiave e dalla trama in chiaro o cifrato. Nella nostra analisi delle tracce vediamo che questi punti possono non essere altro che il valore del registro Y che contiene la trama durante tutta la sua elaborazione. L'aggiornamento di questo registro, che viene effettuato ad ogni ciclo di codifica, quindi 528 volte, risulta perfetto per allocare i nostri punti intermedi poichè, dato il consumo di potenza rilevante, sono facilmente individuabili. Inoltre, essi garantiscono un'immunità alta ai falsi positivi: i valori elaborati in ognuno dei punti intermedi dipende da quelli elaborati ai passi precedenti, per la sequenzialità dell'algoritmo Keeloq.

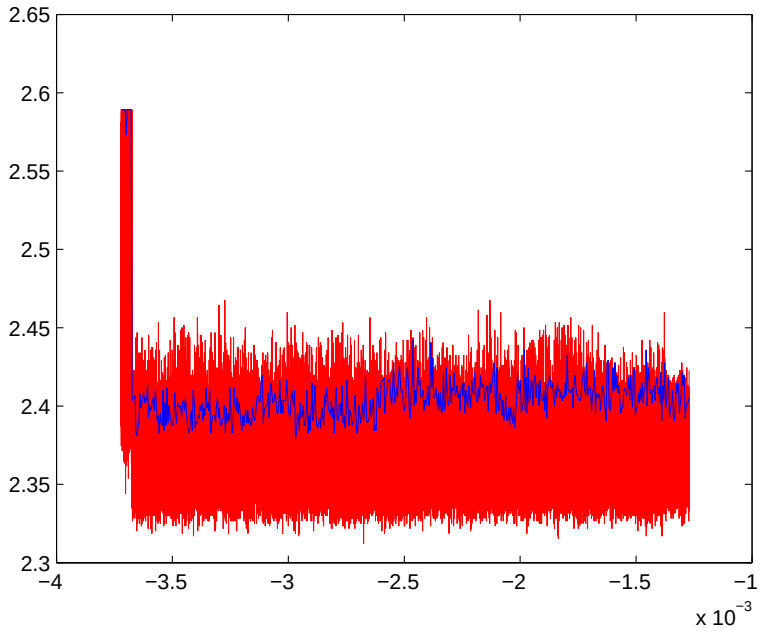

Figura 5.8: Traccia di potenza (colore rosso) e i picchi estratti dagli script automatici(colore blu). Lo script non si aggancia alla traccia e non riesce ad estrarre i picchi

Se ripercorriamo la codifica dall'ultimo ciclo (528) al primo, decodificando la nostra trama cifrata passo dopo passo, vediamo come i valori estratti nei primi cicli influenzano le decodifiche successive. Questo è dovuto al fatto che lo shift fa in modo che il registro della trama introduca nei cicli successivi i bit che ha elaborato in precedenza. Questi influenzeranno l'elaborazione creando così una dipendenza a catena di tutti i bit della trama. Per questo motivo, se durante il recupero dei primi bit della chiave c'è un'errore e vengono erroneamente considerati corretti dei falsi positivi, l'errore non resta latente, bensì impedisce il proseguimento dell'attacco, abbattendo la correlazione degli step successivi.

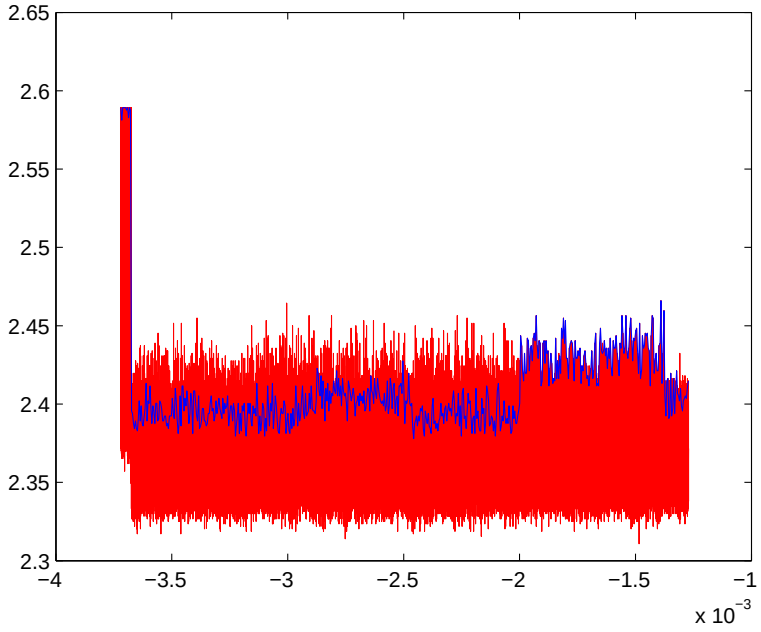

Figura 5.9: Traccia di potenza (colore rosso) e i picchi estratti dagli script automatici(colore blu). Lo script non riesce totalmente ad agganciarsi alla traccia. La causa maggiore di questo fenomeno è la dilatazione del clock che cambia leggermente durante l'elaborazione e rende difficile l'estrazione dei picchi

5.6 Composizione della traccia

La traccia di potenza registrata consente di individuare in essa varie parti distinte tra loro: la lettura della EEPROM, la codifica Keeloq, la preparazione della trasmissione, il preambolo e la trasmissione dei bit. Si può facilmente individuare la presenza della lettura della EEPROM dato il considerevole consumo di potenza che questa fase richiede rispetto al resto della traccia. La lettura della EEPROM è la parte in cui l'HCS accede alla memoria per estrarre il contatore, l'id del telecomando e la chiave con cui deve codificare la trama prima della trasmissione. In questa fase non è possibile estrarre informazioni utili al fine di conoscere la chiave del dispositivo. La fase immediatamente successiva è

la parte di codifica. In questa fase l'HCS elabora i dati estratti dalla memoria e li processa per ottenere la trama cifrata alla fine del ciclo. Questa è la parte di nostro interesse dove andremo ad estrarre i valori. Questa porzione di traccia è caratterizzata da una struttura periodica e distinta dalla parte successiva da una serie di picchi più alti che sono presenti solo in questa fase. L'analisi

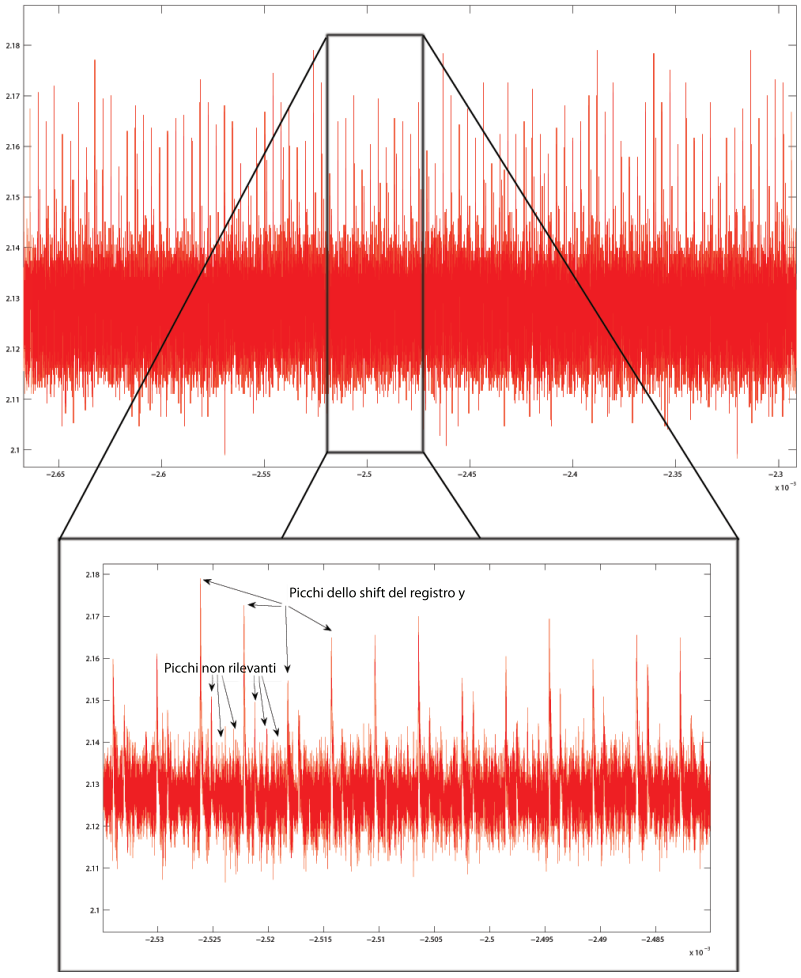

Figura 5.10: Zoom della traccia. I picchi intermedi riguardano le operazioni della codifica Keeloq, i picchi più alti lo shift del registro y che andremo a considerare i nostri valori intermedi

effettuata rileva come ogni 4 cicli di clock è presente un picco più rilevante. Questa periodicità presente in tutta la fase di codifica è stata studiata ed è stato ipotizzato che il picco più alto sia direttamente collegato con il valore del registro della trama Y. Questo perchè, come descritto in precedenza, si ha che lo shift del registro e il conseguente cambio di bit nelle varie celle del registro contribuisce in maniera più rilevante al un consumo di potenza rispetto a tutto il resto delle operazioni del Keeloq. Con questa ipotesi possiamo associare il picco alla distanza di Hamming tra il valore che c'era in precedenza nel registro e quello presente nel ciclo corrente di codifica. In questo modo i punti intermedi interessanti si riducono ai soli picchi più rilevanti che sono presenti nella zona di decodifica. Ai fini dell'attacco non sono utili i valori processati nei restanti cicli di clock poichè sono associati alle operazioni di XOR e di calcolo del risultato della funzione non lineare. Portando avanti questa ipotesi e costruendo l'attacco su questi valori di picco, sono stati ottenuti risultati favorevoli che hanno confermato l'ipotesi fatta.

5.7 Algoritmo di attacco DPA

Dopo aver ottenuto e tabulato i valori intermedi della traccia, possiamo procedere ad eseguire l'attacco DPA seguendo questo schema:

1. Calcolo delle chiavi ipotetiche di m bit
2. Esecuzione un attacco con Correlation Power Analysis (CPA) sul ciclo (528 - m) utilizzando il modello P_{Hyp} e le chiavi k, per ognuna delle chiavi ipotetiche calcolate
3. Estrazione della chiave candidata

Al primo passo andiamo a calcolare le chiavi ipotetiche. Poichè l'attacco viene fatto su gruppi di bit e non su singoli bit, le chiavi ipotetiche saranno tutte le possibili combinazioni di m bit. Nel nostro attacco andiamo a ricercare i bit della chiave 8 alla volta. Quindi al primo passo abbiamo $2^8 = 256$ chiavi ipotetiche.

Quando andiamo a praticare il CPA la prima volta, per trovare i primi 8 bit della chiave, andiamo a lavorare sul ciclo di decodifica $528 - 8 = 520$, quindi prenderemo in considerazione il 520-esimo picco estratto dalla traccia di potenza. Presi i valori corrispondenti al 520-esimo picco delle n tracce di potenza acquisite, andiamo a calcolare quello che è il valore ipotetico al ciclo 520. Per ogni chiave, quindi, calcoleremo la decodifica a partire dal testo cifrato, acquisito durante le misurazioni e come risultato avremo la distanza di Hamming, come dettato dal modello P_{Hyp} utilizzato. Calcolate le n distanze di Hamming per le 2^8 combinazioni di chiavi, andiamo ora a calcolare la correlazione per estrarre la chiave candidata.
La correlazione viene calcolata tra gli n valori estratti dalle tracce di potenza e gli n valori calcolati per ognuna delle chiavi ipotetiche. A questo punto abbiamo come risultato una serie di 2^8 correlazioni, ognuna corrispondente alla relativa chiave ipotetica. L'estrazione della chiave candidata si basa essenzialmente su quale correlazione risulta maggiore. Infatti una maggior correlazione significa

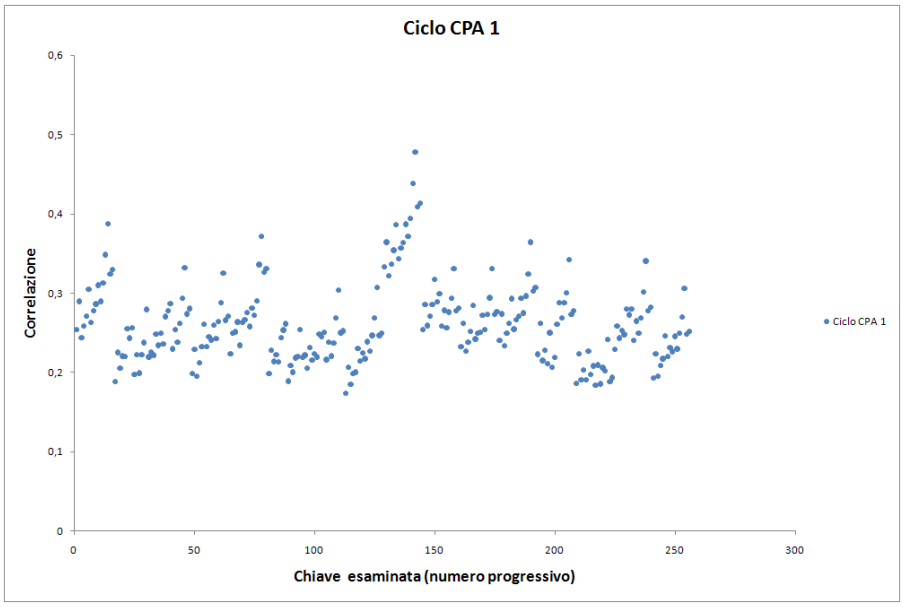

Figura 5.11: Rappresentazione del primo ciclo che recupera i primi 8 bit della chiave

che l'andamento dei dati simulati ben si avvicina all'andamento misurato sul picco corrispondente. Le misurazioni di 30 tracce di potenza hanno portato nella maggior parte dei casi ad un attacco di successo. Vediamo in Fig. 5.11 il grafico che mostra, per ognuna delle 256 chiavi, le rispettive correlazioni. Come possiamo vedere, la chiave candidata corretta ha la correlazione più alta. Anche altre chiavi vicine a quella corretta hanno una correlazione più alta segno che le chiavi simili, hanno un comportamento simulato che si approssima bene, all'avvicinarsi della chiave corretta, a quello delle tracce acquisite e quindi la correlazione tende ad aumentare. Oltre alle chiavi vicine a quella corretta ci sono altre chiavi che sono sopra la media e sono i falsi positivi, ovvero chiavi che hanno una correlazione maggiore della media e che possono rovinare l'attacco se la chiave corretta non ha una correlazione superiore a queste. L'attacco stesso, con l'aumentare del numero di tracce acquisite, aumenta la differenza di correlazione tra i falsi positivi e la chiave corretta.
Trovata la chiave candidata, questa avrà soltanto 8 bit dei 64 bit che andiamo a cercare. Per questo l'attacco viene ripetuto 64/m volte. Ad ogni ripetizione il picco su cui estrarre i valori dalle power analysis viene spostato. Se consideriamo f il numero progressivo dell'attacco, $1 < f < 64/m$, allora il picco considerato all'attacco f sarà $528 - fm$.

Ogni attacco risulta dipendente dai suoi precedenti e quindi i bit, estratti prima come corretti, vengono utilizzati al ciclo successivo per poter decodificare la trama. Questo meccanismo pone molta importanza sul fatto che i bit precedenti devono essere corretti, altrimenti i successivi attacchi si basano su ipotesi non valide e trovano falsi positivi. La consequenzialità degli attacchi è necessaria per trovare i bit della chiave. Per questo motivo bisogna seguire l'ordine con cui i bit entrano nel ciclo di decodifica; non è possibile calcolare i bit della chiave applicando l'attacco in ordine diverso da quello che viene seguito nelle operazioni di decodifica. Nel grafico, contenuto in Fig. 5.13, possiamo avere un'idea d'insieme di come la correlazione durante tutto l'attacco DPA si collochi in un intervallo di valori che va da 0.2 a circa 0.6. La correlazione non è mai 0, poichè il modello è preciso, ma non perfetto e non considera al

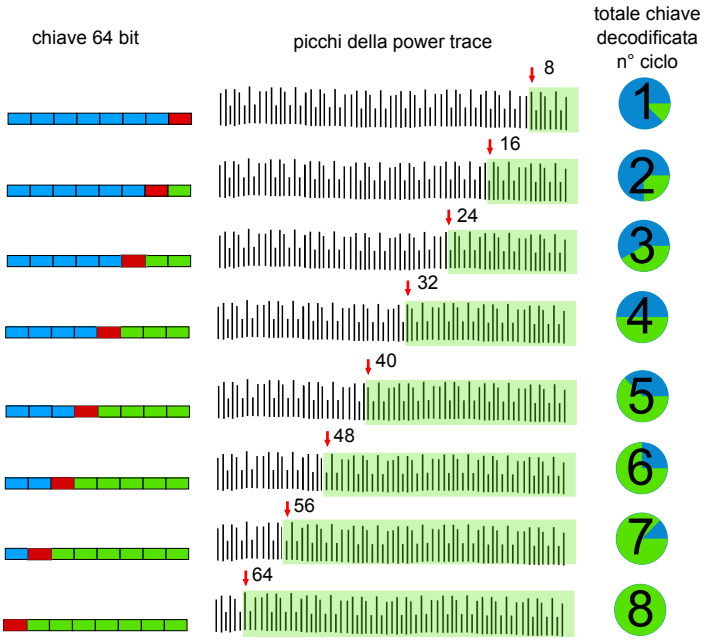

Figura 5.12: Schema degli attacchi DPA: l'attacco viene ripetuto 8 volte in quanto vengono calcolati 8 bit della chiave ad ogni ciclo. I cicli utilizzano i risultati degli attacchi che li precedono.

suo interno molti degli effetti secondari che sono invece presenti nella realtà. Dal confronto, mostrato in Fig. 5.14, possiamo vedere come si manifestano due fenomeni che differenziano il primo dall'ultimo ciclo. Nell'ultimo ciclo infatti, la correlazione è maggiore ed è presente un appiattimento dei valori di correlazione attorno alla media. Questi due fenomeni, meno presenti nel primo ciclo, evidenziano come ci sia un deterioramento delle prestazioni dell'attacco CPA con il susseguirsi dei cicli. Infatti, come si vede dal grafico la chiave candidata corretta è meno evidente e al tempo stesso emergono molti picchi che possono essere falsi positivi. Questo genera la necessità di acquisire più tracce per poter recuperare l'intera chiave. Se in questo attacco avessimo avuto meno tracce di quante ne sono state utilizzate, all'ottavo ciclo la correlazione della candidata corretta avrebbe potuto essere inferiore ad un falso positivo e quindi fallire

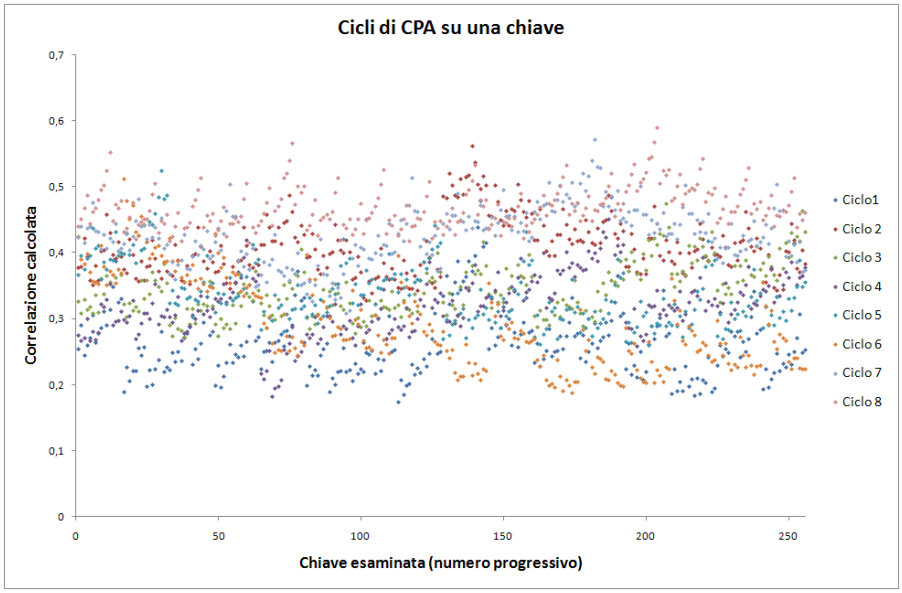

Figura 5.13: Il grafico rappresenta tutti gli 8 cicli CPA da 8 bit che devono essere completati per poter recuperare tutta la chiave da 64 bit

l'attacco.

5.7.1 Centratura dei valori

L'analisi dei picchi maggiori, che riteniamo direttamente connessi al contenuto del registro Y, porta con sè una problematica: i picchi presenti nella fase di decodifica delle tracce sono superiori al numero di cicli di decodifica Keeloq. Il problema di avere più di 528 picchi, ovvero più del numero di cicli del Keeloq, pone una questione su come togliere i picchi in eccesso per estrarre solo quelli coinvolti nella codifica. I picchi presenti sono circa 25 in più dei cicli della specifica dell'algoritmo. Non abbiamo informazioni in merito alla collocazione della fase di codifica e supponendo che venga fatta in modo sequenziale (ovvero che i picchi eccedenti non si trovino tra i 528 picchi, ma soltanto all'inizio o alla fine) abbiamo 25 possibilità di spostare la codifica collocando i picchi o prima o dopo di essa. Per fare questo è necessario capire come discriminare questi picchi in

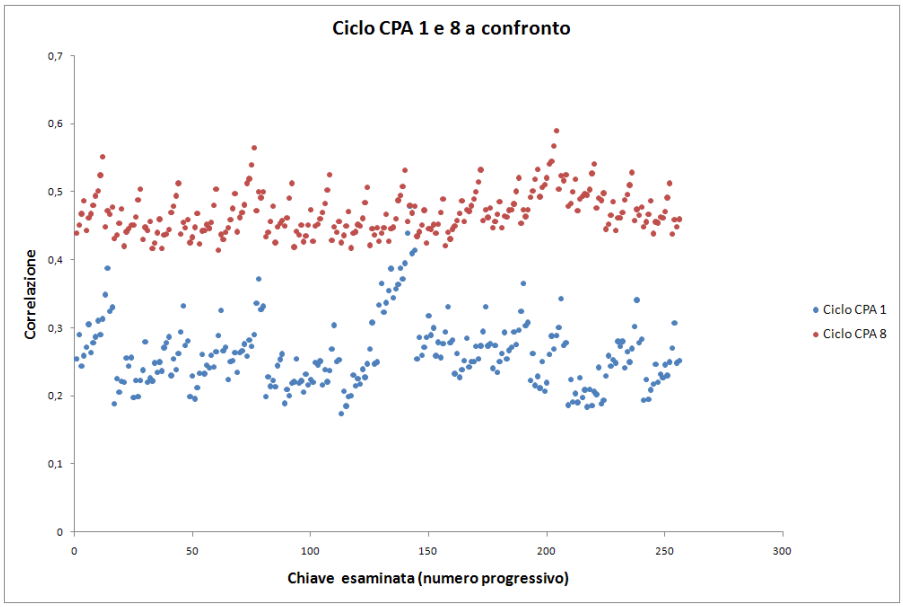

Figura 5.14: Grafico di confronto tra il primo e l'ottavo ciclo CPA che rivelano i primi 8 bit e gli ultimi 8 bit della chiave

quanto non ci sono informazioni utili a riguardo, nè sulla traccia acquisita, nè sulla documentazione riguardante l'algoritmo.
Per discriminare questi picchi abbiamo preparato un ambiente in cui tutto era noto, compresa la chiave che l'attacco DPA deve scoprire. In questo modo, è stato possibile correlare la chiave corretta alla giusta collocazione dei picchi. In questo nuovo setup l'attacco DPA è stato ripetuto 25 volte e ognuna di queste abbiamo shiftato i picchi in eccedenza. In questo modo l'attacco viene simulato coinvolgendo 528 picchi, ogni volta diversi poichè il blocco viene spostato per includere i 25 picchi che eccedono come si può vedere in uno schema esemplificativo in Fig. 5.15.

Fatte queste 25 simulazioni, abbiamo preso la correlazione della chiave corretta presente in tutte le simulazioni. Questa non è stata calcolata solamente sull'ottavo picco per ciclo, ma su tutti gli otto picchi che interessano ogni ottetto, ovvero ogni byte della chiave che corrisponde ad un ciclo di DPA. Nel

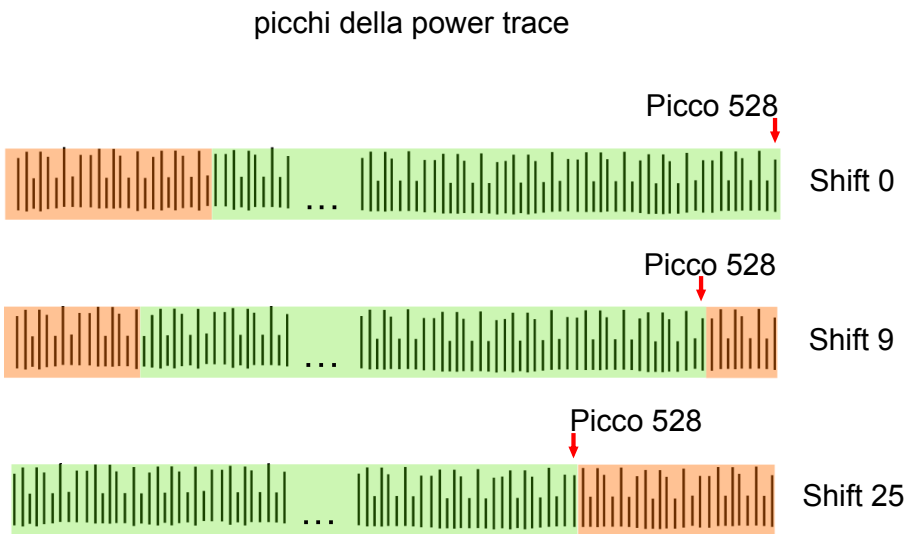

Figura 5.15: Le parti di colore arancione non sono coinvolte nella decodifica. Spostando il picco 528, da cui inizia l'attacco, i picchi vengono considerati in maniera diversa.

grafico in Fig. 5.16 possiamo visualizzare il risultato delle simulazioni. La serie in azzurro riporta le 25 simulazioni fatte con la chiave corretta. Per ognuna delle simulazioni riportiamo nel grafico un punto che ha come coordinate la media delle correlazioni della chiave corretta e la deviazione standard di queste. Ovviamente il risultato più attendibile sarà quello che ha correlazione più alta, che indica il successo del DPA, e la deviazione standard relativa più bassa, che mostra quanto le correlazioni calcolate siano attorno alla media. Questo valore è significativo poichè porta a verificare come i valori della correlazione siano omogenei e non frutto di una casualità data dalla somiglianza delle tracce acquisite con quelle ricavate. Come termine di paragone sono state riportate nel grafico due simulazioni fatte con chiavi non corrette e per distinguerle dalla simulazione corretta, sono state riportate in rosso e verde. I punti cerchiati rappresentano lo stesso shift, quello corretto, riportato alle tre diverse chiavi. Come si può vedere, le simulazioni con le chiavi sbagliate sono caratterizzate da una deviazione standard relativa alta oltre ad una correlazione bassa. Infatti

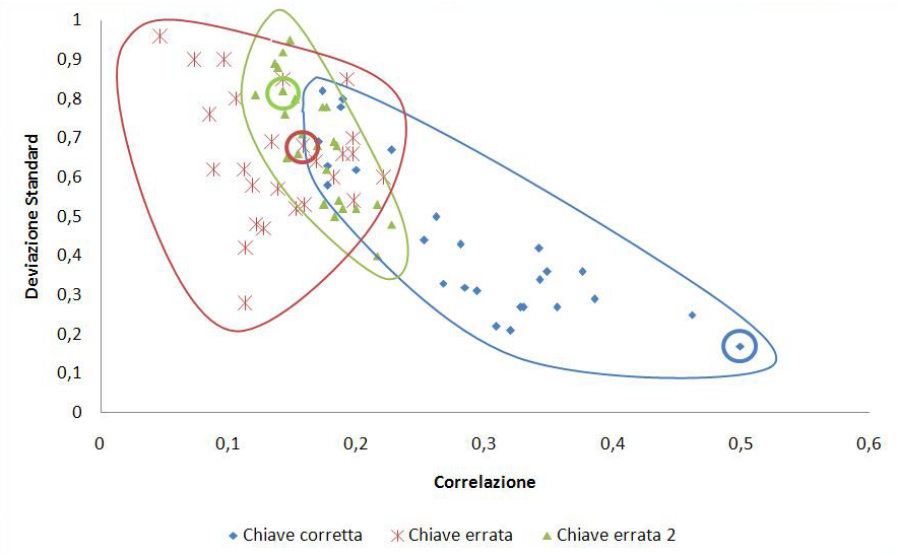

Figura 5.16: Grafico di correlazione mediata e deviazione standard relativa degli shift effettuati per individuare i picchi non coinvolti nella codifica Keeloq.

i risultati si collocano nella parte in alto a sinistra del grafico, risultando più compatti rispetto ai risultati azzurri. Gli shift, risultano tutti incorrelati in maniera uniforme. Lo si può notare poichè lo shift corretto è indistinguibile nelle simulazioni rosse e verdi e non emerge dal gruppo. Al contrario, nella simulazione azzurra, si può vedere come lo shift corretto si distingue dagli altri. Questo ci porta a capire come i risultati ottenuti tramite questa simulazione ci abbia dato un metodo valido per la determinazione dello shift da applicare alla nostra traccia di potenza che discrimini i picchi utili da quelli inutili.

Capitolo 6

Sviluppi futuri

6.1 Attacco DPA al ricevitore

Avendo conseguito un esito positivo nell'attacco al trasmettitore, è stato preso in esame il ricevitore e la possibilità di applicare un attacco DPA su di esso. I moduli che sono messi a disposizione da Microchip, come la famiglia HCS5XX, sono una serie di ricevitori dedicati alla decodifica del Keeloq, in accoppiamento con la famiglia di trasmettitori HCS3XX. L'implementazione e la decodifica di un ricevitore è più complessa di quella di un trasmettitore poichè la stazione ricevente può gestire più trasmettitori e quindi deve sia decodificare la trasmissione, sia verificare il telecomando e gestirlo in maniera differente a seconda della funzione a cui è stato associato. Inoltre in una stazione ricevente c'è implementata tutta la parte di apprendimento che consente di far apprendere alla parte ricevente un nuovo telecomando. Questa modalità aggiuntiva può variare da produttore a produttore in base al grado di sicurezza e alle scelte fatte in termini di progetto. Infatti la stazione ricevente può essere impostata per apprendere i telecomandi in maniera normale (soltanto utilizzando le trasmissioni) o in maniera sicura (utilizzando il seed, un codice diverso trasmesso soltanto durante la trasmissione). Inoltre il legame tra la chiave dispositivo e quella del produttore può essere diverso per ogni caso, a seconda dell'implementazione che il produttore ha scelto in fase di progettazione. Questo porta

ad aumentare ancora i casi che si possono trovare nello studio di una parte ricevente. L'hardware dedicato della famiglia di riceventi HCS5XX è quindi più complesso rispetto a quello dei trasmettitori e inoltre vengono eseguite molte più operazioni durante il suo funzionamento. L'implementazione di funzioni molto più complesse richiede una maggiore difficoltà nello studio del device per applicare un attacco DPA, infatti è necessario capire dove avviene la parte di decodifica per poi studiarla nella power analysis.
Nello studio effettuato sulle riceventi prese in esame, si è riscontrato un fenomeno importante: la maggior parte di queste riceventi utilizza un microcontrollore generico per il funzionamento di tutta la stazione ricevente. Questo comporta che l'algoritmo di decodifica del Keeloq e tutta la gestione dei trasmettitori viene implementata via software sul microcontrollore. In questo caso l'analisi dell'algoritmo cambia totalmente rispetto a quella dei dispositivi con hardware dedicato.

6.1.1 Analisi aspetti hardware

I microcontrollori che possono essere usati per gestire una parte ricevente sono moltissimi e tutti con peculiarità differenti. Bisogna notare che negli ultimi anni vengono impiegati dispositivi più evoluti a basso consumo di potenza. Nel nostro caso, andando a misurare il consumo del dispositivo, abbiamo che la traccia di potenza registrata sullo shunt risulta più disturbata poichè il rumore che va a sovrapporsi al segnale utile è maggiore. In questa situazione la degenerazione dell'attacco DPA aumenta molto e sono necessarie più tracce e un setup più accurato che minimizzi il rumore.
La gestione all'interno di un controllore generico può introdurre altre componenti indesiderate nella traccia di potenza. Innanzitutto è necessario rimuovere le componenti collegate al microcontrollore che possono generare consumi di potenza che rovinano la traccia. Già la sola parte di ingresso, dove viene ricevuta il segnale da decodificare, introduce disturbi sulla traccia di potenza, tuttavia è necessaria e non può essere scollegata. Le periferiche interne al controllore

sotto attacco sono anch'esse fattori di disturbo, poichè non possono essere controllate in un dispositivo sotto attacco e se risultano parti di hardware attive, essendo collegate alla stessa linea di alimentazione, introducono un contributo di consumo di potenza che si somma al nostro segnale utile. Questo rende più complessa l'interpretazione della traccia, perchè non essendo a conoscenza di cosa può essere attivo all'interno del controllore, risulta molto difficile capire cos'è realmente contributo utile e cosa invece fa parte di altre operazioni o consumi interni.

Il processing della decodifica in un microcontrollore generico porta anche ad un problema di sincronizzazione della traccia. La decodifica, assieme alle altre operazioni effettuate a margine, sono implementate sullo stesso hardware e vengono processate allo stesso modo. Per questo motivo la potenza consumata dal dispositivo sarà simile in ogni parte del codice, sia quando il controllore esegue operazioni relative al Keeloq, sia quando ne compie altre. L'implementazione software fa sì che le operazioni Keeloq, che nel trasmettitore erano elaborate tramite hardware dedicato, vengano qui tradotte in operazioni base del microntrollore. Per questo motivo, guardando il consumo di potenza, troveremo una traccia piuttosto uniforme poichè le operazioni svolte sono tutte riconducibili ad istruzioni assembly del microcontrollore e non a specifiche operazioni dedicate. In questo panorama, la decodifica del Keeloq risulta ben mimetizzata all'interno della traccia di potenza registrata e per questo è piuttosto complesso ricondursi alla struttura dell'algoritmo di decodifica. In una prima analisi è necessario trovare la posizione del codice in cui il controllore decodifica la trama. La localizzazione nella traccia di potenza, non essendo fattibile per somiglianza con quella del trasmettitore dove l'hardware era dedicato, viene fatta tramite considerazioni sul codice e sul comportamento del microcontrollore. Nella modalità di learning per esempio, è possibile individuare una zona, tra la fine della ricezione della trama e il segnale di avvenuta memorizzazione, dove la stazione ricevente deve elaborare la chiave cercando di decodificarla. In questo intervallo di traccia possiamo supporre che venga utilizzato l'algoritmo di decodifica Keeloq e che quindi sia la parte in cui deve essere applicato l'attacco DPA. In

questa parte è difficile che ci sia un trigger preciso su cui agganciare la nostra acquisizione, per questo si devono utilizzare altri sistemi. Il trigger che si può utilizzare per acquisire correttamente la porzione di traccia di nostro interesse è un trigger esterno, dato da un controllo di supporto che inizia l'acquisizione esattamente dopo la fine della trasmissione della trama o con un ritardo regolabile a seconda dell'implementazione utilizzata nel controllore preso in analisi.

6.1.2 Analisi degli aspetti software

L'implementazione a livello software del Keeloq è un altro aspetto da tenere presente nelle prestazioni dell'attacco DPA. L'utilizzo di codice per realizzare la decodifica può introdurre costrutti, tipo if o case-switch, che dipendono dai risultati parziali dell'elaborazione dei dati. Questo tipo di codice risulta più difficile da interpretare poichè diventa variabile quando i dati inseriti vengono cambiati. Quando accade questo ci possono essere variazioni nel numero di istruzioni necessarie per la decodifica, il che significa che non c'è una lunghezza fissa del codice. Per questo motivo trovare i punti intermedi fissi di traccia in traccia può risultare complesso e una prima elaborazione delle tracce può portare a risultati sbagliati se non si considera correttamente questo fenomeno. Un importante considerazione che dobbiamo fare quando si considera l'implementazione software della parte ricevente è che la decisione del codice da scrivere è lasciata al progettista. Questo aspetto, seppure scontato, può portare ad alcune complicazioni in fase di analisi del codice. Infatti quando si cerca di capire il funzionamento e come vengono implementate le varie funzioni si fanno delle ipotesi sulla costruzione del programma che si basano sull'implementazione standard fornita dal produttore, in questo caso Microchip. Tuttavia è possibile che il progettista della stazione ricevente abbia scelto strade differenti da quella standard, per ragioni di memoria o per comodità nella gestione delle altre funzioni. Per questo motivo le assunzioni che vengono fatte a priori sull'attacco, non potendo conoscere il codice di programmazione, possono risultare sbagliate. L'attacco DPA effettuato su riceventi che non seguono il codice stan-

dard, nella migliore delle ipotesi può degenerare di molto le sue performance analizzando parti della traccia che non sono pertinenti alla decodifica Keeloq, ma che comunque sono correlati. Nell'ipotesi peggiore l'attacco non ha successo poichè l'implementazione dell'algoritmo, eseguita in maniera non standard, divisa in più parti o molto differente, porta all'analisi di punti della traccia che non sono relazionati con la decodifica e che quindi portano a risultati errati. In queste condizioni la parte ricevente può non essere attaccabile se non con uno studio di simulazione più approfondito e con informazioni aggiuntive sull'implementazione del codice contenuto nella parte ricevente.

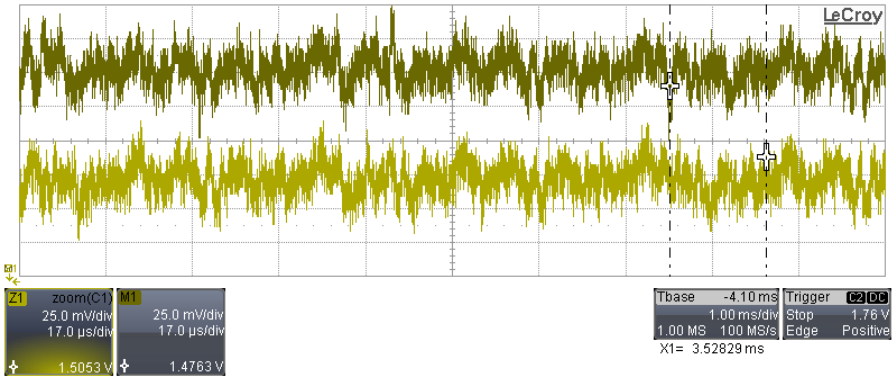

Figura 6.1: Acquisizione di due tracce di potenza dal ricevitore dello stesso periodo e scala, facendo elaborare al ricevitore gli stessi dati, nello stesso contesto

Vediamo in Fig. 6.1 le parti di due power trace che elaborano gli stessi dati. Si può notare come, pur essendoci alcune somiglianze tra le tracce, esse non siano tuttavia uguali e questo influenza di molto l'analisi. La parte di rumore sovrapposto porta a mutare notevolmente le tracce rendendole diverse tra loro e poco confrontabili. C'è anche un contributo di disallineamento tra le due. Come è possibile notare dalla posizione del cursore, pur elaborando gli stessi dati nello stesso contesto, le due tracce sono disallineate, poichè evidentemente alcune operazioni estranee alla codifica sono state processate dal microcontrollore. In questa situazione si rende quindi difficile l'allineamento delle tracce.

6.2 Conclusioni

L'algoritmo Keeloq preso in considerazione è stato molto studiato in letteratura e a livello crittografico sono stati trovati alcuni punti deboli che hanno permesso di progettare attacchi che ne rivelassero la sua debolezza. Tuttavia gli attacchi, poco praticabili nelle applicazioni reali, sono rimasti un valido punto di riferimento per l'analisi degli algoritmi crittografici e la valutazione della loro sicurezza.
Gli attacchi costruiti sull'analisi di potenza, che si basa sulle informazioni contenute nel consumo di potenza, mettono in evidenza le criticità dell'algoritmo Keeloq nelle implementazioni reali di quest'ultimo. L'hardware che contiene il Keeloq rivela delle informazioni correlate con la codifica e decodifica e quindi compromette in maniera rilevante la sicurezza dell'algoritmo. Da questo punto di vista l'utilizzo di questo algoritmo in applicazioni critiche deve essere considerato molto attentamente, poichè pur essendo un algoritmo di codifica piuttosto sicuro, può rivelarsi molto indebolito dall'hardware su cui è implementato. In questo studio è stato dimostrato come l'hardware dedicato, su cui vengono costruiti molti radiotrasmettitori basati su rolling code, è facilmente attaccabile e con le dovute considerazioni si può estrarre la chiave di codifica con dell'attrezzatura di costo contenuto e con una quantità esigua di tempo-macchina. D'altro canto l'implementazione su microcontrollori generici di questo algoritmo pone più incognite e una difficoltà maggiore nel reperire le informazioni sensibili, ma anche se si può tuttavia portare a termine l'attacco. L'attacco basato su DPA risulta efficace e potente per correlare il consumo di potenza con le informazioni sensibili processate all'interno del controllore e le contromisure che vengono prese possono deteriorarne le prestazioni, ma in molti casi non impedire l'attacco. Per questo motivo protocolli di sicurezza (come quelli utilizzati nei sistemi di denaro elettronico) richiedono hardware che ha un alto livello di immunità agli attacci DPA. Enti, come il Cryptography Research (http://www.cryptography.com/), hanno sviluppato programmi di certificazione all'immunità agli attacchi DPA per rendere più sicuri dispositivi che vengono utilizzati in applicazioni critiche.

In questo contesto è importante valutare sia la robustezza dell'algoritmo utilizzato, sia l'immunità dell'hardware agli attacchi basati sull'analisi di potenza che potrebbe essere, come abbiamo visto, il punto debole del sistema che stiamo implementando.

Appendice A

Risultati attacco DPA

Riporto qui in appendice il risultato completo di un attacco DPA. L'attacco per recuparare la chiave da 64 bit è stato ripetuto 8 volte, recuperando 8 bit per ciclo. All'inizio di ogni ciclo viene riportata la chiave in esadecimale, divisa in due parti, chiav_ a e chiav_ b. La chiave viene elaborata divisa in due poichè i tipi di dato del programma scritti in C supportano al massimo 32 bit. I valori dei bit non ancora elaborati sono impostati al valore 'A'. Nei cicli successivi, si può notare come la chiave si completa man mano con i valori calcolati nei cicli precedenti. Nella riga successiva viene riportato, per ogni ciclo, la chiave ipotetica con il valore più alto tra tutte. Questa è la chiave candidata ad essere la chiave corretta. Infine vengono elencate tutte le chiavi ipotetiche con la corrispettiva correlazione calcolata. Questo valore di correlazione è la media delle correlazioni calcolate sugli otto cicli di codifica che interessano gli otto bit in esame.

```
|----------------------------------------|
|****************************************|
 chiav_a = AAAAAAAA , chiav_b = AAAA47AA
|****************************************|
 valore massimo = AAAA47AA, valore +0.6012
|****************************************|
```

chiave	media
AAAA00AA	+0.5332
AAAA01AA	+0.5397
AAAA02AA	+0.5495
AAAA03AA	+0.5515
AAAA04AA	+0.5620
AAAA05AA	+0.5537
AAAA06AA	+0.5635
AAAA07AA	+0.5903
AAAA08AA	+0.5096
AAAA09AA	+0.5118
AAAA0AAA	+0.5201
AAAA0BAA	+0.5327
AAAA0CAA	+0.5094
AAAA0DAA	+0.5067
AAAA0EAA	+0.5113
AAAA0FAA	+0.5309
AAAA10AA	+0.5363
AAAA11AA	+0.5380
AAAA12AA	+0.5388
AAAA13AA	+0.5518
AAAA14AA	+0.5421
AAAA15AA	+0.5404
AAAA16AA	+0.5504
AAAA17AA	+0.5686
AAAA18AA	+0.5090
AAAA19AA	+0.5154
AAAA1AAA	+0.5287
AAAA1BAA	+0.5322
AAAA1CAA	+0.5453
AAAA1DAA	+0.5366
AAAA1EAA	+0.5347
AAAA1FAA	+0.5607
AAAA20AA	+0.5080
AAAA21AA	+0.5114
AAAA22AA	+0.5115
AAAA23AA	+0.5220
AAAA24AA	+0.5286
AAAA25AA	+0.5238
AAAA26AA	+0.5369
AAAA27AA	+0.5569
AAAA28AA	+0.4750
AAAA29AA	+0.4784
AAAA2AAA	+0.4906
AAAA2BAA	+0.4986
AAAA2CAA	+0.4868
AAAA2DAA	+0.4817
AAAA2EAA	+0.4816
AAAA2FAA	+0.5040
AAAA30AA	+0.4885
AAAA31AA	+0.4910
AAAA32AA	+0.4958
AAAA33AA	+0.5043
AAAA34AA	+0.5071
AAAA35AA	+0.5022
AAAA36AA	+0.5109
AAAA37AA	+0.5329
AAAA38AA	+0.4813
AAAA39AA	+0.4848
AAAA3AAA	+0.4923
AAAA3BAA	+0.5016
AAAA3CAA	+0.4951
AAAA3DAA	+0.4889
AAAA3EAA	+0.4934
AAAA3FAA	+0.5136
AAAA40AA	+0.5339
AAAA41AA	+0.5391
AAAA42AA	+0.5417
AAAA43AA	+0.5448
AAAA44AA	+0.5732
AAAA45AA	+0.5649
AAAA46AA	+0.5781
AAAA47AA	+0.6012
AAAA48AA	+0.5155
AAAA49AA	+0.5170
AAAA4AAA	+0.5293
AAAA4BAA	+0.5374

AAAA4CAA |||| +0.5030
AAAA4DAA |||| +0.4988
AAAA4EAA |||| +0.5004
AAAA4FAA |||| +0.5181
AAAA50AA |||| +0.5300
AAAA51AA |||| +0.5312
AAAA52AA |||| +0.5329
AAAA53AA |||| +0.5414
AAAA54AA |||| +0.5209
AAAA55AA |||| +0.5189
AAAA56AA |||| +0.5300
AAAA57AA |||| +0.5459
AAAA58AA |||| +0.4899
AAAA59AA |||| +0.4960
AAAA5AAA |||| +0.5017
AAAA5BAA |||| +0.5053
AAAA5CAA |||| +0.5282
AAAA5DAA |||| +0.5214
AAAA5EAA |||| +0.5234
AAAA5FAA |||| +0.5445
AAAA60AA |||| +0.5295
AAAA61AA |||| +0.5328
AAAA62AA |||| +0.5402
AAAA63AA |||| +0.5447
AAAA64AA |||| +0.5334
AAAA65AA |||| +0.5303
AAAA66AA |||| +0.5419
AAAA67AA |||| +0.5596
AAAA68AA |||| +0.4829
AAAA69AA |||| +0.4872
AAAA6AAA |||| +0.4921
AAAA6BAA |||| +0.4977
AAAA6CAA |||| +0.4992
AAAA6DAA |||| +0.4968
AAAA6EAA |||| +0.5028
AAAA6FAA |||| +0.5233
AAAA70AA |||| +0.5159
AAAA71AA |||| +0.5186
AAAA72AA |||| +0.5211
AAAA73AA |||| +0.5270
AAAA74AA |||| +0.5384
AAAA75AA |||| +0.5339
AAAA76AA |||| +0.5443
AAAA77AA |||| +0.5654
AAAA78AA |||| +0.5073
AAAA79AA |||| +0.5100
AAAA7AAA |||| +0.5219
AAAA7BAA |||| +0.5275
AAAA7CAA |||| +0.5119
AAAA7DAA |||| +0.5058
AAAA7EAA |||| +0.5077
AAAA7FAA |||| +0.5266
AAAA80AA |||| +0.4911
AAAA81AA |||| +0.4978
AAAA82AA |||| +0.4949
AAAA83AA |||| +0.5010
AAAA84AA |||| +0.5446
AAAA85AA |||| +0.5329
AAAA86AA |||| +0.5461
AAAA87AA |||| +0.5760
AAAA88AA |||| +0.4781
AAAA89AA |||| +0.4796
AAAA8AAA |||| +0.4896
AAAA8BAA |||| +0.5038
AAAA8CAA |||| +0.4611
AAAA8DAA |||| +0.4576
AAAA8EAA |||| +0.4555
AAAA8FAA |||| +0.4781
AAAA90AA |||| +0.4954
AAAA91AA |||| +0.4968
AAAA92AA |||| +0.4937
AAAA93AA |||| +0.5090
AAAA94AA |||| +0.4903
AAAA95AA |||| +0.4877
AAAA96AA |||| +0.4967
AAAA97AA |||| +0.5201
AAAA98AA |||| +0.4551
AAAA99AA |||| +0.4622

```
AAAA9AAA |||| +0.4672
AAAA9BAA |||| +0.4740
AAAA9CAA |||| +0.5023
AAAA9DAA |||| +0.4914
AAAA9EAA |||| +0.4906
AAAA9FAA |||| +0.5189
AAAAA0AA |||| +0.4562
AAAAA1AA |||| +0.4598
AAAAA2AA |||| +0.4635
AAAAA3AA |||| +0.4744
AAAAA4AA |||| +0.4737
AAAAA5AA |||| +0.4664
AAAAA6AA |||| +0.4748
AAAAA7AA |||| +0.5007
AAAAA8AA |||| +0.4029
AAAAA9AA |||| +0.4067
AAAAAAAA |||| +0.4098
AAAAABAA |||| +0.4201
AAAAACAA |||| +0.4212
AAAAADAA |||| +0.4152
AAAAAEAA |||| +0.4186
AAAAAFAA |||| +0.4423
AAAAB0AA |||| +0.4347
AAAAB1AA |||| +0.4369
AAAAB2AA |||| +0.4345
AAAAB3AA |||| +0.4448
AAAAB4AA |||| +0.4678
AAAAB5AA |||| +0.4608
AAAAB6AA |||| +0.4714
AAAAB7AA |||| +0.4947
AAAAB8AA |||| +0.4344
AAAAB9AA |||| +0.4379
AAAABAAA |||| +0.4486
AAAABBAA |||| +0.4575
AAAABCAA |||| +0.4419
AAAABDAA |||| +0.4335
AAAABEAA |||| +0.4316
AAAABFAA |||| +0.4552
AAAAC0AA |||| +0.4992
AAAAC1AA |||| +0.5055
AAAAC2AA |||| +0.5068
AAAAC3AA |||| +0.5142
AAAAC4AA |||| +0.5290
AAAAC5AA |||| +0.5210
AAAAC6AA |||| +0.5279
AAAAC7AA |||| +0.5552
AAAAC8AA |||| +0.4659
AAAAC9AA |||| +0.4688
AAAACAAA |||| +0.4725
AAAACBAA |||| +0.4847
AAAACCAA |||| +0.4644
AAAACDAA |||| +0.4599
AAAACEAA |||| +0.4613
AAAACFAA |||| +0.4810
AAAAD0AA |||| +0.5020
AAAAD1AA |||| +0.5044
AAAAD2AA |||| +0.5003
AAAAD3AA |||| +0.5120
AAAAD4AA |||| +0.5059
AAAAD5AA |||| +0.5025
AAAAD6AA |||| +0.5118
AAAAD7AA |||| +0.5304
AAAAD8AA |||| +0.4709
AAAAD9AA |||| +0.4772
AAAADAAA |||| +0.4855
AAAADBAA |||| +0.4913
AAAADCAA |||| +0.5057
AAAADDAA |||| +0.4982
AAAADEAA |||| +0.4924
AAAADFAA |||| +0.5170
AAAAE0AA |||| +0.5107
AAAAE1AA |||| +0.5150
AAAAE2AA |||| +0.5090
AAAAE3AA |||| +0.5203
AAAAE4AA |||| +0.5342
AAAAE5AA |||| +0.5291
AAAAE6AA |||| +0.5414
AAAAE7AA |||| +0.5639
```

```
AAAAE8AA |||| +0.4772
AAAAE9AA |||| +0.4817
AAAAEAAA |||| +0.4900
AAAAEBAA |||| +0.5003
AAAAECAA |||| +0.4842
AAAAEDAA |||| +0.4804
AAAAEEAA |||| +0.4758
AAAAEFAA |||| +0.5003
AAAAF0AA |||| +0.4916
AAAAF1AA |||| +0.4954
AAAAF2AA |||| +0.4932
AAAAF3AA |||| +0.5050
AAAAF4AA |||| +0.5026
AAAAF5AA |||| +0.4980
AAAAF6AA |||| +0.5041
AAAAF7AA |||| +0.5293
AAAAF8AA |||| +0.4764
AAAAF9AA |||| +0.4807
AAAAFAAA |||| +0.4837
AAAAFBAA |||| +0.4951
AAAAFCAA |||| +0.4944
AAAAFDAA |||| +0.4874
AAAAFEAA |||| +0.4880
AAAAFFAA |||| +0.5109
```

```
|----------------------------------------|
|****************************************|
 chiav_a = AAAAAAAA , chiav_b = AAAA4769
|****************************************|
 valore massimo = AAAA4769, valore +0.5788
|****************************************|
```

chiave	media		chiave	media
AAAA4700	+0.4660		AAAA471E	+0.4064
AAAA4701	+0.4907		AAAA471F	+0.4126
AAAA4702	+0.4632		AAAA4720	+0.4461
AAAA4703	+0.4592		AAAA4721	+0.4698
AAAA4704	+0.4570		AAAA4722	+0.4383
AAAA4705	+0.4759		AAAA4723	+0.4362
AAAA4706	+0.4438		AAAA4724	+0.4363
AAAA4707	+0.4463		AAAA4725	+0.4610
AAAA4708	+0.4872		AAAA4726	+0.4317
AAAA4709	+0.5133		AAAA4727	+0.4313
AAAA470A	+0.4765		AAAA4728	+0.4558
AAAA470B	+0.4713		AAAA4729	+0.4900
AAAA470C	+0.4552		AAAA472A	+0.4545
AAAA470D	+0.4714		AAAA472B	+0.4436
AAAA470E	+0.4547		AAAA472C	+0.4213
AAAA470F	+0.4596		AAAA472D	+0.4368
AAAA4710	+0.4320		AAAA472E	+0.4118
AAAA4711	+0.4550		AAAA472F	+0.4184
AAAA4712	+0.4240		AAAA4730	+0.4212
AAAA4713	+0.4218		AAAA4731	+0.4488
AAAA4714	+0.4235		AAAA4732	+0.4216
AAAA4715	+0.4450		AAAA4733	+0.4158
AAAA4716	+0.4199		AAAA4734	+0.4077
AAAA4717	+0.4202		AAAA4735	+0.4254
AAAA4718	+0.4527		AAAA4736	+0.3973
AAAA4719	+0.4847		AAAA4737	+0.3995
AAAA471A	+0.4492		AAAA4738	+0.4433
AAAA471B	+0.4402		AAAA4739	+0.4690
AAAA471C	+0.4119		AAAA473A	+0.4350
AAAA471D	+0.4282		AAAA473B	+0.4296
			AAAA473C	+0.4071
			AAAA473D	+0.4251

AAAA473E |||| +0.4106
AAAA473F |||| +0.4141
AAAA4740 |||| +0.5023
AAAA4741 |||| +0.5258
AAAA4742 |||| +0.4970
AAAA4743 |||| +0.4935
AAAA4744 |||| +0.4907
AAAA4745 |||| +0.5107
AAAA4746 |||| +0.4803
AAAA4747 |||| +0.4810
AAAA4748 |||| +0.5138
AAAA4749 |||| +0.5448
AAAA474A |||| +0.5046
AAAA474B |||| +0.4964
AAAA474C |||| +0.4797
AAAA474D |||| +0.4947
AAAA474E |||| +0.4736
AAAA474F |||| +0.4801
AAAA4750 |||| +0.4710
AAAA4751 |||| +0.4971
AAAA4752 |||| +0.4644
AAAA4753 |||| +0.4606
AAAA4754 |||| +0.4599
AAAA4755 |||| +0.4798
AAAA4756 |||| +0.4531
AAAA4757 |||| +0.4544
AAAA4758 |||| +0.5005
AAAA4759 |||| +0.5294
AAAA475A |||| +0.4939
AAAA475B |||| +0.4862
AAAA475C |||| +0.4602
AAAA475D |||| +0.4765
AAAA475E |||| +0.4586
AAAA475F |||| +0.4630
AAAA4760 |||| +0.5268
AAAA4761 |||| +0.5526
AAAA4762 |||| +0.5186
AAAA4763 |||| +0.5158
AAAA4764 |||| +0.5143
AAAA4765 |||| +0.5373
AAAA4766 |||| +0.5079
AAAA4767 |||| +0.5087
AAAA4768 |||| +0.5482
AAAA4769 |||| +0.5788
AAAA476A |||| +0.5454
AAAA476B |||| +0.5367
AAAA476C |||| +0.5157
AAAA476D |||| +0.5320
AAAA476E |||| +0.5092
AAAA476F |||| +0.5143
AAAA4770 |||| +0.4997
AAAA4771 |||| +0.5257
AAAA4772 |||| +0.4975
AAAA4773 |||| +0.4925
AAAA4774 |||| +0.4881
AAAA4775 |||| +0.5069
AAAA4776 |||| +0.4800
AAAA4777 |||| +0.4814
AAAA4778 |||| +0.5201
AAAA4779 |||| +0.5495
AAAA477A |||| +0.5145
AAAA477B |||| +0.5079
AAAA477C |||| +0.4783
AAAA477D |||| +0.4971
AAAA477E |||| +0.4771
AAAA477F |||| +0.4815
AAAA4780 |||| +0.4147
AAAA4781 |||| +0.4417
AAAA4782 |||| +0.4119
AAAA4783 |||| +0.4058
AAAA4784 |||| +0.4038
AAAA4785 |||| +0.4232
AAAA4786 |||| +0.3930
AAAA4787 |||| +0.3950
AAAA4788 |||| +0.4569
AAAA4789 |||| +0.4879
AAAA478A |||| +0.4435
AAAA478B |||| +0.4365

```
AAAA478C |||| +0.4149
AAAA478D |||| +0.4326
AAAA478E |||| +0.4159
AAAA478F |||| +0.4210
AAAA4790 |||| +0.4004
AAAA4791 |||| +0.4249
AAAA4792 |||| +0.3924
AAAA4793 |||| +0.3895
AAAA4794 |||| +0.3892
AAAA4795 |||| +0.4107
AAAA4796 |||| +0.3865
AAAA4797 |||| +0.3867
AAAA4798 |||| +0.4068
AAAA4799 |||| +0.4376
AAAA479A |||| +0.4014
AAAA479B |||| +0.3922
AAAA479C |||| +0.3636
AAAA479D |||| +0.3787
AAAA479E |||| +0.3615
AAAA479F |||| +0.3667
AAAA47A0 |||| +0.4790
AAAA47A1 |||| +0.5045
AAAA47A2 |||| +0.4709
AAAA47A3 |||| +0.4680
AAAA47A4 |||| +0.4707
AAAA47A5 |||| +0.4952
AAAA47A6 |||| +0.4629
AAAA47A7 |||| +0.4633
AAAA47A8 |||| +0.4786
AAAA47A9 |||| +0.5132
AAAA47AA |||| +0.4755
AAAA47AB |||| +0.4641
AAAA47AC |||| +0.4454
AAAA47AD |||| +0.4602
AAAA47AE |||| +0.4357
AAAA47AF |||| +0.4428
AAAA47B0 |||| +0.4283
AAAA47B1 |||| +0.4589
AAAA47B2 |||| +0.4287
AAAA47B3 |||| +0.4205
AAAA47B4 |||| +0.4168
AAAA47B5 |||| +0.4348
AAAA47B6 |||| +0.4043
AAAA47B7 |||| +0.4070
AAAA47B8 |||| +0.4683
AAAA47B9 |||| +0.4974
AAAA47BA |||| +0.4589
AAAA47BB |||| +0.4526
AAAA47BC |||| +0.4265
AAAA47BD |||| +0.4469
AAAA47BE |||| +0.4304
AAAA47BF |||| +0.4339
AAAA47C0 |||| +0.4635
AAAA47C1 |||| +0.4847
AAAA47C2 |||| +0.4572
AAAA47C3 |||| +0.4531
AAAA47C4 |||| +0.4588
AAAA47C5 |||| +0.4767
AAAA47C6 |||| +0.4452
AAAA47C7 |||| +0.4454
AAAA47C8 |||| +0.4689
AAAA47C9 |||| +0.4972
AAAA47CA |||| +0.4600
AAAA47CB |||| +0.4504
AAAA47CC |||| +0.4401
AAAA47CD |||| +0.4518
AAAA47CE |||| +0.4322
AAAA47CF |||| +0.4385
AAAA47D0 |||| +0.4124
AAAA47D1 |||| +0.4384
AAAA47D2 |||| +0.4082
AAAA47D3 |||| +0.4015
AAAA47D4 |||| +0.4091
AAAA47D5 |||| +0.4274
AAAA47D6 |||| +0.3977
AAAA47D7 |||| +0.3994
AAAA47D8 |||| +0.4602
AAAA47D9 |||| +0.4903
```

```
AAAA47DA |||| +0.4506        AAAA47ED |||| +0.4399
AAAA47DB |||| +0.4416        AAAA47EE |||| +0.4171
AAAA47DC |||| +0.4189        AAAA47EF |||| +0.4222
AAAA47DD |||| +0.4356        AAAA47F0 |||| +0.4153
AAAA47DE |||| +0.4166        AAAA47F1 |||| +0.4411
AAAA47DF |||| +0.4204        AAAA47F2 |||| +0.4145
AAAA47E0 |||| +0.4248        AAAA47F3 |||| +0.4083
AAAA47E1 |||| +0.4519        AAAA47F4 |||| +0.4073
AAAA47E2 |||| +0.4183        AAAA47F5 |||| +0.4260
AAAA47E3 |||| +0.4129        AAAA47F6 |||| +0.3984
AAAA47E4 |||| +0.4155        AAAA47F7 |||| +0.3997
AAAA47E5 |||| +0.4394        AAAA47F8 |||| +0.4324
AAAA47E6 |||| +0.4077        AAAA47F9 |||| +0.4620
AAAA47E7 |||| +0.4078        AAAA47FA |||| +0.4246
AAAA47E8 |||| +0.4596        AAAA47FB |||| +0.4167
AAAA47E9 |||| +0.4934        AAAA47FC |||| +0.3888
AAAA47EA |||| +0.4555        AAAA47FD |||| +0.4062
AAAA47EB |||| +0.4440        AAAA47FE |||| +0.3896
AAAA47EC |||| +0.4229        AAAA47FF |||| +0.3936
```

```
|----------------------------------------|
|****************************************|
 chiav_a = 47AAAAAA , chiav_b = AAAA4769
|****************************************|
 valore massimo = 47AAAAAA, valore +0.5010
|****************************************|
```

chiave	media
00AAAAAA	+0.3376
01AAAAAA	+0.3472
02AAAAAA	+0.3449
03AAAAAA	+0.3403
04AAAAAA	+0.3793
05AAAAAA	+0.3693
06AAAAAA	+0.3969
07AAAAAA	+0.4116
08AAAAAA	+0.3377
09AAAAAA	+0.3338
0AAAAAAA	+0.3396
0BAAAAAA	+0.3517
0CAAAAAA	+0.3328
0DAAAAAA	+0.3385
0EAAAAAA	+0.3533
0FAAAAAA	+0.3533
10AAAAAA	+0.3281
11AAAAAA	+0.3316
12AAAAAA	+0.3334
13AAAAAA	+0.3358
14AAAAAA	+0.3526
15AAAAAA	+0.3514
16AAAAAA	+0.3780
17AAAAAA	+0.3835
18AAAAAA	+0.3201
19AAAAAA	+0.3256
1AAAAAAA	+0.3304
1BAAAAAA	+0.3323
1CAAAAAA	+0.3235
1DAAAAAA	+0.3192
1EAAAAAA	+0.3311
1FAAAAAA	+0.3434
20AAAAAA	+0.3145
21AAAAAA	+0.3244
22AAAAAA	+0.3212
23AAAAAA	+0.3249
24AAAAAA	+0.3603
25AAAAAA	+0.3586
26AAAAAA	+0.3816
27AAAAAA	+0.3945
28AAAAAA	+0.3022
29AAAAAA	+0.3096
2AAAAAAA	+0.3100
2BAAAAAA	+0.3170
2CAAAAAA	+0.2942
2DAAAAAA	+0.2955
2EAAAAAA	+0.3057
2FAAAAAA	+0.3189
30AAAAAA	+0.3165
31AAAAAA	+0.3274
32AAAAAA	+0.3214
33AAAAAA	+0.3209
34AAAAAA	+0.3368
35AAAAAA	+0.3317
36AAAAAA	+0.3536
37AAAAAA	+0.3700
38AAAAAA	+0.3158
39AAAAAA	+0.3188
3AAAAAAA	+0.3220
3BAAAAAA	+0.3327
3CAAAAAA	+0.3267
3DAAAAAA	+0.3296

```
3EAAAAAA |||| +0.3367
3FAAAAAA |||| +0.3458
40AAAAAA |||| +0.4289
41AAAAAA |||| +0.4430
42AAAAAA |||| +0.4431
43AAAAAA |||| +0.4406
44AAAAAA |||| +0.4656
45AAAAAA |||| +0.4578
46AAAAAA |||| +0.4814
47AAAAAA |||| +0.5010
48AAAAAA |||| +0.4266
49AAAAAA |||| +0.4272
4AAAAAAA |||| +0.4334
4BAAAAAA |||| +0.4482
4CAAAAAA |||| +0.4195
4DAAAAAA |||| +0.4266
4EAAAAAA |||| +0.4367
4FAAAAAA |||| +0.4423
50AAAAAA |||| +0.4068
51AAAAAA |||| +0.4119
52AAAAAA |||| +0.4129
53AAAAAA |||| +0.4179
54AAAAAA |||| +0.4235
55AAAAAA |||| +0.4237
56AAAAAA |||| +0.4454
57AAAAAA |||| +0.4539
58AAAAAA |||| +0.3968
59AAAAAA |||| +0.4053
5AAAAAAA |||| +0.4090
5BAAAAAA |||| +0.4126
5CAAAAAA |||| +0.3983
5DAAAAAA |||| +0.3952
5EAAAAAA |||| +0.4034
5FAAAAAA |||| +0.4182
60AAAAAA |||| +0.3588
61AAAAAA |||| +0.3699
62AAAAAA |||| +0.3670
63AAAAAA |||| +0.3705
64AAAAAA |||| +0.4029
65AAAAAA |||| +0.4013
66AAAAAA |||| +0.4215
67AAAAAA |||| +0.4366
68AAAAAA |||| +0.3525
69AAAAAA |||| +0.3598
6AAAAAAA |||| +0.3601
6BAAAAAA |||| +0.3703
6CAAAAAA |||| +0.3405
6DAAAAAA |||| +0.3446
6EAAAAAA |||| +0.3534
6FAAAAAA |||| +0.3655
70AAAAAA |||| +0.3561
71AAAAAA |||| +0.3682
72AAAAAA |||| +0.3653
73AAAAAA |||| +0.3669
74AAAAAA |||| +0.3732
75AAAAAA |||| +0.3703
76AAAAAA |||| +0.3903
77AAAAAA |||| +0.4073
78AAAAAA |||| +0.3528
79AAAAAA |||| +0.3581
7AAAAAAA |||| +0.3614
7BAAAAAA |||| +0.3714
7CAAAAAA |||| +0.3634
7DAAAAAA |||| +0.3662
7EAAAAAA |||| +0.3745
7FAAAAAA |||| +0.3855
80AAAAAA |||| +0.3900
81AAAAAA |||| +0.4056
82AAAAAA |||| +0.4051
83AAAAAA |||| +0.3987
84AAAAAA |||| +0.4313
85AAAAAA |||| +0.4204
86AAAAAA |||| +0.4438
87AAAAAA |||| +0.4649
88AAAAAA |||| +0.3850
89AAAAAA |||| +0.3852
8AAAAAAA |||| +0.3883
8BAAAAAA |||| +0.4038
```

```
8CAAAAAA |||| +0.3773
8DAAAAAA |||| +0.3850
8EAAAAAA |||| +0.3977
8FAAAAAA |||| +0.4032
90AAAAAA |||| +0.3705
91AAAAAA |||| +0.3776
92AAAAAA |||| +0.3789
93AAAAAA |||| +0.3840
94AAAAAA |||| +0.3926
95AAAAAA |||| +0.3925
96AAAAAA |||| +0.4164
97AAAAAA |||| +0.4272
98AAAAAA |||| +0.3640
99AAAAAA |||| +0.3731
9AAAAAAA |||| +0.3762
9BAAAAAA |||| +0.3790
9CAAAAAA |||| +0.3671
9DAAAAAA |||| +0.3635
9EAAAAAA |||| +0.3725
9FAAAAAA |||| +0.3882
A0AAAAAA |||| +0.2881
A1AAAAAA |||| +0.3020
A2AAAAAA |||| +0.2963
A3AAAAAA |||| +0.2993
A4AAAAAA |||| +0.3437
A5AAAAAA |||| +0.3416
A6AAAAAA |||| +0.3606
A7AAAAAA |||| +0.3803
A8AAAAAA |||| +0.2905
A9AAAAAA |||| +0.2998
AAAAAAAA |||| +0.2972
ABAAAAAA |||| +0.3052
ACAAAAAA |||| +0.2780
ADAAAAAA |||| +0.2800
AEAAAAAA |||| +0.2884
AFAAAAAA |||| +0.3028
B0AAAAAA |||| +0.2887
B1AAAAAA |||| +0.3028
B2AAAAAA |||| +0.2963
B3AAAAAA |||| +0.2955
B4AAAAAA |||| +0.3070
B5AAAAAA |||| +0.3022
B6AAAAAA |||| +0.3207
B7AAAAAA |||| +0.3395
B8AAAAAA |||| +0.2784
B9AAAAAA |||| +0.2857
BAAAAAAA |||| +0.2864
BBAAAAAA |||| +0.2962
BCAAAAAA |||| +0.2939
BDAAAAAA |||| +0.2965
BEAAAAAA |||| +0.3030
BFAAAAAA |||| +0.3168
C0AAAAAA |||| +0.3313
C1AAAAAA |||| +0.3489
C2AAAAAA |||| +0.3429
C3AAAAAA |||| +0.3392
C4AAAAAA |||| +0.3843
C5AAAAAA |||| +0.3742
C6AAAAAA |||| +0.3941
C7AAAAAA |||| +0.4191
C8AAAAAA |||| +0.3373
C9AAAAAA |||| +0.3368
CAAAAAAA |||| +0.3366
CBAAAAAA |||| +0.3520
CCAAAAAA |||| +0.3284
CDAAAAAA |||| +0.3367
CEAAAAAA |||| +0.3440
CFAAAAAA |||| +0.3490
D0AAAAAA |||| +0.3226
D1AAAAAA |||| +0.3286
D2AAAAAA |||| +0.3270
D3AAAAAA |||| +0.3313
D4AAAAAA |||| +0.3410
D5AAAAAA |||| +0.3410
D6AAAAAA |||| +0.3619
D7AAAAAA |||| +0.3710
D8AAAAAA |||| +0.3061
D9AAAAAA |||| +0.3170
```

```
DAAAAAAA |||| +0.3174
DBAAAAAA |||| +0.3206
DCAAAAAA |||| +0.3130
DDAAAAAA |||| +0.3089
DEAAAAAA |||| +0.3168
DFAAAAAA |||| +0.3349
E0AAAAAA |||| +0.3363
E1AAAAAA |||| +0.3494
E2AAAAAA |||| +0.3442
E3AAAAAA |||| +0.3472
E4AAAAAA |||| +0.3776
E5AAAAAA |||| +0.3756
E6AAAAAA |||| +0.3942
E7AAAAAA |||| +0.4106
E8AAAAAA |||| +0.3187
E9AAAAAA |||| +0.3282
EAAAAAAA |||| +0.3259
EBAAAAAA |||| +0.3370
ECAAAAAA |||| +0.3101
EDAAAAAA |||| +0.3151
EEAAAAAA |||| +0.3208
EFAAAAAA |||| +0.3360
F0AAAAAA |||| +0.3378
F1AAAAAA |||| +0.3526
F2AAAAAA |||| +0.3476
F3AAAAAA |||| +0.3502
F4AAAAAA |||| +0.3586
F5AAAAAA |||| +0.3555
F6AAAAAA |||| +0.3742
F7AAAAAA |||| +0.3948
F8AAAAAA |||| +0.3352
F9AAAAAA |||| +0.3407
FAAAAAAA |||| +0.3412
FBAAAAAA |||| +0.3521
FCAAAAAA |||| +0.3454
FDAAAAAA |||| +0.3490
FEAAAAAA |||| +0.3541
FFAAAAAA |||| +0.3658
```

```
|----------------------------------------|
|****************************************|
 chiav_a = 4772AAAA , chiav_b = AAAA4769
|****************************************|
 valore massimo = 4772AAAA, valore +0.4623
|****************************************|
```

chiave	media
4700AAAA	+0.3033
4701AAAA	+0.3068
4702AAAA	+0.3519
4703AAAA	+0.3259
4704AAAA	+0.3105
4705AAAA	+0.3099
4706AAAA	+0.3186
4707AAAA	+0.2968
4708AAAA	+0.3007
4709AAAA	+0.3000
470AAAAA	+0.3195
470BAAAA	+0.2976
470CAAAA	+0.2807
470DAAAA	+0.2857
470EAAAA	+0.3192
470FAAAA	+0.2939
4710AAAA	+0.2866
4711AAAA	+0.2940
4712AAAA	+0.3518
4713AAAA	+0.3231
4714AAAA	+0.3206
4715AAAA	+0.3177
4716AAAA	+0.3168
4717AAAA	+0.2967
4718AAAA	+0.3089
4719AAAA	+0.3049
471AAAAA	+0.3195
471BAAAA	+0.2987
471CAAAA	+0.2884
471DAAAA	+0.2957
471EAAAA	+0.3372
471FAAAA	+0.3105
4720AAAA	+0.2524
4721AAAA	+0.2586
4722AAAA	+0.3027
4723AAAA	+0.2766
4724AAAA	+0.2654
4725AAAA	+0.2665
4726AAAA	+0.2741
4727AAAA	+0.2540
4728AAAA	+0.2729
4729AAAA	+0.2740
472AAAAA	+0.2891
472BAAAA	+0.2709
472CAAAA	+0.2524
472DAAAA	+0.2588
472EAAAA	+0.2896
472FAAAA	+0.2669
4730AAAA	+0.3180
4731AAAA	+0.3241
4732AAAA	+0.3712
4733AAAA	+0.3479
4734AAAA	+0.3327
4735AAAA	+0.3319
4736AAAA	+0.3288
4737AAAA	+0.3109
4738AAAA	+0.3196
4739AAAA	+0.3192
473AAAAA	+0.3284
473BAAAA	+0.3074
473CAAAA	+0.2946
473DAAAA	+0.3013

```
473EAAAA |||| +0.3427
473FAAAA |||| +0.3178
4740AAAA |||| +0.3218
4741AAAA |||| +0.3269
4742AAAA |||| +0.3778
4743AAAA |||| +0.3490
4744AAAA |||| +0.3295
4745AAAA |||| +0.3303
4746AAAA |||| +0.3377
4747AAAA |||| +0.3135
4748AAAA |||| +0.3118
4749AAAA |||| +0.3126
474AAAAA |||| +0.3314
474BAAAA |||| +0.3077
474CAAAA |||| +0.2934
474DAAAA |||| +0.2990
474EAAAA |||| +0.3348
474FAAAA |||| +0.3087
4750AAAA |||| +0.3124
4751AAAA |||| +0.3206
4752AAAA |||| +0.3758
4753AAAA |||| +0.3461
4754AAAA |||| +0.3408
4755AAAA |||| +0.3375
4756AAAA |||| +0.3371
4757AAAA |||| +0.3145
4758AAAA |||| +0.3417
4759AAAA |||| +0.3392
475AAAAA |||| +0.3534
475BAAAA |||| +0.3305
475CAAAA |||| +0.3114
475DAAAA |||| +0.3201
475EAAAA |||| +0.3623
475FAAAA |||| +0.3332
4760AAAA |||| +0.3114
4761AAAA |||| +0.3167
4762AAAA |||| +0.3729
4763AAAA |||| +0.3411
4764AAAA |||| +0.3300
4765AAAA |||| +0.3297
4766AAAA |||| +0.3376
4767AAAA |||| +0.3111
4768AAAA |||| +0.3455
4769AAAA |||| +0.3453
476AAAAA |||| +0.3671
476BAAAA |||| +0.3433
476CAAAA |||| +0.3162
476DAAAA |||| +0.3235
476EAAAA |||| +0.3584
476FAAAA |||| +0.3314
4770AAAA |||| +0.3920
4771AAAA |||| +0.3987
4772AAAA |||| +0.4623
4773AAAA |||| +0.4317
4774AAAA |||| +0.4124
4775AAAA |||| +0.4095
4776AAAA |||| +0.4087
4777AAAA |||| +0.3858
4778AAAA |||| +0.3701
4779AAAA |||| +0.3672
477AAAAA |||| +0.3888
477BAAAA |||| +0.3613
477CAAAA |||| +0.3514
477DAAAA |||| +0.3571
477EAAAA |||| +0.3977
477FAAAA |||| +0.3724
4780AAAA |||| +0.2873
4781AAAA |||| +0.2933
4782AAAA |||| +0.3458
4783AAAA |||| +0.3155
4784AAAA |||| +0.2963
4785AAAA |||| +0.2984
4786AAAA |||| +0.3073
4787AAAA |||| +0.2831
4788AAAA |||| +0.3033
4789AAAA |||| +0.3040
478AAAAA |||| +0.3216
478BAAAA |||| +0.2975
```

```
478CAAAA |||| +0.2782
478DAAAA |||| +0.2849
478EAAAA |||| +0.3263
478FAAAA |||| +0.2963
4790AAAA |||| +0.2892
4791AAAA |||| +0.2975
4792AAAA |||| +0.3648
4793AAAA |||| +0.3316
4794AAAA |||| +0.3090
4795AAAA |||| +0.3078
4796AAAA |||| +0.3070
4797AAAA |||| +0.2823
4798AAAA |||| +0.2913
4799AAAA |||| +0.2893
479AAAAA |||| +0.2978
479BAAAA |||| +0.2751
479CAAAA |||| +0.2552
479DAAAA |||| +0.2655
479EAAAA |||| +0.3199
479FAAAA |||| +0.2866
47A0AAAA |||| +0.2524
47A1AAAA |||| +0.2604
47A2AAAA |||| +0.3106
47A3AAAA |||| +0.2771
47A4AAAA |||| +0.2627
47A5AAAA |||| +0.2649
47A6AAAA |||| +0.2727
47A7AAAA |||| +0.2481
47A8AAAA |||| +0.2639
47A9AAAA |||| +0.2659
47AAAAAA |||| +0.2830
47ABAAAA |||| +0.2596
47ACAAAA |||| +0.2331
47ADAAAA |||| +0.2422
47AEAAAA |||| +0.2834
47AFAAAA |||| +0.2530
47B0AAAA |||| +0.3233
47B1AAAA |||| +0.3328
47B2AAAA |||| +0.3946
47B3AAAA |||| +0.3625
47B4AAAA |||| +0.3495
47B5AAAA |||| +0.3492
47B6AAAA |||| +0.3445
47B7AAAA |||| +0.3230
47B8AAAA |||| +0.3341
47B9AAAA |||| +0.3342
47BAAAAA |||| +0.3462
47BBAAAA |||| +0.3200
47BCAAAA |||| +0.3024
47BDAAAA |||| +0.3124
47BEAAAA |||| +0.3592
47BFAAAA |||| +0.3295
47C0AAAA |||| +0.2682
47C1AAAA |||| +0.2723
47C2AAAA |||| +0.3177
47C3AAAA |||| +0.2914
47C4AAAA |||| +0.2709
47C5AAAA |||| +0.2723
47C6AAAA |||| +0.2806
47C7AAAA |||| +0.2571
47C8AAAA |||| +0.2483
47C9AAAA |||| +0.2492
47CAAAAA |||| +0.2681
47CBAAAA |||| +0.2436
47CCAAAA |||| +0.2340
47CDAAAA |||| +0.2395
47CEAAAA |||| +0.2746
47CFAAAA |||| +0.2485
47D0AAAA |||| +0.2455
47D1AAAA |||| +0.2543
47D2AAAA |||| +0.3079
47D3AAAA |||| +0.2782
47D4AAAA |||| +0.2823
47D5AAAA |||| +0.2796
47D6AAAA |||| +0.2783
47D7AAAA |||| +0.2600
47D8AAAA |||| +0.2865
47D9AAAA |||| +0.2829
```

```
47DAAAAA |||| +0.2951
47DBAAAA |||| +0.2760
47DCAAAA |||| +0.2583
47DDAAAA |||| +0.2680
47DEAAAA |||| +0.3057
47DFAAAA |||| +0.2776
47E0AAAA |||| +0.2521
47E1AAAA |||| +0.2590
47E2AAAA |||| +0.3110
47E3AAAA |||| +0.2787
47E4AAAA |||| +0.2741
47E5AAAA |||| +0.2753
47E6AAAA |||| +0.2845
47E7AAAA |||| +0.2612
47E8AAAA |||| +0.2923
47E9AAAA |||| +0.2926
47EAAAAA |||| +0.3096
47EBAAAA |||| +0.2888
47ECAAAA |||| +0.2679
47EDAAAA |||| +0.2759
47EEAAAA |||| +0.3096
47EFAAAA |||| +0.2823
47F0AAAA |||| +0.3373
47F1AAAA |||| +0.3440
47F2AAAA |||| +0.4000
47F3AAAA |||| +0.3711
47F4AAAA |||| +0.3483
47F5AAAA |||| +0.3476
47F6AAAA |||| +0.3472
47F7AAAA |||| +0.3243
47F8AAAA |||| +0.3112
47F9AAAA |||| +0.3092
47FAAAAA |||| +0.3218
47FBAAAA |||| +0.2956
47FCAAAA |||| +0.2919
47FDAAAA |||| +0.2994
47FEAAAA |||| +0.3459
47FFAAAA |||| +0.3165
```

```
|----------------------------------------|
|****************************************|
 chiav_a = 477273AA , chiav_b = AAAA4769
|****************************************|
 valore massimo = 477273AA, valore +0.4788
|****************************************|
```

chiave	media		chiave	media
477200AA	+0.3045		47721EAA	+0.3023
477201AA	+0.3301		47721FAA	+0.3021
477202AA	+0.3242		477220AA	+0.2626
477203AA	+0.3552		477221AA	+0.2842
477204AA	+0.2782		477222AA	+0.2727
477205AA	+0.3044		477223AA	+0.3045
477206AA	+0.2761		477224AA	+0.2505
477207AA	+0.2992		477225AA	+0.2782
477208AA	+0.2952		477226AA	+0.2505
477209AA	+0.3194		477227AA	+0.2814
47720AAA	+0.3081		477228AA	+0.2511
47720BAA	+0.3397		477229AA	+0.2745
47720CAA	+0.2930		47722AAA	+0.2722
47720DAA	+0.3190		47722BAA	+0.3063
47720EAA	+0.2917		47722CAA	+0.2382
47720FAA	+0.3223		47722DAA	+0.2643
477210AA	+0.3472		47722EAA	+0.2529
477211AA	+0.3725		47722FAA	+0.2542
477212AA	+0.3727		477230AA	+0.3222
477213AA	+0.4050		477231AA	+0.3476
477214AA	+0.3138		477232AA	+0.3506
477215AA	+0.3433		477233AA	+0.3832
477216AA	+0.3074		477234AA	+0.2864
477217AA	+0.3370		477235AA	+0.3121
477218AA	+0.2741		477236AA	+0.2915
477219AA	+0.3000		477237AA	+0.2975
47721AAA	+0.2864		477238AA	+0.2571
47721BAA	+0.3232		477239AA	+0.2780
47721CAA	+0.2720		47723AAA	+0.2673
47721DAA	+0.3012		47723BAA	+0.2903
			47723CAA	+0.2615
			47723DAA	+0.2876

47723EAA |||| +0.2834
47723FAA |||| +0.2956
477240AA |||| +0.3077
477241AA |||| +0.3279
477242AA |||| +0.3242
477243AA |||| +0.3518
477244AA |||| +0.2841
477245AA |||| +0.3065
477246AA |||| +0.2792
477247AA |||| +0.3030
477248AA |||| +0.2844
477249AA |||| +0.3058
47724AAA |||| +0.2998
47724BAA |||| +0.3292
47724CAA |||| +0.2751
47724DAA |||| +0.2982
47724EAA |||| +0.2711
47724FAA |||| +0.2965
477250AA |||| +0.3298
477251AA |||| +0.3516
477252AA |||| +0.3572
477253AA |||| +0.3853
477254AA |||| +0.2986
477255AA |||| +0.3231
477256AA |||| +0.2859
477257AA |||| +0.3112
477258AA |||| +0.2841
477259AA |||| +0.3047
47725AAA |||| +0.2913
47725BAA |||| +0.3210
47725CAA |||| +0.2798
47725DAA |||| +0.3023
47725EAA |||| +0.2830
47725FAA |||| +0.3063
477260AA |||| +0.3850
477261AA |||| +0.4052
477262AA |||| +0.4010
477263AA |||| +0.4289
477264AA |||| +0.3654
477265AA |||| +0.3878
477266AA |||| +0.3630
477267AA |||| +0.3882
477268AA |||| +0.3782
477269AA |||| +0.3980
47726AAA |||| +0.3963
47726BAA |||| +0.4229
47726CAA |||| +0.3675
47726DAA |||| +0.3882
47726EAA |||| +0.3625
47726FAA |||| +0.3864
477270AA |||| +0.4312
477271AA |||| +0.4505
477272AA |||| +0.4534
477273AA |||| +0.4788
477274AA |||| +0.4009
477275AA |||| +0.4231
477276AA |||| +0.3922
477277AA |||| +0.4164
477278AA |||| +0.3689
477279AA |||| +0.3887
47727AAA |||| +0.3756
47727BAA |||| +0.4061
47727CAA |||| +0.3666
47727DAA |||| +0.3885
47727EAA |||| +0.3707
47727FAA |||| +0.3941
477280AA |||| +0.2480
477281AA |||| +0.2693
477282AA |||| +0.2876
477283AA |||| +0.2909
477284AA |||| +0.2475
477285AA |||| +0.2436
477286AA |||| +0.2543
477287AA |||| +0.2406
477288AA |||| +0.2470
477289AA |||| +0.2587
47728AAA |||| +0.2673
47728BAA |||| +0.2817

```
47728CAA |||| +0.2408
47728DAA |||| +0.2580
47728EAA |||| +0.2651
47728FAA |||| +0.2546
477290AA |||| +0.2764
477291AA |||| +0.3017
477292AA |||| +0.3157
477293AA |||| +0.3360
477294AA |||| +0.2626
477295AA |||| +0.2822
477296AA |||| +0.2653
477297AA |||| +0.2686
477298AA |||| +0.2447
477299AA |||| +0.2399
47729AAA |||| +0.2653
47729BAA |||| +0.2558
47729CAA |||| +0.2411
47729DAA |||| +0.2322
47729EAA |||| +0.2693
47729FAA |||| +0.2426
4772A0AA |||| +0.2738
4772A1AA |||| +0.2865
4772A2AA |||| +0.3099
4772A3AA |||| +0.3069
4772A4AA |||| +0.2850
4772A5AA |||| +0.2836
4772A6AA |||| +0.2907
4772A7AA |||| +0.2806
4772A8AA |||| +0.2783
4772A9AA |||| +0.2796
4772AAAA |||| +0.3009
4772ABAA |||| +0.3036
4772ACAA |||| +0.2673
4772ADAA |||| +0.2638
4772AEAA |||| +0.2866
4772AFAA |||| +0.2623
4772B0AA |||| +0.3196
4772B1AA |||| +0.3288
4772B2AA |||| +0.3588
4772B3AA |||| +0.3560
4772B4AA |||| +0.3131
4772B5AA |||| +0.2999
4772B6AA |||| +0.3161
4772B7AA |||| +0.2897
4772B8AA |||| +0.2971
4772B9AA |||| +0.2778
4772BAAA |||| +0.3116
4772BBAA |||| +0.2883
4772BCAA |||| +0.3003
4772BDAA |||| +0.2843
4772BEAA |||| +0.3221
4772BFAA |||| +0.3007
4772C0AA |||| +0.2992
4772C1AA |||| +0.3209
4772C2AA |||| +0.3190
4772C3AA |||| +0.3478
4772C4AA |||| +0.2792
4772C5AA |||| +0.3029
4772C6AA |||| +0.2808
4772C7AA |||| +0.2955
4772C8AA |||| +0.2841
4772C9AA |||| +0.3066
4772CAAA |||| +0.2970
4772CBAA |||| +0.3295
4772CCAA |||| +0.2723
4772CDAA |||| +0.2980
4772CEAA |||| +0.2722
4772CFAA |||| +0.2983
4772D0AA |||| +0.3364
4772D1AA |||| +0.3581
4772D2AA |||| +0.3622
4772D3AA |||| +0.3942
4772D4AA |||| +0.3025
4772D5AA |||| +0.3291
4772D6AA |||| +0.2913
4772D7AA |||| +0.3189
4772D8AA |||| +0.2919
4772D9AA |||| +0.3113
```

```
4772DAAA |||| +0.3082        4772EDAA |||| +0.3225
4772DBAA |||| +0.3349        4772EEAA |||| +0.2949
4772DCAA |||| +0.2893        4772EFAA |||| +0.3131
4772DDAA |||| +0.3119        4772F0AA |||| +0.3591
4772DEAA |||| +0.3193        4772F1AA |||| +0.3805
4772DFAA |||| +0.3125        4772F2AA |||| +0.3869
4772E0AA |||| +0.3170        4772F3AA |||| +0.4160
4772E1AA |||| +0.3374        4772F4AA |||| +0.3305
4772E2AA |||| +0.3275        4772F5AA |||| +0.3553
4772E3AA |||| +0.3594        4772F6AA |||| +0.3144
4772E4AA |||| +0.2968        4772F7AA |||| +0.3408
4772E5AA |||| +0.3229        4772F8AA |||| +0.2942
4772E6AA |||| +0.2987        4772F9AA |||| +0.3151
4772E7AA |||| +0.3268        4772FAAA |||| +0.2952
4772E8AA |||| +0.3098        4772FBAA |||| +0.3282
4772E9AA |||| +0.3307        4772FCAA |||| +0.2903
4772EAAA |||| +0.3322        4772FDAA |||| +0.3149
4772EBAA |||| +0.3653        4772FEAA |||| +0.3045
4772ECAA |||| +0.2986        4772FFAA |||| +0.3238
```

```
|----------------------------------------|
|****************************************|
 chiav_a = 47727365 , chiav_b = AAAA4769
|****************************************|
 valore massimo = 47727365, valore +0.4166
|****************************************|
```

chiave	media
47727300	+0.2551
47727301	+0.2645
47727302	+0.2533
47727303	+0.2658
47727304	+0.3056
47727305	+0.3350
47727306	+0.3056
47727307	+0.2983
47727308	+0.2648
47727309	+0.2861
4772730A	+0.2626
4772730B	+0.2630
4772730C	+0.2882
4772730D	+0.3079
4772730E	+0.2880
4772730F	+0.2910
47727310	+0.2023
47727311	+0.2137
47727312	+0.2222
47727313	+0.2299
47727314	+0.2766
47727315	+0.3017
47727316	+0.2551
47727317	+0.2505
47727318	+0.2265
47727319	+0.2407
4772731A	+0.2004
4772731B	+0.2054
4772731C	+0.2288
4772731D	+0.2500
4772731E	+0.2489
4772731F	+0.2471
47727320	+0.2197
47727321	+0.2263
47727322	+0.2216
47727323	+0.2334
47727324	+0.2781
47727325	+0.3050
47727326	+0.2718
47727327	+0.2641
47727328	+0.2433
47727329	+0.2603
4772732A	+0.2339
4772732B	+0.2353
4772732C	+0.2548
4772732D	+0.2711
4772732E	+0.2590
4772732F	+0.2616
47727330	+0.2865
47727331	+0.3004
47727332	+0.3007
47727333	+0.3089
47727334	+0.3386
47727335	+0.3616
47727336	+0.3257
47727337	+0.3244
47727338	+0.2910
47727339	+0.3077
4772733A	+0.2750
4772733B	+0.2800
4772733C	+0.3037
4772733D	+0.3260

```
4772733E |||| +0.3158
4772733F |||| +0.3163
47727340 |||| +0.2712
47727341 |||| +0.2801
47727342 |||| +0.2692
47727343 |||| +0.2815
47727344 |||| +0.3138
47727345 |||| +0.3411
47727346 |||| +0.3133
47727347 |||| +0.3057
47727348 |||| +0.2910
47727349 |||| +0.3095
4772734A |||| +0.2860
4772734B |||| +0.2869
4772734C |||| +0.3110
4772734D |||| +0.3282
4772734E |||| +0.3106
4772734F |||| +0.3135
47727350 |||| +0.3076
47727351 |||| +0.3210
47727352 |||| +0.3252
47727353 |||| +0.3337
47727354 |||| +0.3677
47727355 |||| +0.3890
47727356 |||| +0.3512
47727357 |||| +0.3497
47727358 |||| +0.3128
47727359 |||| +0.3278
4772735A |||| +0.2898
4772735B |||| +0.2946
4772735C |||| +0.3108
4772735D |||| +0.3315
4772735E |||| +0.3287
4772735F |||| +0.3288
47727360 |||| +0.3279
47727361 |||| +0.3380
47727362 |||| +0.3299
47727363 |||| +0.3458
47727364 |||| +0.3869
47727365 |||| +0.4166
47727366 |||| +0.3823
47727367 |||| +0.3757
47727368 |||| +0.3340
47727369 |||| +0.3549
4772736A |||| +0.3241
4772736B |||| +0.3257
4772736C |||| +0.3460
4772736D |||| +0.3654
4772736E |||| +0.3490
4772736F |||| +0.3540
47727370 |||| +0.2914
47727371 |||| +0.3041
47727372 |||| +0.3064
47727373 |||| +0.3145
47727374 |||| +0.3512
47727375 |||| +0.3754
47727376 |||| +0.3340
47727377 |||| +0.3303
47727378 |||| +0.3152
47727379 |||| +0.3291
4772737A |||| +0.2906
4772737B |||| +0.2968
4772737C |||| +0.3202
4772737D |||| +0.3419
4772737E |||| +0.3341
4772737F |||| +0.3324
47727380 |||| +0.2342
47727381 |||| +0.2424
47727382 |||| +0.2332
47727383 |||| +0.2446
47727384 |||| +0.2783
47727385 |||| +0.3032
47727386 |||| +0.2804
47727387 |||| +0.2745
47727388 |||| +0.2431
47727389 |||| +0.2596
4772738A |||| +0.2410
4772738B |||| +0.2416
```

```
4772738C |||| +0.2632
4772738D |||| +0.2784
4772738E |||| +0.2614
4772738F |||| +0.2646
47727390 |||| +0.2305
47727391 |||| +0.2427
47727392 |||| +0.2483
47727393 |||| +0.2554
47727394 |||| +0.2904
47727395 |||| +0.3105
47727396 |||| +0.2744
47727397 |||| +0.2720
47727398 |||| +0.2448
47727399 |||| +0.2572
4772739A |||| +0.2239
4772739B |||| +0.2284
4772739C |||| +0.2446
4772739D |||| +0.2636
4772739E |||| +0.2628
4772739F |||| +0.2621
477273A0 |||| +0.2219
477273A1 |||| +0.2299
477273A2 |||| +0.2247
477273A3 |||| +0.2374
477273A4 |||| +0.2810
477273A5 |||| +0.3072
477273A6 |||| +0.2778
477273A7 |||| +0.2719
477273A8 |||| +0.2373
477273A9 |||| +0.2533
477273AA |||| +0.2304
477273AB |||| +0.2322
477273AC |||| +0.2485
477273AD |||| +0.2641
477273AE |||| +0.2533
477273AF |||| +0.2567
477273B0 |||| +0.2383
477273B1 |||| +0.2496
477273B2 |||| +0.2522
477273B3 |||| +0.2609
477273B4 |||| +0.2918
477273B5 |||| +0.3126
477273B6 |||| +0.2819
477273B7 |||| +0.2804
477273B8 |||| +0.2478
477273B9 |||| +0.2611
477273BA |||| +0.2313
477273BB |||| +0.2367
477273BC |||| +0.2641
477273BD |||| +0.2840
477273BE |||| +0.2759
477273BF |||| +0.2772
477273C0 |||| +0.2156
477273C1 |||| +0.2247
477273C2 |||| +0.2129
477273C3 |||| +0.2248
477273C4 |||| +0.2647
477273C5 |||| +0.2920
477273C6 |||| +0.2699
477273C7 |||| +0.2628
477273C8 |||| +0.2325
477273C9 |||| +0.2491
477273CA |||| +0.2324
477273CB |||| +0.2339
477273CC |||| +0.2564
477273CD |||| +0.2745
477273CE |||| +0.2575
477273CF |||| +0.2594
477273D0 |||| +0.2033
477273D1 |||| +0.2144
477273D2 |||| +0.2244
477273D3 |||| +0.2339
477273D4 |||| +0.2790
477273D5 |||| +0.3021
477273D6 |||| +0.2633
477273D7 |||| +0.2598
477273D8 |||| +0.2194
477273D9 |||| +0.2333
```

```
477273DA |||| +0.1937
477273DB |||| +0.1980
477273DC |||| +0.2224
477273DD |||| +0.2433
477273DE |||| +0.2444
477273DF |||| +0.2449
477273E0 |||| +0.2326
477273E1 |||| +0.2419
477273E2 |||| +0.2364
477273E3 |||| +0.2510
477273E4 |||| +0.2959
477273E5 |||| +0.3239
477273E6 |||| +0.2940
477273E7 |||| +0.2880
477273E8 |||| +0.2408
477273E9 |||| +0.2578
477273EA |||| +0.2322
477273EB |||| +0.2334
477273EC |||| +0.2544
477273ED |||| +0.2716
477273EE |||| +0.2573
477273EF |||| +0.2614
477273F0 |||| +0.2616
477273F1 |||| +0.2723
477273F2 |||| +0.2740
477273F3 |||| +0.2807
477273F4 |||| +0.3146
477273F5 |||| +0.3342
477273F6 |||| +0.3019
477273F7 |||| +0.2991
477273F8 |||| +0.2878
477273F9 |||| +0.2990
477273FA |||| +0.2693
477273FB |||| +0.2744
477273FC |||| +0.2913
477273FD |||| +0.3108
477273FE |||| +0.3055
477273FF |||| +0.3038
```

```
|----------------------------------------|
|****************************************|
 chiav_a = 47727365 , chiav_b = 65AA4769
|****************************************|
 valore massimo = 65AA4769, valore +0.4832
|****************************************|
```

chiave	media		chiave	media
00AA4769	+0.2235		1EAA4769	+0.1787
01AA4769	+0.2389		1FAA4769	+0.1870
02AA4769	+0.1933		20AA4769	+0.2273
03AA4769	+0.2067		21AA4769	+0.2464
04AA4769	+0.2649		22AA4769	+0.1956
05AA4769	+0.2938		23AA4769	+0.2128
06AA4769	+0.2387		24AA4769	+0.2935
07AA4769	+0.2412		25AA4769	+0.3226
08AA4769	+0.2419		26AA4769	+0.2571
09AA4769	+0.2602		27AA4769	+0.2644
0AAA4769	+0.2258		28AA4769	+0.2355
0BAA4769	+0.2367		29AA4769	+0.2548
0CAA4769	+0.2488		2AAA4769	+0.2168
0DAA4769	+0.2699		2BAA4769	+0.2313
0EAA4769	+0.2195		2CAA4769	+0.2349
0FAA4769	+0.2273		2DAA4769	+0.2599
10AA4769	+0.2066		2EAA4769	+0.2065
11AA4769	+0.2255		2FAA4769	+0.2161
12AA4769	+0.1747		30AA4769	+0.2165
13AA4769	+0.1905		31AA4769	+0.2341
14AA4769	+0.2456		32AA4769	+0.1840
15AA4769	+0.2753		33AA4769	+0.1990
16AA4769	+0.2164		34AA4769	+0.2335
17AA4769	+0.2220		35AA4769	+0.2646
18AA4769	+0.2180		36AA4769	+0.2081
19AA4769	+0.2387		37AA4769	+0.2149
1AAA4769	+0.1973		38AA4769	+0.2117
1BAA4769	+0.2066		39AA4769	+0.2325
1CAA4769	+0.2093		3AAA4769	+0.1930
1DAA4769	+0.2322		3BAA4769	+0.2050
			3CAA4769	+0.2128
			3DAA4769	+0.2345

```
3EAA4769 |||| +0.1773
3FAA4769 |||| +0.1868
40AA4769 |||| +0.2941
41AA4769 |||| +0.3152
42AA4769 |||| +0.2651
43AA4769 |||| +0.2774
44AA4769 |||| +0.3518
45AA4769 |||| +0.3818
46AA4769 |||| +0.3211
47AA4769 |||| +0.3276
48AA4769 |||| +0.3295
49AA4769 |||| +0.3452
4AAA4769 |||| +0.3088
4BAA4769 |||| +0.3266
4CAA4769 |||| +0.3248
4DAA4769 |||| +0.3521
4EAA4769 |||| +0.3045
4FAA4769 |||| +0.3106
50AA4769 |||| +0.3315
51AA4769 |||| +0.3456
52AA4769 |||| +0.2956
53AA4769 |||| +0.3155
54AA4769 |||| +0.3599
55AA4769 |||| +0.3928
56AA4769 |||| +0.3442
57AA4769 |||| +0.3463
58AA4769 |||| +0.2758
59AA4769 |||| +0.3024
5AAA4769 |||| +0.2620
5BAA4769 |||| +0.2693
5CAA4769 |||| +0.3007
5DAA4769 |||| +0.3212
5EAA4769 |||| +0.2645
5FAA4769 |||| +0.2779
60AA4769 |||| +0.4034
61AA4769 |||| +0.4165
62AA4769 |||| +0.3708
63AA4769 |||| +0.3890
64AA4769 |||| +0.4528
65AA4769 |||| +0.4832
66AA4769 |||| +0.4284
67AA4769 |||| +0.4301
68AA4769 |||| +0.3692
69AA4769 |||| +0.3905
6AAA4769 |||| +0.3572
6BAA4769 |||| +0.3694
6CAA4769 |||| +0.3784
6DAA4769 |||| +0.4021
6EAA4769 |||| +0.3498
6FAA4769 |||| +0.3598
70AA4769 |||| +0.3128
71AA4769 |||| +0.3339
72AA4769 |||| +0.2848
73AA4769 |||| +0.2966
74AA4769 |||| +0.3402
75AA4769 |||| +0.3665
76AA4769 |||| +0.3139
77AA4769 |||| +0.3226
78AA4769 |||| +0.3351
79AA4769 |||| +0.3501
7AAA4769 |||| +0.3125
7BAA4769 |||| +0.3271
7CAA4769 |||| +0.3395
7DAA4769 |||| +0.3661
7EAA4769 |||| +0.3134
7FAA4769 |||| +0.3195
80AA4769 |||| +0.2331
81AA4769 |||| +0.2524
82AA4769 |||| +0.2069
83AA4769 |||| +0.2194
84AA4769 |||| +0.2723
85AA4769 |||| +0.2992
86AA4769 |||| +0.2433
87AA4769 |||| +0.2509
88AA4769 |||| +0.2561
89AA4769 |||| +0.2712
8AAA4769 |||| +0.2393
8BAA4769 |||| +0.2561
```

```
8CAA4769 |||| +0.2564
8DAA4769 |||| +0.2799
8EAA4769 |||| +0.2323
8FAA4769 |||| +0.2384
90AA4769 |||| +0.2248
91AA4769 |||| +0.2405
92AA4769 |||| +0.1918
93AA4769 |||| +0.2149
94AA4769 |||| +0.2657
95AA4769 |||| +0.2997
96AA4769 |||| +0.2417
97AA4769 |||| +0.2467
98AA4769 |||| +0.2298
99AA4769 |||| +0.2560
9AAA4769 |||| +0.2158
9BAA4769 |||| +0.2252
9CAA4769 |||| +0.2305
9DAA4769 |||| +0.2511
9EAA4769 |||| +0.1977
9FAA4769 |||| +0.2120
A0AA4769 |||| +0.3055
A1AA4769 |||| +0.3178
A2AA4769 |||| +0.2805
A3AA4769 |||| +0.2999
A4AA4769 |||| +0.3654
A5AA4769 |||| +0.3943
A6AA4769 |||| +0.3379
A7AA4769 |||| +0.3408
A8AA4769 |||| +0.3152
A9AA4769 |||| +0.3348
AAAA4769 |||| +0.3061
ABAA4769 |||| +0.3169
ACAA4769 |||| +0.3163
ADAA4769 |||| +0.3349
AEAA4769 |||| +0.2935
AFAA4769 |||| +0.3045
B0AA4769 |||| +0.2644
B1AA4769 |||| +0.2847
B2AA4769 |||| +0.2416
B3AA4769 |||| +0.2539
B4AA4769 |||| +0.2911
B5AA4769 |||| +0.3145
B6AA4769 |||| +0.2679
B7AA4769 |||| +0.2788
B8AA4769 |||| +0.2675
B9AA4769 |||| +0.2821
BAAA4769 |||| +0.2503
BBAA4769 |||| +0.2660
BCAA4769 |||| +0.2735
BDAA4769 |||| +0.2979
BEAA4769 |||| +0.2476
BFAA4769 |||| +0.2539
C0AA4769 |||| +0.2656
C1AA4769 |||| +0.2802
C2AA4769 |||| +0.2390
C3AA4769 |||| +0.2519
C4AA4769 |||| +0.3208
C5AA4769 |||| +0.3477
C6AA4769 |||| +0.2962
C7AA4769 |||| +0.2974
C8AA4769 |||| +0.3021
C9AA4769 |||| +0.3165
CAAA4769 |||| +0.2877
CBAA4769 |||| +0.2999
CCAA4769 |||| +0.3032
CDAA4769 |||| +0.3243
CEAA4769 |||| +0.2834
CFAA4769 |||| +0.2886
D0AA4769 |||| +0.3044
D1AA4769 |||| +0.3200
D2AA4769 |||| +0.2754
D3AA4769 |||| +0.2917
D4AA4769 |||| +0.3402
D5AA4769 |||| +0.3683
D6AA4769 |||| +0.3210
D7AA4769 |||| +0.3246
D8AA4769 |||| +0.2566
D9AA4769 |||| +0.2774
```

```
DAAA4769 |||| +0.2402
DBAA4769 |||| +0.2492
DCAA4769 |||| +0.2764
DDAA4769 |||| +0.2992
DEAA4769 |||| +0.2479
DFAA4769 |||| +0.2556
E0AA4769 |||| +0.2947
E1AA4769 |||| +0.3085
E2AA4769 |||| +0.2647
E3AA4769 |||| +0.2808
E4AA4769 |||| +0.3481
E5AA4769 |||| +0.3739
E6AA4769 |||| +0.3183
E7AA4769 |||| +0.3229
E8AA4769 |||| +0.2844
E9AA4769 |||| +0.3017
EAAA4769 |||| +0.2671
EBAA4769 |||| +0.2807
ECAA4769 |||| +0.2873
EDAA4769 |||| +0.3098
EEAA4769 |||| +0.2595
EFAA4769 |||| +0.2671
F0AA4769 |||| +0.2226
F1AA4769 |||| +0.2390
F2AA4769 |||| +0.1916
F3AA4769 |||| +0.2067
F4AA4769 |||| +0.2564
F5AA4769 |||| +0.2838
F6AA4769 |||| +0.2317
F7AA4769 |||| +0.2377
F8AA4769 |||| +0.2445
F9AA4769 |||| +0.2616
FAAA4769 |||| +0.2259
FBAA4769 |||| +0.2387
FCAA4769 |||| +0.2619
FDAA4769 |||| +0.2833
FEAA4769 |||| +0.2305
FFAA4769 |||| +0.2400
```

```
|---------------------------------------|
|***************************************|
 chiav_a = 47727365 , chiav_b = 65734769
|***************************************|
 valore massimo = 65734769, valore +0.4571
|***************************************|
```

chiave	media
65004769	+0.2603
65014769	+0.2711
65024769	+0.2949
65034769	+0.3195
65044769	+0.2234
65054769	+0.2374
65064769	+0.2267
65074769	+0.2483
65084769	+0.1842
65094769	+0.1927
650A4769	+0.2098
650B4769	+0.2354
650C4769	+0.2149
650D4769	+0.2314
650E4769	+0.2256
650F4769	+0.2441
65104769	+0.2730
65114769	+0.2829
65124769	+0.3065
65134769	+0.3324
65144769	+0.2491
65154769	+0.2644
65164769	+0.2512
65174769	+0.2690
65184769	+0.2269
65194769	+0.2338
651A4769	+0.2564
651B4769	+0.2845
651C4769	+0.2562
651D4769	+0.2720
651E4769	+0.2685
651F4769	+0.2893
65204769	+0.2217
65214769	+0.2359
65224769	+0.2590
65234769	+0.2858
65244769	+0.1909
65254769	+0.2063
65264769	+0.1946
65274769	+0.2164
65284769	+0.1720
65294769	+0.1812
652A4769	+0.2061
652B4769	+0.2329
652C4769	+0.2097
652D4769	+0.2279
652E4769	+0.2171
652F4769	+0.2386
65304769	+0.2810
65314769	+0.2933
65324769	+0.3250
65334769	+0.3530
65344769	+0.2509
65354769	+0.2667
65364769	+0.2501
65374769	+0.2728
65384769	+0.1944
65394769	+0.2051
653A4769	+0.2259
653B4769	+0.2569
653C4769	+0.2367
653D4769	+0.2560

```
653E4769 |||| +0.2499
653F4769 |||| +0.2742
65404769 |||| +0.2733
65414769 |||| +0.2838
65424769 |||| +0.3187
65434769 |||| +0.3446
65444769 |||| +0.2417
65454769 |||| +0.2535
65464769 |||| +0.2411
65474769 |||| +0.2655
65484769 |||| +0.1918
65494769 |||| +0.1999
654A4769 |||| +0.2190
654B4769 |||| +0.2453
654C4769 |||| +0.2244
654D4769 |||| +0.2398
654E4769 |||| +0.2379
654F4769 |||| +0.2581
65504769 |||| +0.2927
65514769 |||| +0.3021
65524769 |||| +0.3265
65534769 |||| +0.3543
65544769 |||| +0.2688
65554769 |||| +0.2842
65564769 |||| +0.2716
65574769 |||| +0.2930
65584769 |||| +0.2394
65594769 |||| +0.2479
655A4769 |||| +0.2758
655B4769 |||| +0.3080
655C4769 |||| +0.2669
655D4769 |||| +0.2809
655E4769 |||| +0.2761
655F4769 |||| +0.2986
65604769 |||| +0.3088
65614769 |||| +0.3186
65624769 |||| +0.3382
65634769 |||| +0.3591
65644769 |||| +0.2823
65654769 |||| +0.2934
65664769 |||| +0.2877
65674769 |||| +0.3082
65684769 |||| +0.2744
65694769 |||| +0.2811
656A4769 |||| +0.3050
656B4769 |||| +0.3317
656C4769 |||| +0.2963
656D4769 |||| +0.3105
656E4769 |||| +0.3054
656F4769 |||| +0.3232
65704769 |||| +0.3890
65714769 |||| +0.3974
65724769 |||| +0.4311
65734769 |||| +0.4571
65744769 |||| +0.3559
65754769 |||| +0.3687
65764769 |||| +0.3569
65774769 |||| +0.3781
65784769 |||| +0.3048
65794769 |||| +0.3108
657A4769 |||| +0.3320
657B4769 |||| +0.3619
657C4769 |||| +0.3438
657D4769 |||| +0.3578
657E4769 |||| +0.3610
657F4769 |||| +0.3836
65804769 |||| +0.2046
65814769 |||| +0.2158
65824769 |||| +0.2435
65834769 |||| +0.2666
65844769 |||| +0.1744
65854769 |||| +0.1852
65864769 |||| +0.1787
65874769 |||| +0.1996
65884769 |||| +0.1498
65894769 |||| +0.1566
658A4769 |||| +0.1823
658B4769 |||| +0.2005
```

```
658C4769 |||| +0.1796
658D4769 |||| +0.1947
658E4769 |||| +0.1926
658F4769 |||| +0.2119
65904769 |||| +0.2223
65914769 |||| +0.2316
65924769 |||| +0.2555
65934769 |||| +0.2784
65944769 |||| +0.2004
65954769 |||| +0.2124
65964769 |||| +0.2045
65974769 |||| +0.2229
65984769 |||| +0.1717
65994769 |||| +0.1804
659A4769 |||| +0.2029
659B4769 |||| +0.2288
659C4769 |||| +0.1965
659D4769 |||| +0.2093
659E4769 |||| +0.2092
659F4769 |||| +0.2289
65A04769 |||| +0.2261
65A14769 |||| +0.2377
65A24769 |||| +0.2569
65A34769 |||| +0.2783
65A44769 |||| +0.1980
65A54769 |||| +0.2093
65A64769 |||| +0.2034
65A74769 |||| +0.2234
65A84769 |||| +0.1846
65A94769 |||| +0.1927
65AA4769 |||| +0.2172
65AB4769 |||| +0.2422
65AC4769 |||| +0.2147
65AD4769 |||| +0.2295
65AE4769 |||| +0.2220
65AF4769 |||| +0.2412
65B04769 |||| +0.2819
65B14769 |||| +0.2920
65B24769 |||| +0.3225
65B34769 |||| +0.3467
65B44769 |||| +0.2605
65B54769 |||| +0.2738
65B64769 |||| +0.2602
65B74769 |||| +0.2812
65B84769 |||| +0.2185
65B94769 |||| +0.2274
65BA4769 |||| +0.2466
65BB4769 |||| +0.2741
65BC4769 |||| +0.2498
65BD4769 |||| +0.2639
65BE4769 |||| +0.2631
65BF4769 |||| +0.2850
65C04769 |||| +0.2494
65C14769 |||| +0.2595
65C24769 |||| +0.2814
65C34769 |||| +0.3030
65C44769 |||| +0.2271
65C54769 |||| +0.2383
65C64769 |||| +0.2267
65C74769 |||| +0.2511
65C84769 |||| +0.1786
65C94769 |||| +0.1872
65CA4769 |||| +0.2040
65CB4769 |||| +0.2297
65CC4769 |||| +0.2075
65CD4769 |||| +0.2210
65CE4769 |||| +0.2152
65CF4769 |||| +0.2356
65D04769 |||| +0.2582
65D14769 |||| +0.2675
65D24769 |||| +0.2881
65D34769 |||| +0.3122
65D44769 |||| +0.2398
65D54769 |||| +0.2530
65D64769 |||| +0.2399
65D74769 |||| +0.2593
65D84769 |||| +0.2157
65D94769 |||| +0.2228
```

```
65DA4769 |||| +0.2441
65DB4769 |||| +0.2688
65DC4769 |||| +0.2460
65DD4769 |||| +0.2573
65DE4769 |||| +0.2517
65DF4769 |||| +0.2736
65E04769 |||| +0.2380
65E14769 |||| +0.2506
65E24769 |||| +0.2712
65E34769 |||| +0.2948
65E44769 |||| +0.2068
65E54769 |||| +0.2180
65E64769 |||| +0.2108
65E74769 |||| +0.2333
65E84769 |||| +0.2001
65E94769 |||| +0.2086
65EA4769 |||| +0.2310
65EB4769 |||| +0.2566
65EC4769 |||| +0.2378
65ED4769 |||| +0.2533
65EE4769 |||| +0.2465
65EF4769 |||| +0.2690
65F04769 |||| +0.3021
65F14769 |||| +0.3128
65F24769 |||| +0.3396
65F34769 |||| +0.3644
65F44769 |||| +0.2773
65F54769 |||| +0.2908
65F64769 |||| +0.2802
65F74769 |||| +0.3046
65F84769 |||| +0.2229
65F94769 |||| +0.2331
65FA4769 |||| +0.2563
65FB4769 |||| +0.2882
65FC4769 |||| +0.2607
65FD4769 |||| +0.2754
65FE4769 |||| +0.2743
65FF4769 |||| +0.3003
```

Elenco delle figure

Bibliografia

[1] A. Bogdanov, "Attacks on the keeloq block cipher and authentication systems." vol. In 3rd Conference on RFID Security 2007 (RFIDSec 2007), 2007.

[2] Microchip, "Hopping code decoder using a pic16c56," vol. AN642, Available on http://www.keeloq.boom.ru/decryption.pdf, 1998.

[3] A. Bogdanov, "Cryptanalysis of the keeloq block cipher," 2007. [Online]. Available: http://eprint.iacr.org/2007/055/

[4] D. W. Alex Biryukov, "Slide attacks," vol. Proceedings of Fast Software Encryption 1999, volume 1636 of LNCS, Spinger-Verlag, 1999.

[5] ——, "Advanced slide attacks," vol. Proceedings of Fast Software Encryption 2000, volume 1807 of LNCS, Spinger-Verlag, 2000.

[6] G. V. B. Nicolas T. Courtis and D. Wagner, "Algebric and slide attacks on keeloq," vol. Proceedings of Fast Software Encryption 2008, Lecture Notes in Computer Science, Spinger-Verlag, 2008.

[7] O. D. E. B. B. P. Sebastian Indesteege, Nathan Keller, "A pratical attack on keeloq." [Online]. Available: http://www.springerlink.com

[8] T. P. Stefan Mangard, Elisabeth Oswald, *Power Analysis Attacks*. Springer, 2001.

[9] A. M. C. P. Thomas Eisenbarth, Timo Kasper, "On the power of power analysis in the real world: A complete break of the keeloq code hopping scheme."