POLITECNICO DI MILANO

Dipartimento di Elettronica e Informazione

Laurea Magistrale in Ingegneria delle Telecomunicazioni

ARCHITETTURE PRIVACY-FRIENDLY PER L'AGGREGAZIONE DI MISURE IN SMART GRID

Relatore: Prof. Giacomo VERTICALE

Correlatore: Ing. Cristina ROTTONDI

Tesi Magistrale di:

Daniele POLENGHI

Matricola: 764290

Marco SAVI

Matricola: 763605

Anno Accademico 2011–2012

SOMMARIO

In questo lavoro di tesi consideriamo due architetture di rete volte a garantire la privacy degli utenti che inviano le loro misure di consumo ad un'entità. Queste architetture si occupano dell'aggregazione spaziale e/o temporale delle misure, in modo tale da non rendere possibile, a tale entità, la conoscenza dei consumi individuali dei singoli utenti. Vengono realizzati due emulatori per la valutazione delle prestazioni delle due differenti architetture, specialmente in relazione ai protocolli di comunicazione da esse adottati.

Viene poi introdotto il concetto di privacy differenziale e viene definito come un attaccante può cercare di ridurre il livello di privacy differenziale degli utenti, ipotizzando due diversi modelli per la generazione delle misure di consumo; è quindi illustrata una contromisura che può essere adottata al fine di rendere l'attacco inefficiente. Infine, vengono mostrati i risultati sperimentali ottenuti a sostegno della teoria elaborata.

INDICE

CAPITOLO 1

INTRODUZIONE

La tecnologia digitale che abilita una comunicazione diretta tra le società di erogazione dell'energia elettrica e i suoi utenti e, di conseguenza, il rilevamento di misure lungo le linee di trasmissione, prende il nome di Smart Grid. Come Internet, una Smart Grid consiste in controlli, in elaboratori, in automazione, in nuove tecnologie e in nuove attrezzature che collaborano. In questo caso, però, esse lavorano con la rete di distribuzione elettrica per rispondere in modo efficiente al rapido cambiamento nella domanda di elettricità da parte degli utenti [1]. Questa funzionalità rende quindi disponibile una nuova tipologia di dato che permette l'apertura del sistema a nuove ed interessanti categorie di servizi. Tali dati, però, devono essere protetti. La conoscenza in tempo reale dei consumi di un utente, infatti, può rivelare informazioni sensibili riguardanti l'individuo come, ad esempio, le sue abitudini giornaliere: se ne deduce che la privacy dell'utente è un aspetto fondamentale e imprescindibile.

Da questa esigenza è nata la necessità di diminuire la granularità spaziale e temporale con cui l'informazione ricavata da tali dati viene fornita alle società che si occupano dei nuovi servizi. Ciò significa che da questa informazione non deve essere possibile risalire alla singola misura di consumo effettuata da un singolo utente in un determinato istante di tempo: un livello di privacy per l'utente così definito può essere ottenuto effettuando un'*aggregazione* spaziale e/o temporale delle misure. Per aggregazione si intende la somma di più misure: l'aggregazione temporale consiste nella somma di più misure consecutive relative a un singolo utente, mentre l'aggregazione temporale consiste nella somma di misure effettuate nello stesso istante di tempo da utenti diversi. In questo modo, le società possono trattare dell'informazione che dà un'indicazione solamente sul comportamento di un gruppo più o meno ampio di utenti e su intervalli di tempo più o meno ampi, e possono offrire differenti tipologie di servizio come, ad esempio, l'analisi statistica dei consumi di una determinata popolazione.

In questo lavoro di tesi ci siamo dapprima soffermati su due differenti architetture che permettono l'aggregazione spaziale e temporale di misure di consumo degli utenti e che consentono di rendere disponibili le misure aggregate con la granularità spaziale e temporale voluta all'entità richiedente. Queste due architetture sono chiamate *architettura centralizzata* e *architettura distribuita*, e si differenziano in base alle funzionalità delle differenti tipologie di nodi al loro interno. In particolare, l'architettura distribuita è una naturale evoluzione dell'architettura centralizzata. Per entrambe le architetture è stato definito un protocollo di comunicazione che prevede lo scambio in rete di messaggi diversi.

In base alle due architetture descritte, abbiamo realizzato due emulatori per valutare le prestazioni dei protocolli di comunicazione. Abbiamo definito

le specifiche dei protocolli ed abbiamo emulato gli scambi di messaggi tra i nodi coinvolti dalle architetture, valutando le prestazioni dei protocolli sulla base di differenti misure.

Ci siamo poi soffermati su un ulteriore importante aspetto riguardante la privacy degli utenti partecipanti all'aggregazione di misure, che prende il nome di *privacy differenziale*. Per un partecipante è garantita privacy differenziale se un attaccante, che si assume essere l'entità richiedente le misure aggregate, non è in grando di ricavare informazioni sulle singole misure di consumo dell'utente nonostante la conoscenza di *informazione ausiliaria* sull'utente stesso, ottenuta indipendentemente dalle misure aggregate. La privacy differenziale può essere garantita sommando alle misure di ogni partecipante un rumore opportunamente dimensionato, volto a mascherare la presenza del singolo utente all'interno della misura aggregata rumorosa.

Per una valutazione del livello di privacy differenziale del sistema, abbiamo elaborato due differenti modelli e ci siamo avvalsi di semplici concetti della teoria dei segnali. Il primo modello, chiamato *modello gaussiano*, ipotizza che la curva di consumo per un singolo utente possa essere modellata da un processo casuale gaussiano, mentre il secondo modello, chiamato *modello empirico*, realizza le curve di consumo per gli utenti sulla base di curve di consumo realistiche di singoli elettrodomestici. Abbiamo allora definito in che modo, nei due modelli, l'attaccante, che ha a disposizione una serie temporale di misure aggregate rumorose, può cercare di ridurre il rumore dalle misure aggregate al fine di abbattere il livello di privacy differenziale degli utenti. Oltre a questo, abbiamo studiato una possibile strategia di difesa che renda inefficiente tale attacco. Infine abbiamo valutato, per mezzo di un simulatore, le prestazioni di attacco e difesa da noi considerati, definendo uno specifico contesto ed elaborando uno specifico problema decisionale.

Questo documento è strutturato come segue. Nel Capitolo 2 viene fornita una panoramica generale sugli aspetti di privacy nelle Smart Grid e l'attuale stato dell'arte. Nel Capitolo 3 vengono ripresi i concetti teorici fondamentali per una comprensione completa del lavoro. Nel Capitolo 4 vengono illustrate nel dettaglio l'architettura centralizzata e l'architettura distribuita per l'aggregazione di misure con i relativi protocolli di comunicazione. Nel Capitolo 5 vengono descritti gli emulatori, la specifica dei protocolli e i risultati sperimentali ottenuti. Nel Capitolo 6 viene trattata la privacy differenziale sulla base dei due modelli illustrando, per ognuno di essi, un possibile attacco, una possibile difesa e le prestazioni di attacco e difesa.

CAPITOLO 2

PRIVACY NELLE SMART GRID: STATO DELL'ARTE

Per lungo tempo le aziende che forniscono servizi pubblici come l'erogazione di elettricità, acqua e gas sono state incapaci di controllare e misurare in tempo reale i consumi di questi beni, rendendo difficile una gestione efficiente delle reti di distribuzione. L'ideazione delle Smart Grid, che presuppongono la creazione di una nuova infrastruttura e l'installazione di contatori intelligenti a casa dell'utente (Smart Meter), ha completamente rimesso in discussione le relazioni tra le compagnie che si occupano dell'erogazione di questi servizi pubblici e gli utenti. Il nuovo scenario che si sta andando a configurare prevede l'inserimento in questo sistema di nuove figure che giocheranno differenti ruoli nella gestione dei servizi, nelle infrastrutture e nell'elaborazione delle informazioni, con differenti compagnie sia pubbliche che private coinvolte in questo nuovo contesto.

Lo sviluppo dei sistemi di lettura automatica dei contatori (Automatic Meter Reading, AMR) e di gestione automatica dei contatori (Automatic Meter Management, AMM) è in una fase di grande sviluppo grazie alla spinta delle organizzazioni governative in tutto il mondo, con l'obiettivo di migliorare l'efficienza globale nel consumo di energia e delle altre risorse naturali, oltre che di rimuovere barriere e vincoli nel mercato di questi servizi. In molti stati sono stati intrapresi diversi progetti, in particolare volti alla gestione della rete elettrica, che hanno dimostrato come l'AMR possa portare grandi vantaggi economici alle aziende attraverso la riduzione dei costi operazionali: l'Italia si inserisce perfettamente in questo contesto in quanto il sistema AMR italiano copre ormai il 100% degli utenti residenziali. Inoltre lo sviluppo dell'AMM, che permette una comunicazione bidirezionale tra i contatori elettronici e le piattaforme di gestione del servizio di erogazione di energia elettrica, introduce la possibilità di definire nuovi servizi basati sulle informazioni ottenute dai contatori e sul controllo remoto della domanda, sfruttando al meglio la generazione e la distribuzione di energia elettrica ottenuta dalle fonti rinnovabili.

La messa a punto di sistemi AMR e AMM determina grandi sfide dal punto di vista tecnico sui più svariati problemi come, ad esempio, l'infrastruttura di comunicazione, i protocolli di comunicazione, la messa a punto dei contatori, le piattaforme di gestione delle informazioni. Tra queste, la più critica in termini di costi è sicuramente l'infrastruttura di comunicazione: soluzioni tecniche prevedono l'utilizzo della comunicazione tramite powerline (PLC) sulle linee elettriche a bassa/media tensione oppure della comunicazione wireless sfruttando la tecnologia adottata dalle reti di sensori. Per quanto riguarda i protocolli di comunicazione, invece, sono portate avanti diverse iniziative sia da parte degli enti di standardizzazione che da parte

delle associazioni industriali.

Anche se la sicurezza delle reti è un problema conosciuto e parecchio studiato in termini di confidenzialità ed integrità dei dati, il dominio delle Smart Grid introduce nuovi problemi legati alla privacy in relazione alla protezione delle informazioni sensibili che potrebbero essere dedotte dai dati stessi dell'utente. In particolare, la sicurezza dei dati nelle reti di telecomunicazioni si focalizza sulla confidenzialità, che è un aspetto differente dalla privacy dell'utente: la prima ha a che fare con la protezione dei dati da accessi non autorizzati, mentre la seconda ha a che fare con la protezione dell'individuo e può essere estesa su differenti dimensioni. L'obiettivo principale in questo contesto è la protezione di quei dati che potrebbero rivelare informazioni sensibili riguardanti l'individuo, come ad esempio le sue caratteristiche fisiche, culturali, economiche e sociali, oppure il suo comportamento. Per questo motivo, il rispetto della privacy in questa nuova infrastruttura (Automatic Metering Infrastructure, AMI) è particolarmente rilevante nel caso di utenti domestici, e in qualche modo meno critico nel caso di utenti business i quali, comunque, potrebbero trarne beneficio.

Mettere a punto un'architettura per raccogliere le misurazioni effettuate dei contatori elettronici volta alla tutela della privacy richiede il coinvolgimento di differenti livelli di sicurezza: il trasporto sicuro dei dati sulla rete di comunicazione, la memorizzazione sicura delle misure collezionate e l'ideazione di procedure adeguate per l'accesso ai dati.

Riguardo l'infrastruttura di comunicazione, si hanno in generale i seguenti requisiti:

1. Identificazione precisa delle entità business (aziende che si occupano del servizio di erogazione del bene di consumo e terze parti che si occupano di altri tipi di servizi);

2. I dati devono essere raccolti con la granularità minima necessaria per rendere possibili le operazioni compiute dalla Smart Grid. In particolare i dati dovrebbero essere aggregati e resi anonimi, a meno che non sia strettamente necessario fare in altro modo;

3. I dati raccolti possono essere associati all'identificativo dell'utente solo nel caso sia strettamente necessario;

4. L'infrastruttura deve essere scalabile rispetto a un grande numero di contatori (100000 o più) con un tempo di invio delle misure che è nell'ordine dei minuti;

5. I dati devono essere inviati in modo affidabile. Almeno il 99% delle misure deve essere ricevuto dall'entità preposta all'utilizzo dei dati;

6. I contatori devono avere basso costo, nell'ordine dei 100$.

Pertanto, i nuovi servizi e le nuove applicazioni resi possibili dall'AMI richiedono sistemi di sicurezza nuovi ed innovativi per garantire politiche di privacy complesse e flessibili nello scenario fin qui descritto.

In accordo con i modelli di Smart Grid considerati fino ad oggi dalle autorità di regolazione e standardizzazione, l'elemento innovativo di questa nuova architettura è la piattaforma di servizi che può essere aperta ad applicazioni non solo relative alle compagnie di erogazione, ma anche a terze parti, le quali si possono inserire in un nuovo mercato portando un valore aggiunto ai dati trattati. A differenza dei sistemi tradizionali, non è soltanto la risorsa (gas, acqua, elettricità) ad avere un valore economico diretto, ma anche le informazioni relative al suo utilizzo e alla sua distribuzione. Per capire questo concetto, si può ad esempio pensare all'importanza che l'informazione sul consumo e sulla generazione distribuita dell'energia elettrica (ad esempio

mediante fonti di energia rinnovabili come i pannelli solari) può avere per operare in anticipo sul mercato dell'energia, con un risparmio di costi non indifferente; si può anche pensare all'importanza che può avere lo storico dei dati riguardo a malfunzionamenti o guasti sulla rete per ridurre i costi di mantenimento attraverso attività pianificate.

Quanto detto finora si ripercuote in termini di sicurezza sulla creazione di un'infrastruttura che permetta di accedere ai dati memorizzati ed inviati dai contatori con differenti livelli di aggregazione spaziale e temporale. Inoltre, poiché i dati raccolti possono rivelare informazioni sensibili sugli utenti anche non in relazione diretta con le misure (presenza di individui in casa, abitudini, ecc.), è importante nascondere le informazioni relative ad ogni singola utenza e all'identità dell'utente. Infine, è necessario garantire anonimato alle relazioni tra le parti coinvolte per prevenire un attacco esterno che possa permettere all'attaccante di osservare i flussi di dati attraverso l'AMI.

Questi temi sono ampiamente trattati in letteratura. Le soluzioni proposte in [2, 3] prevedono due differenti architetture di rete, una centralizzata e una distribuita, che consentono l'aggregazione spaziale e temporale delle misure dei singoli utenti, richieste da terze parti, al fine di preservarne la privacy. I termini *centralizzata* e *distribuita* si riferiscono, come vedremo più avanti, all'organizzazione dei nodi responsabili dell'aggregazione delle misure. In questo lavoro descriveremo entrambe le architetture ma ci soffermeremo prevalentemente su quella distribuita, essendo un'evoluzione di quella centralizzata, caratterizzata da una maggiore scalabilità.

Entrambe le architetture prendono spunto da [4], il quale sfrutta un approccio che prende il nome di MultiParty Computation (MPC), in cui vengono messe a punto tecniche per il calcolo di una funzione aggregata basata

su degli input che non vengono svelati ai partecipanti. Tale articolo descrive un protocollo di aggregazione che sfrutta il crittosistema di Pallier, il quale presenta proprietà omomorfe [5]. L'approccio MPC può essere distribuito su tutti gli utenti oppure considerare uno o più server. La soluzione distribuita è stata considerata in parecchie ricerche [4, 6, 7, 5], mentre la soluzione client-server, sfruttata nelle architetture da noi considerate, è stata studiata per applicazioni come l'anonimizzazione del traffico [8]. In [2, 3], la struttura del protocollo viene mantenuta invariata, ma al posto del crittosistema di Pallier si fa riferimento allo schema SSS (Shamir Secret Sharing) [9], che ha una minore complessità computazionale e permette di aggregare gli stessi dati in accordo a differenti regole di aggregazione con una crescita limitata del traffico dei messaggi in rete. L'articolo [4] adotta, come in questo lavoro, il modello di attaccante onesto-ma-curioso, in cui i nodi eseguono correttamente il protocollo, ma possono tenere traccia dei dati in ingresso e cercare di sferrare attacchi volti alla ricostruzione delle misure individuali.

L'idea di usare uno schema di condivisione come SSS, in cui un segreto (la misura) viene distribuito su più partecipanti, è presa in prestito da [8]. Tale articolo propone uno schema di aggregazione per il rispetto della privacy riguardo a misurazioni effettuate sul traffico di rete. Anche in [5] viene utilizzato uno schema di condivisione, ma in combinazione con il crittosistema di Pallier.

In [3] si fa inoltre ampio riferimento alle tecniche di instradamento distribuito definite dal protocollo p2p Chord [10].

Un ulteriore aspetto da tenere in considerazione per quanto riguarda la sicurezza dei dati degli utenti prende il nome di *privacy differenziale*. La privacy differenziale ha lo scopo di difendere le misure degli utenti anche

nel caso in cui l'attaccante possegga dell'*informazione ausiliaria* sulle misure stesse. Ad esempio, nel caso in cui l'attaccante conosca, in un determinato intervallo di tempo, l'andamento temporale del consumo energetico di un determinato utente e del consumo energetico aggregato di più utenti, viene definito un sistema di sicurezza tale per cui l'attaccante non sia in grado di stabilire se la misura dell'utente è presente nella misura aggregata o meno.

Questo aspetto viene per la prima volta trattato in [11], facendo riferimento ad un contesto generale di dati presenti in un database. In questo articolo viene definito il problema, un formalismo adeguato per la sua trattazione e una soluzione, che consiste nel sommare ai dati un rumore opportunamente dimensionato al fine di offuscarli una volta effettuata l'aggregazione.

In [12] viene definito come un aggregatore può ottenere statistiche da dati aggregati su più partecipanti senza che sia compromessa la privacy di un singolo partecipante. Come in [11], non si fa riferimento a un contesto specifico ma, a differenza di esso, viene adottato un nuovo modello di rumore e un nuovo algoritmo per la randomizzazione distribuita dei dati che garantisca la privacy differenziale sui dati aggregati, prevedendo la presenza di un sottinsieme di partecipanti malevoli che potrebbero rilevare le proprie statistiche all'aggregatore.

I concetti espressi in [11, 12] vengono ripresi in [7] e adattati al contesto delle Smart Grid. L'obiettivo dell'articolo è di definire uno schema in cui un'entità possa collezionare periodicamente misure aggregate dai contatori e derivare statistiche aggregate senza che venga rivelata alcuna informazione riguardo ai consumi di energia da parte dei singoli utenti. Viene definito un ulteriore modello di rumore e viene aggiunto un livello di robustezza al sistema. Viene anche effettuata una trattazione di come varia il livello di privacy garantito all'aumentare del numero di slot temporali considerati.

La soluzione definita da [11, 12, 7] per garantire privacy differenziale, ovvero di sommare ai singoli dati un rumore opportunamente dimesionato, porta inevitabilmente ad un'alterazione delle misure aggregate rispetto al valore che assumerebbero nel caso in cui non si volesse garantire la privacy di ogni singolo partecipante. Risulta intuitivo che, all'aumentare della privacy garantita per ogni singolo utente, diminuisce l'utilità delle misure aggregate, che rischiano di essere eccessivamente alterate, diventando inutili per qualsiasi valutazione statistica.

In [12] viene anche valutato un bound superiore per l'errore commesso sulla misura aggregata: in questo modo viene effettuata un'analisi dell'utilità della misura aggregata rumorosa ottenuta.

In [13] viene effettuato uno studio sul trade-off tra privacy e utilità nel contesto specifico delle Smart Grid, definendo come soluzione ottima il filtraggio delle componenti spettrali a bassa potenza per ogni serie temporale di misure del singolo contatore, al fine di offuscare le singole misurazioni mantenendo però la loro rilevanza statistica.

Infine, in [14] viene definito, in un contesto generale, un ulteriore modello di rumore, correlato al segnale, al fine di migliorare la privacy dei dati degli utenti senza diminuirne l'utilità. L'algoritmo per la generazione di tale rumore prende il nome di PESP (Privacy Enhanced State-dependent Perturbation).

CAPITOLO 3

ASPETTI TEORICI DI BASE

In questo Capitolo, prima di addentrarci nel lavoro da noi svolto, vengono illustrate le nozioni teoriche di base necessarie per la sua comprensione. Si fa riferimento allo schema crittografico adottato nelle architetture da noi considerate per avere garanzia di privacy da parte dell'utente, al protocollo di look-up Chord e a concetti chiave sulla privacy differenziale.

3.1 Schema a soglia di Shamir

Lo schema a soglia di Shamir (Shamir Secret Sharing, SSS) viene proposto per la prima volta in [9]. Questo schema viene utilizzato per dividere un segreto in w parti chiamate quote o *share*. Ogni share viene consegnata ad un differente partecipante e, per recuperare il segreto, è necessaria la cooperazione di un qualsiasi sottoinsieme $t \leq w$ di partecipanti. Il recupero del segreto è impossibile per sottoinsiemi di cardinalità inferiore a t. Formalmente, uno schema a soglia così definito prende il nome di *schema* (t, w) [15].

Lo schema a soglia di Shamir si basa sull'estensione del semplice concetto di algebra lineare per cui sono necessari due punti per determinare una retta, tre punti per determinare una parabola e così via. Lo schema nasce in un contesto di *aritmetica modulare*: viene scelto un numero primo q più grande di qualsiasi segreto e più grande del numero di partecipanti w. Tutte le operazioni sono svolte mod q. Il segreto M è rappresentato da un numero mod q ($M \in Z_q$), che si vuole dividere tra w entità in modo tale che t di essere siano necessarie per ricostruire il messaggio.

In primo luogo devono essere scelti in modo casuale $t-1$ interi mod p, chiamati $\rho_1, \rho_2, \ldots, \rho_{t-1}$. Il polinomio

$$\rho(x) \equiv M + \rho_1 x + \rho_2 x^2 + \ldots + \rho_{t-1} x^{t-1} \bmod q \tag{3.1}$$

gode della proprietà che $\rho(0) \equiv M \pmod q$. Per ognuno dei w partecipanti scegliamo degli interi distinti $x_1, x_2, \ldots, x_w \pmod q$ e diamo a ciascun partecipante una coppia (x_i, y_i) con $y_i \equiv \rho(x_i) \pmod q$. Una scelta possibile per le x può essere $1, 2, \ldots, w$, pertanto distribuiremo le coppie $(1, \rho(1)), \ldots, (w, \rho(w))$, una per ogni partecipante. Si noti che il numero primo q è noto a tutti, ma il polinomio $\rho(x)$ è segreto. Supponiamo che t persone collaborino e mettano in comune le proprie coppie, supponendo per semplicità di notazione che siano le coppie $(x_1, y_1), \ldots, (x_t, y_t)$. Per ricostruire il messaggio M si può scrivere un sistema lineare. Il polinomio $\rho(x)$, infatti, ha grado $t-1$, pertanto

$$y_k \equiv M + \rho_1 x_k + \rho_2 x_k^2 + \ldots + \rho_{t-1} x_k^{t-1} \bmod q, \quad 1 \leq k \leq t$$

Se indichiamo $\rho_0 = M$, possiamo scrivere

$$\begin{pmatrix} 1 & x_1 & \cdots & x_1^{t-1} \\ 1 & x_2 & \cdots & x_2^{t-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_t & \cdots & x_t^{t-1} \end{pmatrix} \begin{pmatrix} \rho_0 \\ \rho_1 \\ \vdots \\ \rho_{t-1} \end{pmatrix} \equiv \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_t \end{pmatrix} (\mathrm{mod} q)$$

Il sistema ha soluzione unica mod q se il determinante della matrice è diverso da zero mod q. La matrice scritta sopra è una matrice di Vandermonde. Si può dimostrare che tale matrice ha la proprietà di avere determinante nullo mod q soltanto se due degli x_i coincidono mod q. Quindi, finchè gli x_k sono distinti, il sistema ha una sola soluzione e può essere ricavato, invertendo la matrice, il vettore delle ρ, tra cui $\rho_0 = M$.

Un metodo alternativo per il recupero di M prende il nome di *interpolazione di Lagrange*. Per ricostruire il segreto è sufficiente applicare la formula

$$M \equiv \sum_{k=1}^{t} y_k \prod_{j=1, j\neq k}^{t} \frac{-x_j}{x_k - x_j} \bmod q.$$

L'interpolazione di Lagrange permette il recupero di M senza dover calcolare il vettore delle ρ con un notevole risparmio dal punto di vista computazionale.

E' importante notare che lo schema a soglia di Shamir gode della proprietà di *cifratura omomorfa* rispetto all'operazione di addizione [4]. La cifratura omomorfa rappresenta un gruppo di funzioni che permettono di effettuare determinate operazioni algebriche sul testo cifrato piuttosto che sul testo in chiaro.

Una funzione di cifratura $E_k(\cdot)$ è omomorfa rispetto all'addizione se, dati due messaggi $x, y \in Z_q$, vale la proprietà

$$E_k(x + y) = E_k(x) + E_k(y)$$

senza che siano conosciuti i due messaggi x, y o la chiave privata k.

Questa proprietà, relativamente al sistema a soglia di Shamir, assume un importante significato. Infatti, effettuare la somma di più segreti e suddividerla in w share è equivalente a suddividere singolarmente ognuno dei segreti in w share per mezzo di polinomi random diversi e successivamente effettuare la somma delle share corrispondenti, ovvero ottenute sostituendo nei differenti polinomi random lo stesso x_i. Questo rende possibile che i partecipanti aggreghino più segreti senza conoscere il loro valore in chiaro, in quanto ogni partecipante effettua la somma esclusivamente sulle share da esso conosciute. La cooperazione di $t \leq w$ partecipanti consente di recuperare la somma dei segreti, senza che i singoli segreti sommati siano rivelati.

3.1.1 Algoritmo di Berlekamp-Welch

L'algoritmo di Berlekamp-Welch [16] è un algoritmo per la correzione d'errore efficiente. Esso permette di recuperare un polinomio di grado $t-1$, come il polinomio 3.1, anche in presenza di share errate. Questo significa che il messaggio M può essere recuperato anche nel caso di partecipanti malevoli che alterano il proprio valore numerico della share y_i. Infatti M coincide con il termine noto del polinomio recuperato. Il recupero può essere effettuato per un numero massimo di share errate pari a e, con

$$e < \frac{w - t + 1}{2}$$

supponendo di avere a disposizione w share.

Consideriamo il polinomo $\rho(x)$ illustrato nell'equazione 3.1. Conoscendo le coppie (x_i, y_i), con in generale $y_i = \rho(x_i)$, ma sapendo che al massimo un numero pari ad e di y_i sono valori errati, ovvero non ottenuti dalla valutazione

del polinomio 3.1, l'obiettivo è il recupero del polinomio $\rho(x)$ corretto, che rivela $\rho_0 = M$ e permette il recupero del messaggio M.

L'algoritmo innanzitutto prevede la correzione del polinomio bivariato

$$Q(x, y) = f_0(x) - f_1(x) \cdot y$$

dove f_0 è un polinomio di grado massimo $2(t-1)$ ed f_1 è un polinomio di grado massimo $t-1$ a coefficienti modulari mod q. Viene imposta la condizione $f_1(0) = 1$. I coefficienti modulari $f_{0,i}$ e $f_{1,i}$ sono le variabili che vogliamo determinare. Grazie al grado massimo definito per i due polinomi e alla condizione $f_1(0) = 1$, si vede che il numero di variabili da determinare è pari a

$$v = (2(t-1)+1) + ((t-1)+1) - 1 = 3t - 2$$

Sostituendo in $Q(x, y)$ i valori di x_i e y_i si ottiene una equazione lineare rispetto ai coefficienti dei polinomi $f_{0,i}$ e $f_{1,i}$. Supponendo, come detto, di avere a disposizione w share, si ottiene un numero di equazioni lineari pari a w. Scegliendo da queste un sottoinsieme di v equazioni, corrispondenti a v share, è possibile risolvere il sistema lineare. E' necessario, quindi, che $w \geq v$. Una volta determinati i coefficienti modulari $f_{0,i}$ e $f_{1,i}$ e quindi i polinomi f_0 e f_1, si può calcolare il polinomio cercato mediante la divisione tra polinomi mod q

$$\rho(x) = \frac{f_0(x)}{f_1(x)} \bmod q$$

A questo punto, è stato recuperato correttamente $M = \rho_0$.

Questo metodo funziona bene anche nel caso in cui non è prevista l'alterazione da parte dei partecipanti delle share a loro disposizione ma nel caso di errori, ad esempio di trasmissione, che non consentono di avere a disposizione tutte le w share. In questo caso si pone $y_i = 0$ per il sottinsieme e di share mancanti, ed il recupero robusto è possibile se $e < \frac{w-t+1}{2}$.

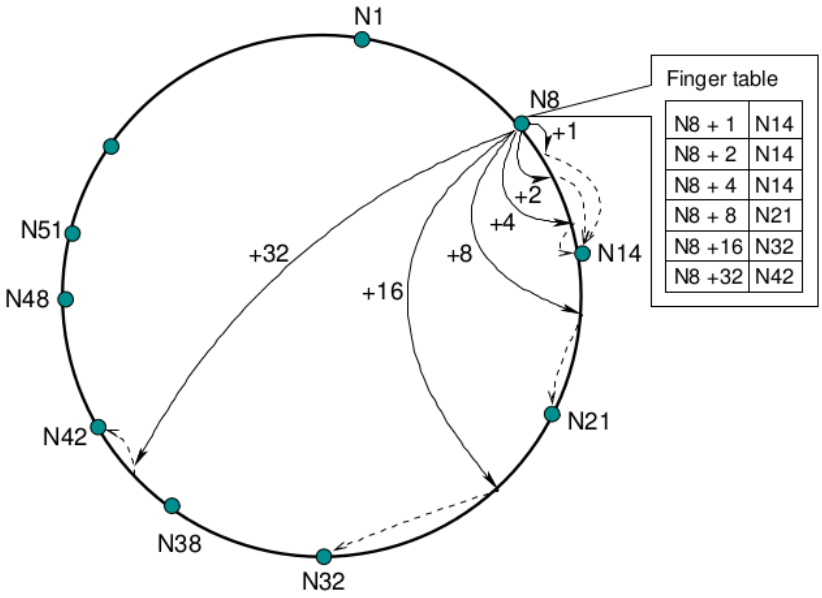

Figura 3.1: Esempio di anello Chord e di finger table per un nodo

3.2 Chord

Chord [10] è un protocollo di look-up peer-to-peer distribuito che permette ad un nodo di localizzare in modo efficiente il nodo responsabile di una determinata chiave, la quale rappresenta un determinato oggetto. Nella nostra trattazione, quando nella Sezione 4.2 e nella Sezione 5.3 descriveremo l'architettura distribuita e la specifica dei messaggi in tale architettura, faremo ampio riferimento a Chord. Esso, infatti, prevede un *algoritmo di instradamento* in cui il numero di messaggi scambiati è lineare rispetto al numero di nodi. Il protocollo Chord necessita che ogni nodo sia caratterizzato da un identificatore univoco (Chord ID) di lunghezza k bit. I Chord ID vengono quindi ordinati su un *anello* modulo 2^k, che prende il nome di *anello Chord* (Figura 3.1).

L'algoritmo di instradamento prevede che ogni nodo n mantenga una tabella di instradamento con k righe, che prende il nome di *finger table*. La

i-sima riga della tabella al nodo n contiene il Chord ID del primo nodo s che succede n di almeno 2^{i-1} (modulo 2^k) sull'anello Chord con $1 \leq i \leq k$. Questo significa calcolare $s = successor(n + 2^{i-1}) \bmod 2^k$, dove *successor* indica il successore, ovvero il primo nodo presente sull'anello dopo $n + 2^{i-1} \bmod 2^k$. Formalmente chiamiamo s la i-sima *finger* del nodo n, e la definiamo come $n.finger[i]$. Si noti che la prima finger di n è l'immediato successore di n sull'anello, quindi definiamo come successore, per un nodo n, $successor = finger[1].node$.

Questo schema ha una caratteristica importante: ogni nodo mantiene l'informazione, per mezzo della finger table, solamente di un piccolo numero di altri nodi, ed ha maggiori informazioni rispetto ai nodi più vicini a sè piuttosto che ai nodi più lontani.

3.3 La privacy differenziale

Ipotizziamo che un aggregatore sia in grado di ottenere la somma di dati provenienti da più utenti con lo scopo di effettuare delle misure statistiche sui dati aggregati, ma che si voglia proteggere la privacy dei singoli partecipanti, soprattutto se l'aggregatore non è considerato attendibile. Supponiamo che l'aggregatore sia in grado di ottenere dell'*informazione ausiliaria* che prescinde dalla conoscenza dei dati aggregati. L'obiettivo della privacy differenziale è evitare che un utente che patecipa alla somma di dati aggregati possa vedere ridotta la propria privacy nonostante la presenza di questa informazione ausiliaria [11]. L'informazione ausiliaria può essere ottenuta da database pubblici, conoscenze personali su ogni partecipante o grazie alla collusione con un sottinsieme ridotto di partecipanti. Il concetto chiave che permette di garantire la privacy differenziale per un utente è che le statistiche

rivelate all'aggregatore non siano eccessivamente influenzate dalla presenza o meno di uno specifico partecipante. Per raggiungere questo scopo viene scelta come soluzione di sommare del rumore casuale ai dati disaggregati al fine di introdurre un grado di casualità ai dati stessi [11, 12, 7]. Tale rumore può essere definito in differenti modi.

Definiamo il problema seguendo la stessa notazione scelta in [12]. Supponiamo di avere un aggregatore e n partecipanti, numerati $1, \ldots, n$. Chiamiamo $[n] := \{1, 2, \ldots, n\}$. In ogni istante di tempo $t \in N$ ogni partecipante $i \in [n]$ presenta un valore $x_{i,t} \in Z_q$ con q numero primo. Ci soffermiamo su un istante di tempo specifico e omettiamo l'indice temporale t, scrivendo x_i. Definiamo $\boldsymbol{x} = (x_1, \ldots, x_n)$ il vettore dei valori di ogni partecipante in un determinato istante di tempo. Il valore $\text{sum}(\boldsymbol{x}) = \sum_1^n x_i$ è il dato aggregato a cui l'aggregatore è interessato, che assume valori in un dominio $\mathcal{O}$. Per garantire privacy a ogni singolo partecipante anche in presenza di informazione ausiliaria sul partecipante stesso, ognuno di essi genera un rumore casuale indipendente su un determinato dominio Ω, rappresentato dal vettore $\boldsymbol{r} = (r_1, \ldots, r_n) \in \Omega^n$. Ogni partecipante ottiene una versione rumorosa del proprio dato semplicemente sommando il rumore generato al dato stesso, ottenendo complessivamente il vettore $\hat{\boldsymbol{x}} = (\hat{x_1}, \ldots, \hat{x_n}) = (x_1 + r_1, \ldots, x_n + r_n)$. All'aggregatore viene fornita

$$\text{sum}(\hat{\boldsymbol{x}}) = \sum_1^n \hat{x_i} = \sum_1^n (x_i + r_i) = \sum_1^n x_i + \sum_1^n r_i \bmod q$$

la quale deve essere il più simile possibile a $\text{sum}(\boldsymbol{x})$ per non compromettere l'utilità del dato aggregato. E' richiesto, come già detto, che la partecipazione di un utente all'aggregazione non riveli informazioni sull'utente stesso. Questo significa che la somma aggregata $\text{sum}(\hat{\boldsymbol{x}})$ deve essere all'incirca la

stessa, sia che un particolare utente partecipi o meno al sistema. Ipotizziamo inoltre la presenza nel sistema di utenti compromessi che colludono con l'aggregatore rivelando la propria misura o la statistica del rumore aggiunto all'aggregatore. In questo caso, è necessario assicurare che i rimanenti utenti non compromessi aggiungano tutti un rumore sufficiente a garantire la propria privacy, che sarà maggiore rispetto al caso in cui nel sistema non siano presenti utenti compromessi.

L'approccio che prevede l'aggiunta di rumore casuale da parte di ogni partecipante per garantire un rumore aggregato sufficiente a preservare la privacy di ogni singolo utente prende il nome di *privacy differenziale distribuita* (DD-Privacy) [17]: formalizziamo questa nozione.

Dato un sottinsieme K di partecipanti, chiamiamo $\boldsymbol{r_K} = \{r_i : i \in K\}$ e $\bar{K}$ il complementare di K. Si richiede che in ogni istante di tempo $t \in N$ sia applicata la seguente definizione di privacy.

Definizione 3.1. (ϵ, δ)-DD-Privacy. *Supponiamo* $\epsilon > 0$, $0 \leq \delta < 1$ *e* $0 < \gamma \leq 1$. *Diciamo che il processo di randomizzazione ottenuto sommando i rumori* r_i *ai dati* x_i *garantisce* (ϵ, δ)*-DD-Privacy in presenza di una frazione* γ *di partecipanti non compromessi se si verifica la seguente condizione. Per ogni coppia di vettori vicini* $\boldsymbol{x}$ *e* $\boldsymbol{y}$*, per ogni sottinsieme* $S \subseteq \mathcal{O}$ *e per ogni sottinsieme* $\bar{K}$ *di partecipanti non compromessi di dimensione almeno* γn*,*

$$\Pr[\text{sum}(\hat{\boldsymbol{x}}) \in S \mid \boldsymbol{r_K}] \leq e^{\epsilon} \cdot \Pr[\text{sum}(\hat{\boldsymbol{y}}) \in S \mid \boldsymbol{r_K}] + \delta$$

In questa definizione, due vettori $\boldsymbol{x}$ e $\boldsymbol{y}$ sono *vicini* se differiscono esattamente di un solo elemento. Questo corrisponde esattamente al caso in cui un singolo utente cambi il proprio dato. Quando K è l'insieme dei nodi compromessi, tale definizione richiede che la casualità introdotta dai restanti nodi onesti sia sufficiente per garantire la privacy differenziale. Per questo motivo, la probabilità è condizionata dal vettore $\boldsymbol{r_K}$.

In parole semplici, un algoritmo che garantisce (ϵ, δ)-DD-Privacy produce in output dei valori indistinguibili per input simili (nello specifico, per input che variano di un solo elemento) e quindi la modifica del dato di un qualsiasi utente sull'insieme globale dei dati cambia la probabilità di avere un output differente di un fattore al massimo pari al fattore moltiplicativo e^{ϵ}, con una correzione dovuta al fattore di somma δ. I parametri ϵ e δ, quindi, permettono di controllare il livello di privacy. Valori piccoli per ϵ e δ implicano un livello di privacy maggiore, in quanto restrigono l'influenza del dato di un utente sull'output.

3.3.1 La sensitività

La *sesitività* [11] è un paramentro fondamentale per il dimensionamento del rumore al fine di garantire (ϵ, δ)-DD-Privacy. L'ampiezza del rumore, infatti, viene scelta direttamente proporzionale alla sensitività Δ, la quale non è altro che il massimo cambiamento a cui un singolo partecipante può portare sull'output, ovvero sulla somma aggregata $\text{sum}(\boldsymbol{x}) = \sum_{i=1}^{n} x_i$.

Definizione 3.2. Sensitività. *Per una qualsiasi funzione f, la sensitività di f è pari a:*

$$\Delta f = \max_{D1,D2} |f(D_1) - f(D_2)|$$

per tutti gli insiemi D_1 e D_2 che differiscono al massimo di un elemento.

Tale definizione, adottata nel nostro caso specifico, in cui la funzione f è la funzione di somma sum e gli insiemi D_1 e D_2 non sono altro che due vettori *vicini* $\boldsymbol{x}$ e $\boldsymbol{y}$, può essere così scritta:

$$\Delta = \max_{\boldsymbol{x},\boldsymbol{y}} |\,\text{sum}(\boldsymbol{x}) - \text{sum}(\boldsymbol{y})|$$

In generale, più la sensitività Δ è piccola meglio le tecniche per garantire privacy differenziale funzionano, poiché è possibile garantire un livello di

privacy voluto con un'ampiezza di rumore inferiore, senza compromettere l'utilità dei dati aggregati.

E' importante notare che la sensitività è una proprietà che dipende esclusivamente dalla funzione f adottata, e non è influenzata in alcun modo dall'insieme dei dati considerati.

3.3.2 Modello di rumore

Dopo aver illustrato gli aspetti fondamentali della privacy differenziale, è necessario definire quale modello è adottato per il rumore, ovvero come vengono scelti i valori di $\boldsymbol{r} = (r_1, \ldots, r_n)$. Come già accennato, se si suppone che una frazione dei partecipanti sia compromessa e colluda con l'aggregatore, è necessario che i rimanenti partecipanti onesti distribuiscano la generazione del rumore tra di loro. Si ipotizza che ogni partecipante conosca a priori una stima per difetto della frazione γ di partecipanti onesti. Questo è necessario perchè si suppone che ogni partecipante generi del rumore che dipende da γ. Il modello di rumore che consideriamo, definito e formalizzato in [12], garantisce che con alta probabilità la somma aggregata accumuli un rumore sufficiente dai partecipanti onesti mantenendo l'errore rispetto alla somma aggregata non rumorosa basso, senza compromettere la sua utilità.

In letteratura [11, 7] viene comunemente aggiunto un rumore campionato da una distribuzione di Laplace. [12], invece, sceglie di utilizzare una *distribuzione geometrica simmetrica*. Essendo la distribuzione geometrica simmetrica illimitata e considerando che stiamo lavorando con gruppi discreti, c'è il rischio che tale rumore possa superare la dimensione del gruppo. Il modello di rumore che consideriamo assicura che la probabilità di un evento simile sia piccola. La distribuzione geometrica simmetrica può essere vista come un'approssimazione discreta della distribuzione di Laplace.

Definizione 3.3. Distribuzione geometrica simmetrica. *Sia* $\alpha > 1$. *Chiamiamo* $\mathrm{Geom}(\alpha)$ *la distribuzione geometrica simmetrica che assume valori interi tali per cui la densità di probabiltà in* k *sia pari a* $\frac{\alpha-1}{\alpha+1} \cdot \alpha^{-|k|}$.
Chiamiamo $\mathrm{Geom}^+(\alpha)$ *la distribuzione geometrica unilaterale che assume valori interi positivi tali per cui la densità di probabilità in* k *sia pari a* $(\alpha - 1) \cdot \alpha^{-k}$.

Si può dimostrare che se alla misura viene sommato un rumore r ottenuto da una distribuzione geometrica simmetrica con $\alpha = \exp(\frac{\epsilon}{\Delta})$, con Δ definito nella Sezione 3.3.1, allora si ha garanzia di ϵ-differential privacy. L'obiettivo, quindi, è assicurare che se almeno γn partecipanti sono onesti e non compromessi in output si ottenga un accumulo di rumore di ampiezza simile. Vediamo in che modo ottenere questo risultato.

Considerato il vettore $\boldsymbol{x} = (x_1, \ldots, x_n)$ dei dati generati dai partecipanti in un determinato periodo e $\alpha = \exp(\frac{\epsilon}{\Delta})$, definiamo la probabilità $\beta = \frac{1}{\gamma n} \log \frac{1}{\delta}$. Questa probabilità diminuisce all'aumentare del numero di partecipanti onesti nel sistema.

Ogni partecipante $i \in [n]$ estrae un valore r_i in accordo alla seguente distribuzione:

$$r_i \leftarrow \begin{cases} \mathrm{Geom}(\alpha) & \text{con probabilità } \beta \\ 0 & \text{con probabilità } 1 - \beta \end{cases}$$

Il dato rumoroso è quindi ottenuto calcolando $\hat{x_i} = x_i + r_i \bmod q$

Tale modello di rumore non solo garantisce la privacy differenziale, ma assicura che il rumore accumulato dall'output sia limitato in modo tale che l'errore rispetto all'output non rumoroso sia piccolo. Infatti il meccanismo garantisce (ϵ, δ)-DD-Privacy con un errore approssimativamente dell'ordine

$O(\frac{\Delta}{\epsilon}\sqrt{\frac{1}{\gamma}})$ indipendente dal numero di partecipanti, quando un rumore accumulato di ampiezza $\Theta(\frac{\Delta}{\epsilon})$ è necessario per garantire privacy differenziale [11].

Per concludere, è necessario definire come operativamente è possibile ottenere una variabile casuale intera di rumore r_i da una distribuzione geometrica simmetrica $\text{Geom}(\alpha)$ a partire dalla distribuzione geometrica unilaterale $\text{Geom}^+(\alpha)$.

Definizione 3.4. *Sia $\alpha > 1$. G è una variabile casuale avente distribuzione* $\text{Geom}(\alpha)$ *e W sia una variabile casuale avente distribuzione* $\text{Geom}^+(\alpha)$. *Segue che r_i e G possono essere ottenuti nel seguente modo.*

$$r_i := \begin{cases} G & \textit{con probabilità } \beta \\ 0 & \textit{con probabilità } 1-\beta \end{cases}$$

$$G := \begin{cases} 0 & \textit{con probabilità } \frac{\alpha-1}{\alpha+1} \\ W & \textit{con probabilità } \frac{1}{\alpha+1} \\ -W & \textit{con probabilità } \frac{1}{\alpha+1} \end{cases}$$

La distribuzione geometrica simmetrica presenta una varianza pari a

$$\sigma^2_{\text{Geom}(\alpha)} = \frac{2\alpha}{(\alpha-1)^2}$$

e valor medio nullo ($\mu_{\text{Geom}(\alpha)} = 0$).

Il rumore, però, è aggiunto da un partecipante solamente con probabilità β. Questo significa che media e varianza equivalente di rumore per ogni singolo partecipante dipendono da β.

L'aggiunta o meno di rumore con probabilità β può essere modellata da una variabile casuale y distribuita secondo la distribuzione di Bernoulli: tale variabile casuale assume valore 1 con probabilità β e valore 0 con probabilità

$1-\beta$. Media e varianza della variabile y valgono $\mathrm{E}[y] = \beta$ e $\mathrm{var}[y] = \beta(1-\beta)$. La variabile casuale r che indica l'ampiezza di rumore aggiunto, invece, è distribuita come già detto secondo la distribuzione Geom(α) e la sua media e varianza valgono $\mathrm{E}[r] = 0$ e $\mathrm{var}[r] = \frac{2\alpha}{(\alpha-1)^2}$.

E' necessario allora calcolare media e varianza del prodotto delle due variabili casuali $y \cdot r$ per ottenere media e varianza equivalente di rumore aggiunto da ogni singolo partecipante. Essendo le variabili casuali y e r incorrelate, si può scrivere

$$\mu_{eq} = \mathrm{E}[r \cdot y] = \mathrm{E}[r] \cdot \mathrm{E}[y] = 0$$

Il valore medio del prodotto delle due variabili aleatorie è nullo, essendo nulla $\mathrm{E}[r]$.

Per quanto riguarda la varianza, invece, vale la relazione

$$\begin{aligned}
\sigma^2_{eq} = \mathrm{var}[r \cdot y] &= \mathrm{E}^2[r] \cdot \mathrm{var}[y] + \mathrm{E}^2[y] \cdot \mathrm{var}[r] + \mathrm{var}[y] \cdot \mathrm{var}[r] = \\
&= \mathrm{E}^2[y] \cdot \mathrm{var}[r] + \mathrm{var}[y] \cdot \mathrm{var}[r] = \\
&= \beta^2 \cdot \frac{2\alpha}{(\alpha-1)^2} + \beta(1-\beta) \cdot \frac{2\alpha}{(\alpha-1)^2} = \\
&= \beta^2 \cdot \frac{2\alpha}{(\alpha-1)^2} + \beta \cdot \frac{2\alpha}{(\alpha-1)^2} - \beta^2 \cdot \frac{2\alpha}{(\alpha-1)^2} = \\
&= \beta \cdot \frac{2\alpha}{(\alpha-1)^2}
\end{aligned}$$

Ciò significa che il valore equivalente di rumore aggiunto da ogni singolo utente è β volte più piccolo del rumore realmente aggiunto dagli utenti che aggiungono del rumore.

E' possibile infine calcolare media e varianza del rumore totale sui dati aggregati. Se n sono i partecipanti, il valor medio del rumore totale sarà la somma dei valori medi delle variabili aleatorie che rappresentano il rumore

equivalente. Avendo queste variabili tutte lo stesso valor medio pari a μ_{eq}, si ottiene

$$\mu_{tot} = n \cdot \mu_{eq} = 0$$

Il rumore complessivo avrà quindi valor medio nullo.

Per quanto riguarda la varianza, sappiamo che le variabili aleatorie di rumore sono indipendenti tra i differenti partecipanti. Questo significa che la varianza della somma delle variabili aleatorie che rappresentano il rumore equivalente è pari alla somma delle varianze, ed avendo tali variabili aleatorie tutte la medesima varianza pari a σ^2_{eq} si ottiene

$$\sigma^2_{tot} = n \cdot \sigma^2_{eq} = n \cdot \beta \cdot \frac{2\alpha}{(\alpha - 1)^2}$$

3.3.3 La cross-correlazione

La *cross-correlazione* [18] è una funzione che misura la similarità tra due segnali continui o discreti. Nella nostra trattazione faremo riferimento a serie di campioni temporali e quindi a segnali discreti: ci riferiamo alla sua definizione in questo caso specifico. La cross-correlazione tra due segnali discreti x_n e y_n è definita come

$$R_{xy}[m] = \sum_{m=-\infty}^{m=+\infty} x_{n+m} \cdot y^*_n$$

dove l'asterisco all'esponente indica il *complesso coniugato*.

Caso particolare di cross-correlazione è la cross-correlazione di un segnale con se stesso: in questo caso si parla di *autocorrelazione*. L'autocorrelazione di un segnale x è definita come

$$R_x[m] = \sum_{m=-\infty}^{m=+\infty} x_{n+m} \cdot x^*_n$$

Effettuare la cross-correlazione tra due segnali o l'autocorrelazione di un segnale consiste operativamente nel considerare uno dei due segnali traslato ed effettuare la somma dei prodotti di tutti i campioni corrispondenti in base alla traslazione considerata. Per ogni valore di traslazione pari a m si ottiene l'm-simo campione della crosscorrelazione (o autocorrelazione). In base a ciò, l'autocorrelazione presenta sempre un valore di picco nel caso in cui $m = 0$, poiché in questa particolare situazione il segnale è completamente sovrapposto con se stesso.

CAPITOLO 4

ARCHITETTURE DI RETE

4.1 Architettura centralizzata

L'architettura e il protocollo di comunicazione di riferimento sono ampiamente trattati in [2]. In questa Sezione vengono ripresi e dettagliatamente descritti. Una trattazione specifica dell'architettura centralizzata è necessaria soprattutto per comprendere il funzionamento dell'architettura distribuita, che sarà illustrata nella Sezione 4.2.

Con riferimento alla Figura 4.1, l'achitettura prevede la presenza di tre insiemi di nodi:

- L'insieme $\mathcal{P}$ dei *Producer*, che rappresentano i contatori;
- L'insieme $\mathcal{N}$ dei *PPN*, che sono i nodi adibiti all'aggregazione nel dominio crittografato;

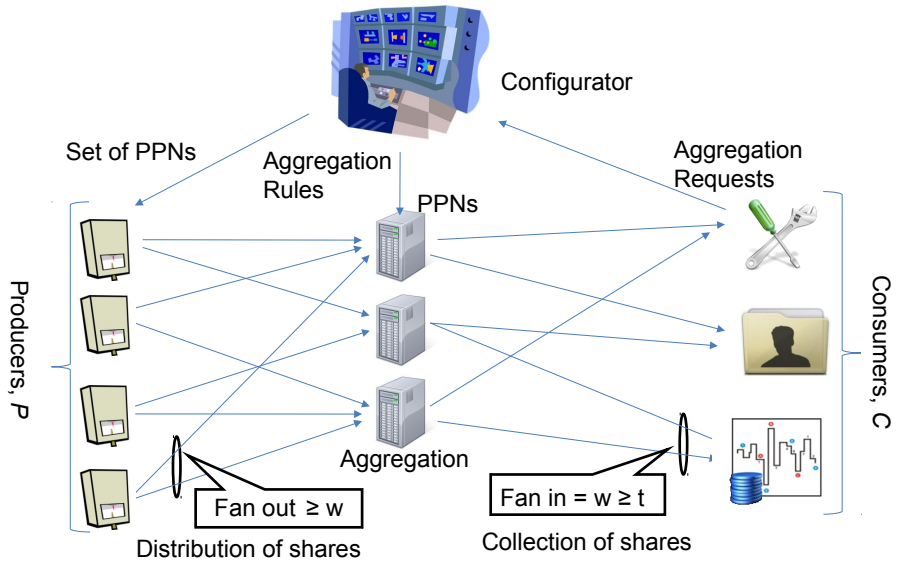

Figura 4.1: L'architettura centralizzata

- L'insieme $\mathcal{C}$ dei *Consumer*, i quali ricevono informazioni aggregate spazialmente e/o temporalmente e rappresentano le società di erogazione della risorsa o società terze (ad esempio le compagnie di fatturazione oppure istituti di statistica).

E' presente poi nell'architettura un nodo che prende il nome di *Configurator*: esso ha il compito di controllare che le richieste di aggregazione da parte dei Consumer siano conformi alle politiche di privacy del sistema e, in caso affermativo, di configurare i PPN con le corrette regole di aggregazione. Le misure effettuate da ogni Producer sono divise in più share secondo lo schema SSS (Shamir Secret Sharing), descritto nel Capitolo 3, e possono essere recuperate solamente se almeno $t \leq w$ share sono disponibili, con w numero totale di share in cui le misure individuali sono divise. I Producer, come si evince dalla Figura 4.1, inviano ogni share a un differente PPN: questo significa che le misure individuali possono essere recuperate se si ha collusione di almeno t PPN, portando ad una violazione della privacy.

Per aggregare i dati, ogni PPN indipendentemente dagli altri somma le share ottenute da differenti Producer e/o dallo stesso Producer in istanti di tempo differenti e invia le share così sommate, che prendono il nome di share aggregata, al Consumer, il quale recupera le misure aggregate per mezzo di un algoritmo di recupero, che può essere l'interpolazione di Lagrange oppure l'algoritmo di Berlekamp-Welch (vedere Capitolo 3) in cui è prevista la presenza di PPN malevoli che alterano le share aggregate. Nel primo caso, il Consumer può recuperare le misure aggregate solo se almeno t PPN hanno effettuato l'aggregazione correttamente sulle share inviate dall'insieme di Producer specificato dal Consumer. Nel secondo caso il Consumer può recuperare le misure aggregate anche se un numero $e < \frac{w-t+1}{2}$ di PPN altera il valore della propria share aggregata.

Grazie alle proprietà omomorfe dello schema di Shamir rispetto all'addizione, le share aggregate ottenute dalla procedura descritta sopra sono equivalenti a delle share ottenute prima effettuando l'aggregazione delle misure individuali e successivamente cifrando nel dominio crittografato.

Si considera un modello di attaccante *onesto-ma-curioso*: si assume che i PPN e i Consumer seguano il protocollo, ma cerchino di dedurre informazioni supplementari sulle singole misure tenendo traccia di tutti i dati che ricevono effettuando operazioni per cercare di recuperare i valori delle misure non aggregate (nel caso di PPN) o dei dati aggregati con una granularità minore oppure generati da un sottinsieme dei Producer monitorati (nel caso di Consumer). Nello specifico, due o più Consumer che colludono possono ottenere misure individuali o aggregate risultanti da una combinazione lineare delle misure conosciute da quell'insieme di Consumer. Per esempio, le misure individuali del Producer p possono essere ottenute dalle misure aggregate di due Consumer collusi, i cui set di Producer monitorati differiscono

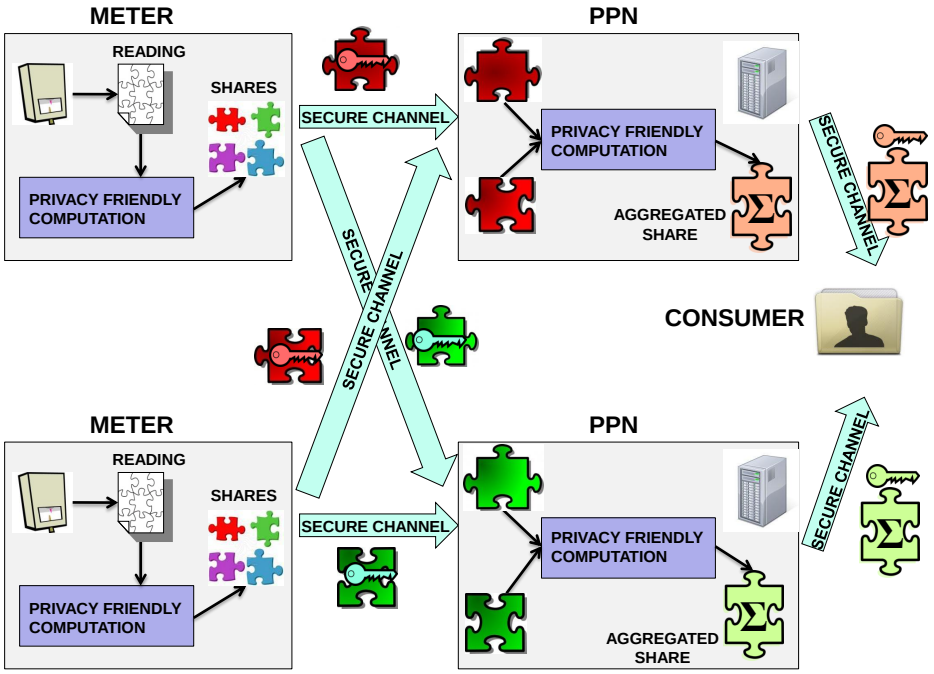

Figura 4.2: Il protocollo

solo per la presenza o meno di p, semplicemente calcolando la differenza tra i due valori aggregati. Il ruolo del Configurator è proprio di evitare che i Consumer possano definire regole di aggregazione che possano portare a una violazione della privacy degli utenti. Dato che nessun sottoinsieme di PPN con cardinalità inferiore a t può recuperare nè le misure individuali nè quelle aggregate, il processo di aggregazione descritto è sicuro sotto l'assunzione di un modello *onesto-ma-curioso*.

Si assume infine che i canali di comunicazione tra i Producer e i PPN e tra i PPN e i Consumer siano confidenziali e autenticati (Figura 4.2) in modo tale da rendere l'architettura computazionalmente sicura contro attacchi portati a termine da osservatori passivi, che potrebbero collezionare più share inviate da un dato Producer e recuperare le misure individuali.

4.1.1 Il protocollo di comunicazione

Assumiamo che il tempo sia suddiviso in *round* di durata fissa nell'ordine dei minuti e che tutti i nodi siano sincronizzati. Ogni Producer, PPN e Consumer è identificato da un numero differente. Il protocollo di comunicazione è suddiviso in due fasi: la prima è la fase di setup avviata dal Consumer e coinvolge il Consumer, il Configurator e i PPN; la seconda fase è eseguita ad ogni round e coinvolge i Producer, i PPN e il Consumer. Questa fase gestisce l'aggregazione spaziale e temporale e il recupero delle misure aggregate. Nella Figura 4.3 è illustrato lo scambio dei messaggi. Le lettere f, p, n e c indicano rispettivamente il Configurator, il Producer, il PPN e il Consumer coinvolti nella comunicazione.

Durante la fase di configurazione sono scambiati i seguenti messaggi:

1. SpecifyAggregationRule

   $$c \rightarrow f : \Pi_c \| k_c$$

   Il Consumer c specifica una regola di aggregazione in termini di: (1) insieme dei Producer che il Consumer vuole monitorare, Π_c, e (2) numero di intervalli di tempo su cui i dati devono essere aggregati (finestra di aggregazione) k_c. La regola di aggregazione (Π_c, k_c) è inviata al Configurator. Senza perdita di generalità, si assume che ogni Consumer specifichi una sola regola di aggregazione.

2. ConfigurePPN

   $$f \rightarrow n : \Pi_c \| k_c \| R_c$$

   Il Configurator controlla la conformità della regola di aggregazione alla politica del sistema, seleziona un insieme Ω_c di $w \geq t$ PPN e comunica a

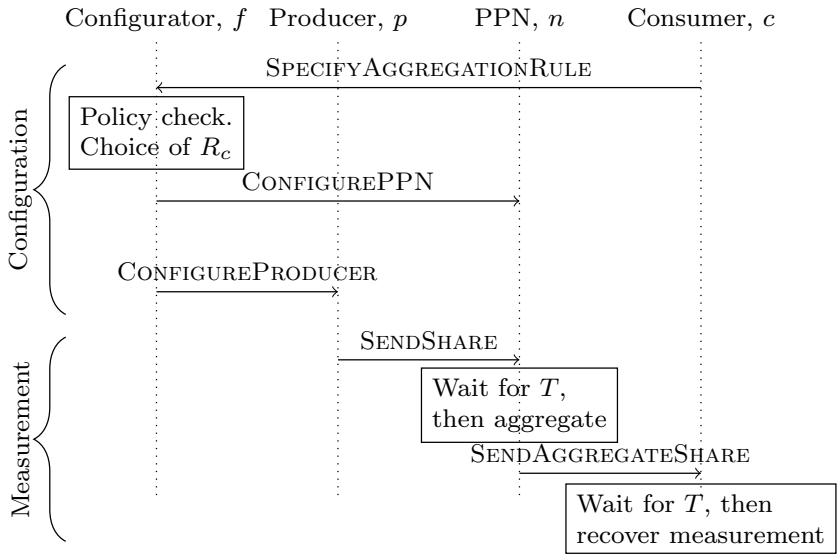

Figura 4.3: Scambio dei messaggi

ogni PPN le regole di aggregazione spaziali e temporali corrispondenti. In caso di rete di comunicazione perfettamente affidabile, w può essere assunta pari a t. Il Configurator invia anche un identificativo univoco random R_c. Questo numero è conosciuto solo dal Configurator e dai PPN.

3. ConfigureProducer

$$f \to p\colon \Omega_c$$

Per ogni Consumer c, il Configurator comunica a ogni Producer in Π_c l'insieme Ω_c di PPN a cui deve inviare una share delle proprie misure.

Una volta che la fase di setup è completata, ogni round vengono effettuate le seguenti operazioni. Si consideri l'i-simo round:

4. SendShare

$$p \to n\colon i \| \mu_i^p(n)$$

Ogni Producer c genera una misura μ_i^p. Se tale Producer è coinvolto in più regole di aggregazione, potrebbe dover inviare le sue share ad un numero di PPN maggiore di w. Indichiamo come w_p il numero di share necessarie che, in generale, può essere differente da nodo a nodo. Sfruttando lo schema di Shamir, il Producer divide le sue misure in w_p share e le invia ai w_p PPN.

Indichiamo con $\mu_i^p(n)$ lo share della misura μ_i^p inviata dal Producer p all'n-simo PPN al round i. Le share sono calcolate dal Producer, seguendo il protocollo SSS (Capitolo 3), usando il seguente polinomio random:

$$\mu_i^p(n) = \mu_i^p + \sum_{\nu=1}^{t-1} r_\nu n^\nu \bmod q \quad \forall n \in \Omega_c$$

Gli interi r_v sono un insieme di numeri interi casuali uniformemente distribuiti nell'intervallo $[0, q)$ e cambiati ad ogni round. Il numero primo q è un parametro di sistema più grande di ogni possibile misura aggregata.

5. SendAggregateShare

$$n \to c\colon AT \| i \| \sum_{j=1}^{M_c} v_j \| \sigma_i^c(n)$$

dove $M_c = |\Pi_c|$ e v_j è uguale a 1 se tutte le k_c share dal j-simo Producer in Π_c sono state ricevute dal PPN e 0 altrimenti. Per ogni regola di aggregazione comunicata dal Configurator, ogni PPN attende l'arrivo di share dai Producer per un tempo pari a T, quindi, in modo indipendente dagli altri PPN, effettua l'aggregazione dei dati ricevuti in accordo con la regola, calcolando la misura aggregata $\sigma_i^c(n)$ come:

$$\sigma_i^c(n) = \sum_{p \in \Pi_c} \sum_{a=i-k_c+1}^{i} \mu_a^p(n) \bmod q$$

Le misure sono inviate al Consumer c. In caso di errori di comunicazione, ritardi, o guasto di nodi, può succedere che alcune share non arrivino o non arrivino in tempo ai PPN. Se anche solo una singola share di un Producer è mancante, allora tutte le misure per quel Producer sono assunte pari a 0 per l'intera finestra di aggregazione. Poiché il Consumer può recuperare solamente misure aggregate che sono state calcolate sugli stessi input (ovvero sugli stessi Producer), il messaggio di SendAggregateShare include un *Aggregation Tag* (AT), calcolato come:

$$AT = h\left(R_c \| i \| \sum_{j=1}^{M_c} v_j 2^{(j-1)}\right)$$

dove h è una funzione di hash crittograficamente sicura. L'AT è uguale tra differenti PPN se i rispettivi input sono gli stessi, mentre è differente con alta probabilità se gli input sono differenti. Il messaggio include anche il numero di round e la cardinalità dell'insieme dei Producer. L'aggregazione è effettuata solamente ai round che sono multipli della finestra di aggregazione k_c.

Si assume che una share possa non essere ricevuta dai PPN per due differenti tipologie d'errore:

- Gli **errori di link** sono errori in trasmissione e accadono indipendentemente per ogni comunicazione tra Producer e PPN. Un messaggio viene considerato perso anche se arriva al PPN troppo tardi.
- Gli **errori di nodo** sono causati da problemi d'accesso lato Producer. Il Producer risulta irraggiungibile in rete, quindi nessun PPN è in grado di ricevere share da parte di quel Producer.

Una volta che il Consumer ha ricevuto, in un determinato tempo di round, tutte le share aggregate provenienti dai PPN, recupera le misure aggregate

considerando solamente l'insieme più grande di share con lo stesso AT. L'idea di includere l'identificatore R_c nell'AT rende difficile, per il Consumer, controllare se i PPN hanno effettuato l'aggregazione su uno specifico sottoinsieme di $\mathcal{P}$: il Consumer, quindi, non può conoscere quali Producer hanno sperimentato un errore di link o di nodo, ma può sapere solamente il loro numero. Includere il numero di round, invece, rende difficile per il Consumer capire se l'insieme di Producer su cui è stata effettuata l'aggregazione è cambiato da un round con l'altro. Una volta che il Consumer ha recuperato le misure aggregate correttamente, può ottenere una stima delle misure aggregate totali anche se i dati relativi a qualche Producer sono mancanti.

4.2 Architettura distribuita

L'architettura centralizzata descritta nella Sezione 4.1 mostra come è possibile far pervenire delle misure aggregate spazialmente e/o temporalmente ad un'entità che prende il nome di Consumer. A tal fine, l'architettura si avvale dell'utilizzo di nodi chiamati PPN (Privacy Preserving Node), i quali hanno il compito di garantire la privacy degli utenti per mezzo dell'aggregazione di share generate dagli utenti (Producer). Il termine *centralizzata* si riferisce al fatto che sono necessari dei nodi, i PPN, ideati esclusivamente per effettuare l'aggregazione delle share ed inoltrare la share aggregata al Consumer. Tali nodi sono ad elevata capacità computazionale poiché possono essere raggiunti da un numero elevato di Producer, in base alle regole di aggregazione stabilite dal Consumer. Oltre a ciò il Configurator, in base alle regole di aggregazione scelte dal Consumer, definisce durante la fase di setup quali PPN saranno responsabili dell'aggregazione e comunica la sua scelta ai Producer, i quali invieranno le proprie share a tali PPN nella successiva fase

di aggregazione. La scelta dei PPN responsabili dell'aggregazione è a completa discrezione del Configurator, il quale quindi definisce a priori i flussi di dati in rete.

L'architettura descritta in questa Sezione è un'evoluzione dell'architettura centralizzata. Essa prende il nome di architettura *distribuita*: tale termine si riferisce al fatto che, come verrà spiegato nel dettaglio, i nodi responsabili dell'aggregazione (Gateway) sono nodi con una capacità computazionale ridotta rispetto ai PPN, poiché solo una parte dei contatori (Meter) che sono monitorarti dall'entità che ha richiesto l'aggregazione (External Entity) è collegata ad essi. Anche in questa architettura il Configurator è responsabile della fase di setup, ma non ha alcun compito nella definizione dei flussi di dati in rete. Esso si limita ad approvare o meno le regole di aggregazione scelte dall'External Entity che, una volta ottenuta l'autorizzazione, si avvale della collaborazione di alcuni Gateway da lei scelti per la creazione di una topologia di rete che le consenta di ottenere le misure aggregate volute. Questo significa che i Gateway si auto-organizzano per effettuare correttamente l'aggregazione. Un'architettura di questo tipo è maggiormente scalabile e più aderente alla realtà, poiché i Gateway possono essere delle semplici apparecchiature poste nelle vicinanze dei Meter, ad esempio all'interno dei condomini.

L'architettura e il protocollo di comunicazione di riferimento sono trattati in [3]. In questo Capitolo vengono ripresi e dettagliatamente descritti.

Con riferimento alla figura 4.4, come già brevemente accennato, l'architettura prevede la presenza di tre differenti insiemi di nodi:

- L'insieme $\mathcal{M}$ dei *Meter*, ovvero i contatori che generano le misure relative al consumo e alla produzione della risorsa per ogni utente;
- L'insieme $\mathcal{G}$ dei *Gateway*, che effettuano l'aggregazione dei dati direttamente nel dominio crittografato. I Gateway ricevono misurazioni da

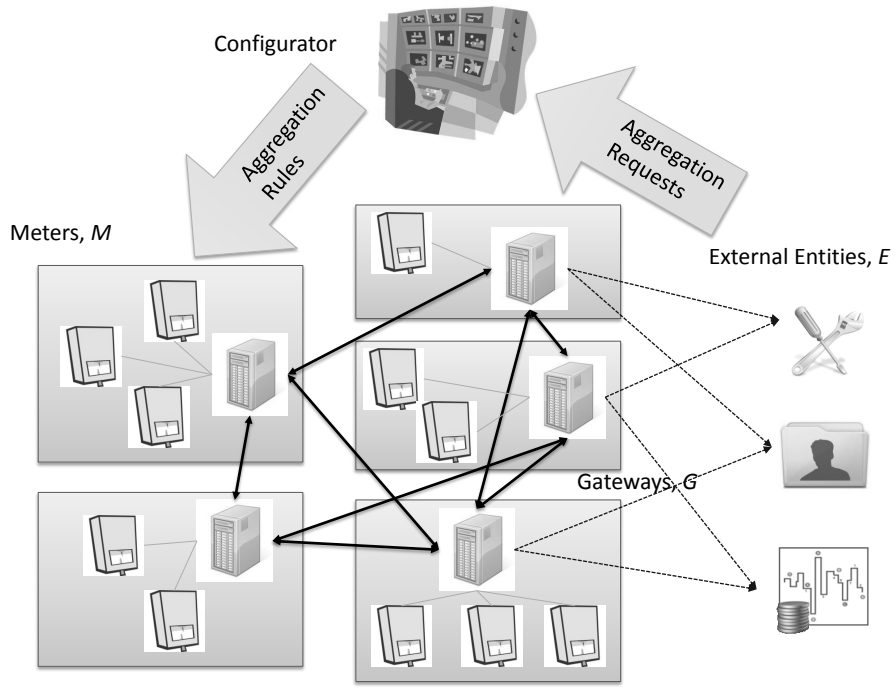

Figura 4.4: L'architettura distribuita

più Meter, ad esempio tutti i Meter di un condominio;

- L'insieme $\mathcal{E}$ delle *External Entity* che ricevono le informazioni aggregate e rappresentano le società di erogazione della risorsa o terze parti.

L'architettura include anche un *Configurator*, che riceve le richieste di aggregazione specificate dalle External Entity. Esso controlla se l'External Entity è autorizzata a monitorare i Meter richiesti e se la granularità dell'aggregazione richiesta è conforme con la politica di sistema. In caso affermativo, il Configurator autorizza l'aggregazione.

E' necessario specificare che i Gateway sono considerati come nodi funzionali e che l'architettura è agnostica rispetto alla loro collocazione fisica: Meter e Gateway possono sia essere implementati in un singolo dispositivo che essere collocati indipendentemente l'uno dall'altro. Assumiamo che ogni

Meter è associato a un solo Gateway, che ogni External Entity può comunicare con un sottinsieme di Gateway e che i Gateway sono interconnessi per mezzo di una rete di comunicazione pubblica. Assumiamo inoltre che i Meter abbiano identificatori da cui l'identificatore del Gateway a cui sono fisicamente collegati sia facilmente ricavabile, ad esempio per mezzo dell'utilizzo di identificatori gerarchici.

Le misurazioni effettuate dai Meter vengono suddivise dal Gateway a cui sono collegati in w share secondo lo schema SSS (Shamir Secret Sharing, Capitolo 3) e quindi inviate a differenti Gateway. Ogni share è identificato da un numero s tale che $1 \leq s \leq w$. Grazie alle proprietà di omomorfismo proprie dello schema SSS rispetto all'addizione, le share generate da differenti Meter e caratterizzate dallo stesso numero s possono essere aggregate indipendentemente dai Gateway, in accordo con le regole specificate dall'External Entity. Infine, le misure aggregate possono essere recuperate dall'External Entity in presenza di almeno $t \leq w$ share aggregate, con t paramentro definito dal sistema, per mezzo dell'algoritmo di Lagrange. E' possibile sfruttare per il recupero robusto l'algoritmo di Berlekamp-Welch (Capitolo 3), il quale permette di recuperare correttamente le misure aggregate anche in presenza di un numero $e < \frac{w-t+1}{2}$ di share aggregate errate, con w numero totale di share aggregate. In questo caso, a differenza dell'architettura centralizzata, non possiamo però ipotizzare la presenza in rete di Gateway malevoli che alterano le share aggregate, in quanto ogni Gateway può avere a che fare con ognuna delle w share, e un Gateway malevolo potrebbe alterare tutte le share aggregate rendendo impossibile il recupero delle misure aggregate. La ricezione da parte dell'External Entity di share aggregate errate, però, potrebbe essere causata da errori sistematici. Per aumentare la robustezza del protocollo è possibile considerare una soglia t molto inferiore a w.

Il processo di aggregazione funziona nel seguente modo: le misure dei Meter sono divise in share individuali. Queste share individuali e le share aggregate parzialmente sono sommate in accordo con le regole di aggregazione dell'External Entity approvate dal Configurator. Questi dati sono inviati agli altri Gateway o all'External Entity, nel caso in cui il processo di aggregazione sia completato.

La Figura 4.5 mostra la procedura di collezione e di aggregazione effettuata da ogni Gateway. Il Gateway riceve come input due differenti tipi di dati: (1) le misurazioni ottenute dai Meter ad esso collegati; (2) le share aggregate parzialmente e calcolate dagli altri Gateway. Poiché la capacità computazionale dei Gateway è limitata, il numero totale di share ricevute da ogni Gateway (*fan-in*) è limitato da una soglia.

Come nel caso di architettura centralizzata, illustrata nella Sezione 4.1, assumiamo nel modello di sicurezza un attaccante *onesto-ma-curioso*: tutti i Gateway e l'External Entity eseguono il protocollo correttamente, ma possono tener traccia di tutti i dati ricevuti al fine di recuperare informazioni addizionali sulle misure non aggregate generate dai singoli Meter.

Assumiamo anche che tutti i canali di comunicazione tra Meter, Gateway, External Entity e Configurator siano confidenziali ed autenticati usando il protocollo di sicurezza SSL/TLS.

4.2.1 Il protocollo di comunicazione

Ipotizziamo che il tempo sia diviso in intervalli (*round*) nell'ordine dei minuti: è richiesto che tutti i nodi siano sincronizzati ad uno stesso riferimento temporale. Ogni Meter, Gateway ed External Entity è associata ad un identificatore unico. Il protocollo di comunicazione si suddivide in due fasi: una fase iniziale di setup è avviata dall'External Entity e coinvolge l'Ex-

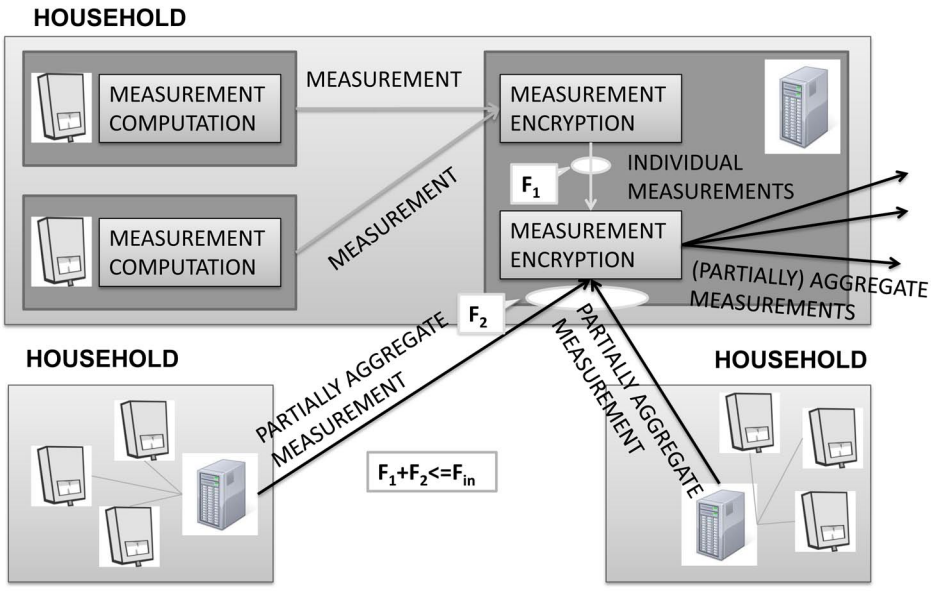

Figura 4.5: Collezione e aggregazione al Gateway

ternal Entity, il Configurator, i Gateway e i Meter. Terminata questa fase, ad ogni tempo di round viene gestita l'aggregazione spaziale e/o temporale delle misure: questa fase coinvolge i Meter, i Gateway e l'External Entity.

L'instradamento delle share aggregate tra i Gateway con destinazione l'External Entity è effettuato in modo completamente distribuito e definito nella fase di setup. I Gateway decidono l'instradamento dei flussi di comunicazione sfruttando una variante del *protocollo di instradamento Chord*, descritto brevemente nella Sezione 3.2. Un approccio completamente distribuito piuttosto che centralizzato, nel quale invece il Configurator a priori definisce i flussi tra i Gateway, è migliore in quanto non è necessario che il Configurator e i Gateway abbiano una conoscenza completa della topologia della rete. L'algoritmo di instradamento è basato completamente su Chord e richiede che tutti i Gateway condividano una famiglia di funzioni di hash indipendenti h_s di numero pari al numero di share w in cui è suddivisa ogni singola misura. Ogni Gateway effettua l'hash del suo ID w volte usando le

funzioni di hash $h_1 \dots h_w$. In questo modo si ottenegono w anelli indipendenti in accordo con le regole standard di Chord. Ogni anello Chord così creato è responsabile dell'instradamento, ad ogni round, di una delle share parzialmente aggregate appartenente all'insieme S: ad esempio, il primo anello è responsabile per la share avente $s = 1$ e così via. Per ogni anello s, l'External Entity invia ad un Gateway scelto in modo casuale il numero s. Quel Gateway prende il nome di *Gateway di frontiera* ed è responsabile per l'aggregazione di quella specifica share aggregata: esso ha il compito di inviare all'External Entity la share una volta terminato il suo processo di aggregazione. E' importante che l'External Entity scelga un Gateway di frontiera diverso per ogni anello in modo da evitare che un Gateway possa recuperare le misure aggregate. La definizione dei percorsi delle share aggregate in rete viene effettuata partizionando opportunamente gli anelli Chord in base agli ID Chord mediante l'invio di messaggi di SetupTree, che verranno descritti dettagliatamente più avanti. Questi messaggi permettono la creazione di un albero di instradamento per ogni anello s avente come radice il rispettivo Gateway di frontiera. Una volta creato l'albero, l'inoltro delle share parzialmente aggregate durante la fase di aggregazione segue il percorso inverso.

Identificati il Configurator, i Gateway, i Meter e l'External Entity rispettivamente con le lettere f, i e j, m ed e rispettivamente, descriviamo i messaggi scambiati nella *fase di setup* (Figura 4.6):

1. SpecifyAggregationRule

$$e \to f \colon \mathcal{M}_e \| k_e$$

L'External Entity $e \in \mathcal{E}$ comunica al Configurator f una regola di aggregazione $(\mathcal{M}_e, k_e)$, dove $\mathcal{M}_e$ indica l'insieme dei Meter che l'External

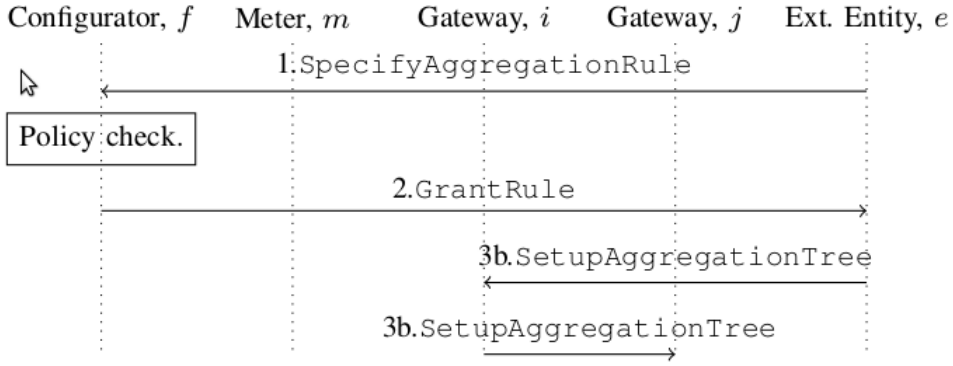

Figura 4.6: Fase di setup

Entity vuole monitorare, e k_e esprime la finestra di aggregazione, ovvero il numero di intervalli di tempo su cui i dati devono essere aggregati. D'ora in poi assumeremo che ogni External Entity specifichi una sola regola di aggregazione.

2. GrantRule

$$f \to e\colon \text{Grant}_f$$

Il Configurator controlla la conformità delle regole di aggregazione specificate da ogni External Entity alle politiche di sicurezza proprie del sistema. Se la richiesta è accettata, il Configurator invia all'External Entity un *grant* definito come $\text{Grant}_f = \mathcal{M}_e \| k_e \| T_{\text{exp}} \| \text{sig}_f(\mathcal{M}_e \| k_e \| T_{\text{exp}})$, dove T_{exp} è il tempo di scadenza del grant. Il grant contiene la firma di $\mathcal{M}_e \| k_e \| T_{\text{exp}}$ per mezzo della funzione di firma sig_f che utilizza la chiave privata del Configurator. In questo modo i parametri $\mathcal{M}_e$, k_e, T_{exp} non possono essere in alcun modo modificati da alcun nodo della rete.

3. SetupTree

$$e \to i \text{ (or } i \to j)\colon s \| \text{Grant}_f \| (ID_{\min}, ID_{\max})$$

L'External Entity e contatta un numero i di Gateway di frontiera pari al numero w delle share. Ognuno di questi Gateway riceve un differente numero di share s indicante che quel Gateway sarà responsabile dell'aggregazione dello share s. Consideriamo allora fisso il numero s e ci soffermiamo sulla descrizione del messaggio di SetupTree per un singolo anello.

Insieme al grant, ogni Gateway riceve una coppia di ID Chord, $ID_{\min}$ e $ID_{\max}$, che indicano un intervallo di ID Chord che il Gateway è delegato ad aggregare: $ID_{\min}$ coincide sempre con il proprio Chord ID. Nel caso di Gateway di frontiera, l'intervallo di ID Chord che è delegato ad aggregare coincide con l'intero anello Chord. Ogni Gateway in primo luogo controlla che il grant sia corretto verificando la firma del Configurator, quindi identifica quali Meter in $\mathcal{M}_e$ sono ad esso localmente connessi. Successivamente partiziona l'intervallo $(ID_{\min}, ID_{\max})$ in base alla propria finger table seguendo le regole standard di Chord, ovvero andando a identificare i successori per le differenti finger (vedere Sezione 3.2). Tali successori sono i Gateway $j \in \mathcal{G}$ responsabili per i Meter in $\mathcal{M}_e$ rimanenti, e ad essi viene viene inoltrato un messaggio di SetupTree in cui sono presenti (1) il grant e (2) la nuova coppia di ID che identifica l'intervallo per cui j è responsabile.

Terminata la fase di invio dei messaggi di SetupTree, è creato un albero con radice nel Gateway di frontiera. I Gateway quindi sono pronti per aggregare le share dai Meter locali con le share parzialmente aggregate provenienti dagli altri Gateway (nodi figli) seguendo il percorso inverso rispetto al percorso seguito in rete dai SetupTree.

Una volta definito l'albero di aggregazione nella fase di setup per ogni share

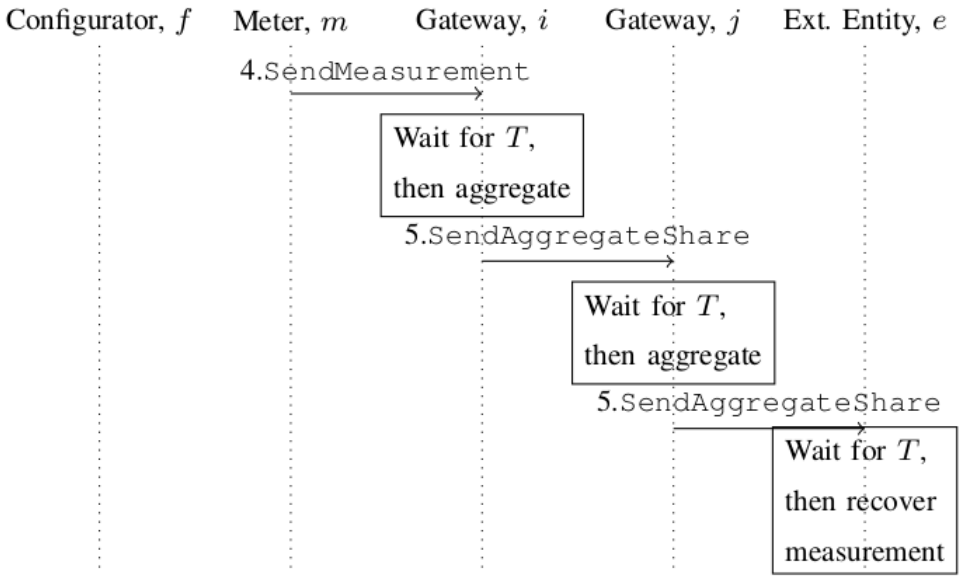

Figura 4.7: Fase di aggregazione

s e, quindi, fissati i flussi di comunicazione tra i diversi Gateway e tra i Gateway di frontiera e l'External Entity, per ogni tempo di round τ vengono scambiati i seguenti messaggi (*fase di aggregazione*, vedere Figura 4.7):

4. SendMeasurement

$$m \to i \colon \tau \| \phi_m(\tau)$$

La misura $\phi_m(\tau)$ generata dal Meter m al tempo di round τ viene inviata al Gateway i a cui il Meter è connesso. Tale misura viene divisa dal Gateway in w share.

5. SendAggregateShare

$$i \to j \text{ (or } i \to e) \colon \tau \| s \| ID_e \| \sigma_e^i(\tau, s)$$

Per ogni regola di aggregazione specificata dal Configurator, ogni Gateway attende l'arrivo di share per un tempo T quindi, indipendentemente dagli altri Gateway, effettura l'aggregazione direttamente nel

dominio crittografato in accordo con la regola di aggregazione specificata, calcolando le share parzialmente aggregate come $\sigma_e^i(\tau, s) = \sum_{k \in \Omega_i} \sigma_e^k(\tau, s)$, dove l'insieme Ω_i include i Gateway che inviano una share parzialmente aggregata ad i ed i Meter locali coinvolti nell'aggregazione. L'aggregazione viene effettuata dopo un numero di tempi di round multipli della finestra di aggregazione k_e.

La share parzialmente aggregata è quindi inviata al successivo Gateway $j \in \mathcal{G}$ lungo il percorso associato al numero di share corrispondente s oppure, nel caso di Gateway di frontiera, essendo la procedura di aggregazione conclusa, la share aggregata viene inviata all'External Entity e, la quale attende la ricezione delle w share aggregate ed infine recupera le misure aggregate. Nel messaggio sono inclusi anche il tempo di round τ e l'identificativo univoco ID_e dell'External Entity e.

CAPITOLO 5

GLI EMULATORI

Dopo aver introdotto le infrastrutture di rete e i protocolli di comunicazione, come descritti nel Capitolo 4, ci soffermiamo sulla presentazione degli emulatori che abbiamo realizzato al fine di testarne le prestazioni.

Sono stati realizzati due differenti emulatori, uno che si riferisce all'architettura centralizzata (Sezione 4.1) e uno che si riferisce invece all'architettura distribuita (Sezione 4.2). Entrambi gli emulatori sono stati scritti in Python [19], uno tra i più diffusi linguaggi di scripting: essi sfruttano le potenzialità del framework Twisted [20]. Nelle sezioni successive verranno descritti l'ambiente di lavoro e in che modo abbiamo scritto il codice sorgente, con particolare rilievo sull'implementazione dei messaggi previsti dai protocolli e dai messaggi scambiati in rete. Infine verranno illustrati i risultati sperimentali, soffermandoci soprattutto sull'architettura distribuita.

L'emulatore relativo all'architettura centralizzata, che abbiamo implementato per primo e di più semplice realizzazione rispetto all'emulatore re-

lativo all'architettura distribuita, ci ha permesso di prendere confidenza con l'ambiente di lavoro.

5.1 Twisted

Twisted [20] è un framework event-driven basato su Python e studiato per lo sviluppo di applicazioni Internet; è rilasciato sotto licenza open source dal MIT e permette in modo semplice di implementare protocolli di rete. Le motivazioni che ci hanno spinto a sceglierlo come ambiente di lavoro sono essenzialmente due. In primo luogo, le librerie di Twisted realizzate in Python permettono di creare applicazioni client e server senza doversi preoccupare dell'instaurazione delle connessioni TCP o SSL tra le parti coinvolte: una volta definiti indirizzo IP, porta ed eventuali informazioni aggiuntive (ad esempio il proprio certificato X.509 e il certificato X.509 della Certification Authority nel caso di connessione SSL/TLS), la gestione della connessione è esclusivamente a carico di Twisted. In secondo luogo, Twisted permette di sfruttare come paradigma di programmazione la programmazione ad eventi. Come noto, nella programmazione ad eventi il flusso di esecuzione del programma è determinato da eventi come, ad esempio, la ricezione di un messaggio: differenti eventi provocano differenti risposte da parte del programma. Ciò ha permesso di organizzare il codice sorgente dei nostri script in modo semplice. La gestione degli eventi è sicuramente l'aspetto più importante di Twisted, poiché ha portato ad una notevole semplificazione del codice sorgente.

5.2 Architettura centralizzata

Come descritto nella Sezione 4.1, l'architettura centralizzata prevede l'esistenza di quattro differenti tipologie di nodi: i Producer, i Consumer, i PPN e il Configurator. Ognuno di questi nodi necessita la scrittura di uno script differente.

Abbiamo iniziato la stesura del codice sorgente partendo dal Producer: questo script gestisce la generazione delle misure e la divisione di ogni singola misura in un numero predefinito di share, oltre che il corretto invio di ognuna di queste share ai differenti PPN, supponendo di conoscere indirizzo IP e porta per ognuno. L'emulatore prevede la possibilità di considerare due eventi inaspettati: il malfunzionamento di un nodo oppure il malfunzionamento di un link. Nel primo caso il Producer, ad ogni tempo di round, risulta irraggiungibile con una certa probabilità: questo significa che nessuno share, in quel tempo di round, viene inviato ad alcun PPN. Nel secondo caso, invece, ad ogni tempo di round fallisce con una certa probabilità la comunicazione tra Producer e PPN: in questo caso la misura viene effettuata, viene suddivisa in share, ma una o più share possono non essere ricevute dai PPN.

Successivamente ci siamo dedicati alla stesura del codice sorgente relativo al PPN. Il PPN rimane in ascolto in attesa della ricezione di messaggi da parte dei Producer ed effettua la somma aggregata spazialmente e temporalmente delle share ricevute dopo ogni finestra di aggregazione, secondo le regole descritte nella Sezione 4.1. Il PPN si preoccupa anche della generazione dell'Aggregation Tag. Per l'Aggregation Tag abbiamo utilizzato la funzione di hash SHA-224. Twisted ci ha permesso di pianificare l'evento di aggregazione delle share: queste, una volta ricevute dai PPN, vengono salvate in una matrice opportunamente dimensionata ($|\Pi_c| \times K_c$) ed aggregate dopo ogni finestra di aggregazione. La share aggregata, una volta creata, viene

inviata al Consumer, supponendo noti il suo indirizzo IP e la sua porta.

Il Consumer, quindi, rimane in ascolto in attesa delle share aggregate da parte dei PPN. Una volta ricevute tutte le share aggregate, la prima operazione che compie è controllare il loro Aggregation Tag. Una volta effettuata questo controllo, procede con il tentativo di recupero delle misure aggregate. Abbiamo implementato nel codice sia il recupero standard per mezzo dell'interpolazione di Lagrange che il recupero robusto per mezzo dell'algoritmo di Berlekamp-Welch (Capitolo 3). Per verificare il corretto funzionamento dell'algoritmo abbiamo modificato il codice sorgente di alcuni PPN rendendoli malevoli: questi PPN non trasmettono al Consumer la share aggregata corretta ma una share aggregata differente e senza alcun significato. Abbiamo verificato che, al di sotto della soglia di PPN malevoli imposta dalla teoria, le misure aggregate possono essere comunque recuperate. L'implementazione dell'algoritmo di Berlekamp-Welch ci ha spinto ad includere nel codice le librerie di SAGE [21], un software di matematica basato su Python che permette di effettuare calcoli anche complessi per mezzo di una sintassi molto semplice. Questo è stato necessario perché l'algoritmo di Berlekamp-Welch prevede operazioni non banali, come ad esempio il rapporto tra due polinomi con coefficienti modulari.

Per ultimo, abbiamo completato l'architettura introducendo il Configurator: esso rimane in ascolto in attesa della ricezione di una regola di aggregazione da parte del Consumer, quindi inoltra ai PPN e ai Producer coinvolti nell'aggregazione la regola specificata. Questa fase avviene all'avvio dell'emulatore. Ovviamente è stato modificato coerentemente anche il codice sorgente dei PPN e dei Producer per permettere la ricezione dei messaggi da parte del Configurator.

5.2.1 Composizione dei messaggi

I messaggi definiti dal protocollo presentano una struttura standard: la prima riga prende il nome di *Intestazione*. Essa è seguita da una serie di righe formattate secondo la dicitura:

```
Name: <value>
```

Queste righe trasportano le informazioni necessarie per le operazioni compiute dai vari nodi coinvolti nell'architettura di rete. Ci sono alcune righe che sono presenti in tutti i messaggi, mentre altre sono specifiche per ogni differente tipo di messaggio. La scelta di uno standard testuale secondo la formattazione `Name: <value>` piuttosto che un approccio bit-oriented è stata presa a causa della maggiore semplicità di gestione e comprensione di messaggi di questo tipo: abbiamo fatto riferimento ad HTTP, specialmente alla parte relativa ai messaggi di risposta.

L'*intestazione* trasporta l'informazione relativa al nome, alla versione del protocollo e al tipo di messaggio, che viene specificato sia da un codice numerico di due cifre che dal suo nome esteso. Questo è un esempio di intestazione:

```
AP/1.0 01 SpecifyAggregationRule
```

In questo esempio si nota che si ha a che fare con la versione 1.0 del protocollo AP (Aggregation Protocol) e il messaggio trasportato è un messaggio di *SpecifyAggregationRule*, univocamente definito dal codice di due cifre 01. Per completezza riportiamo la corrispondenza tra codice numerico e nome del messaggio per tutti i messaggi:

```
01 SpecifyAggregationRule
02 ConfigurePpn
```

```
03 ConfigureProducer
04 SendShare
05 SendAggregateShare
```

L'ordine numerico è stato scelto in base all'ordine in cui i messaggi vengono elaborati e inviati in rete.

Inoltre, due righe sono sempre presenti indipendentemente dal tipo di messaggio scambiato. Queste due righe sono:

```
From: <value>
Date: <value>
```

La riga `From: <value>` indica al destinatario chi ha generato il messaggio. In questo caso `<value>` è l'identificativo numerico univoco del mittente all'interno dell'architettura. La riga `Date: <value>` indica il timestamp del messaggio. Esso segue il formato descritto nell'RFC 822 [22]. Di seguito riportiamo un esempio per ciascuna di queste due righe:

```
From: 34
Date: Wed, 11 Jul 2012 14:27:22 GMT
```

Tutte le altre righe sono specifiche per il tipo di messaggio al quale appartengono. Per questo motivo di seguito, piuttosto che una descrizione riga per riga, viene riportato un esempio per ogni possibile tipologia di messaggio e vengono descritti nel caso specifico i differenti campi.

5.2.2 Descrizione dei messaggi

Di seguito riportiamo un esempio per ogni differente tipo di messaggio descrivendo le righe da cui è composto.

Il messaggio *SpecifyAggregationRule* è così fatto:

```
AP/1.0 01 SpecifyAggregationRule
From: 12
Date: Wed, 11 Jul 2012 14:26:42 GMT
Pi_c: 3,5,7
K_c: 3
```

La riga `From: <value>` definisce l'identificativo del Consumer che invia la richiesta per ottenere i dati aggregati.
La riga `Date: <value>` trasporta il timestamp del messaggio.
La riga `Pi_c: <value>` definisce l'insieme dei Producer che il Consumer vuole monitorare; gli identificativi devono essere scritti separati da una virgola senza spazi.
La riga `K_c: <value>` indica la finestra di aggregazione.

Il messaggio *ConfigurePpn* è così fatto:

```
AP/1.0 02 ConfigurePpn
From: 1
Date: Wed, 11 Jul 2012 14:26:42 GMT
Pi_c: 3,5,7
K_c: 3
R_c: 746
```

La riga `From: <value>` definisce l'identificativo del Configurator.
La riga `R_c: <value>` trasporta l'identificatore random, scelto dal Configurator, associato al Consumer che richiede l'aggregazione.

Il messaggio *ConfigureProducer* è così fatto:

```
AP/1.0 03 ConfigureProducer
```

```
From: 1
Date: Wed, 11 Jul 2012 14:26:42 GMT
O_c: 20,21,23
```

La riga `From: <value>` definisce l'identificativo del Configurator.
La riga `O_c: <value>` definisce l'insieme dei PPN ai quali il Producer deve inviare una share della misura; gli identificativi devono essere scritti separati da una virgola senza spazi.

Il messaggio *SendShare* è così fatto:

```
AP/1.0 04 SendShare
From: 3
Date: Wed, 11 Jul 2012 14:27:22 GMT
Round: 4
ShareLenght: 7
Share: 2160529
```

La riga `From: <value>` definisce l'identificativo del Producer che ha generato la share.
La riga `Round: <value>` indica il numero di round in cui è stata effettuata la misura e conseguentemente calcolato la share. Si ipotizza che la rete sia sincronizzata.
La riga `ShareLenght: <value>` indica la lunghezza (in cifre) della share.
La riga `Share: <value>` trasporta il valore numerico della share calcolato dal Producer.

Il messaggio *SendAggregateShare* è così fatto:

```
AP/1.0 05 SendAggregateShare
```

```
From: 20
Date: Wed, 11 Jul 2012 14:27:17 GMT
Round: 3
AT: 48fae86781b8aa89b8254740d7d17d3d153e5b7c7f4b2473f928ead3
NumberProd: 3
AggrShareLenght: 6
AggrShare: 735289
```

La riga `From: <value>` definisce l'identificativo del PPN che ha calcolato la share aggregata.
La riga `Round: <value>` indica il numero di round in cui è stata calcolata la share aggregata. Questo valore sarà sempre multiplo di K_c.
La riga `AT: <value>` trasporta l'Aggregation Tag (AT).
La riga `NumberProd: <value>` definisce la cardinalità dell'insieme dei Producer che competono all'aggregazione, ossia il numero di Producer che ha inviato correttamente tutte le share nell'arco della finestra di aggregazione.
La riga `AggrShareLenght: <value>` indica la lunghezza (in cifre) della share aggregata.
La riga `AggrShare: <value>` trasporta il valore numerico della share aggregata calcolata dal PPN.

5.3 Architettura distribuita

Come descritto nella Sezione 4.2, l'architettura distribuita non prevede la presenza di nodi specifici per l'aggregazione ma si avvale dell'utilizzo di Gateway posti nelle abitazioni degli utenti o nelle loro vicinanze. Questa

architettura prevede quattro differenti tipologie di nodi: i Meter, i Gateway, l'External Entity e il Configurator.

In questo caso, differentemente dall'architettura centralizzata, abbiamo iniziato la stesura del codice sorgente partendo dal Configurator e dall'External Entity, implementando lo scambio mutuo di messaggi proprio della fase di configurazione dei dispositivi in rete: il Configurator rimane in ascolto in attesa della ricezione di un messaggio da parte dell'External Entity che definisca una regola di aggregazione, quindi risponde all'External Entity con il *grant*.

Successivamente abbiamo scritto il codice dell'External Entity e del Gateway relativamente ai messaggi *SetupTree*, necessari per un corretto instradamento delle share dai Gateway verso l'External Entity. Poiché l'instradamento distribuito fonda le sue basi sull'instradamento di Chord [10] e ad ogni Gateway è associato un identificatore Chord per ogni anello Chord, abbiamo ideato un semplice algoritmo deterministico per definire la corrispondenza tra ID del nodo e Chord ID per ogni anello. L'algoritmo utilizza la funzione di hash MD5, la quale prende come input una stringa nella quale sono concatenati un parametro non ambiguo per ogni Gateway (ad esempio l'ID del Gateway, oppure il suo indirizzo IP o la porta TCP sulla quale il Gateway si mette in ascolto nel caso di emulazione in locale) e il numero dell'anello Chord. La stringa ottenuta in output viene convertita in intero, del quale viene poi calcolato il valore modulo 2^k, dove k è il numero di bit del Chord ID. In questo modo, quindi, per ogni Gateway possono essere calcolati in modo semplice gli identificatori Chord.

Per quanto riguarda l'instradamento Chord (Sezione 3.2), abbiamo sfruttato un'implementazione di Chord open source scritta in Python [23] che permette di ottenere in locale la finger table per ogni nodo appartentente

all'anello Chord conosciuti i Chord ID dei nodi. Per sfruttare questo codice abbiamo ipotizzato uno scenario di rete statico, in cui le finger table non si modificano nel tempo: questo significa che non sono contemplate join e leave di Gateway. Ad ogni nodo, quindi, viene fornita la propria finger table per ogni anello Chord: l'emulatore, in primo luogo, crea la topologia Chord sottostante, ovvero i differenti anelli Chord per le differenti share, quindi avvia lo scambio di messaggi. La parte più impegnativa è stata scrivere il codice relativo all'algoritmo di inoltro dei messaggi di *SetupTree* in rete per la creazione degli alberi di instradamento delle share, sfruttando il partizionamento degli anelli Chord. Al termine della fase di scambio dei messaggi di *SetupTree* ogni Gateway ha memorizzata una matrice di puntatori che indica al Gateway il Chord ID del suo nodo padre per ogni anello Chord (può essere un altro Gateway o l'External Entity nel caso in cui il Gateway, per quel determinato anello Chord, sia di frontiera).

Ci siamo quindi dedicati alla stesura della parte di codice per il Gateway necessaria per l'aggregazione e l'invio delle share ai nodi padre in modo analogo a quanto fatto per l'emulatore nel contesto di un'architettura centralizzata.

Per completare l'architettura, abbiamo introdotto il codice relativo al Meter. Esso è molto semplice poiché il Meter invia solamente le misure al Gateway a cui è collegato. Ci siamo soffermati su due differenti scenari: il primo considera il Meter come un'entità separata dal Gateway e prevede l'invio in rete dei messaggi *SendMeasurement*, mentre il secondo prevede che i meter siano un'entità fisicamente co-locata con il Gateway e il codice sorgente del Meter è incorporato in quello del Gateway. Per mezzo di quest'ultima soluzione l'emulatore è reso più snello e sia l'onere computazionale che l'occupazione di memoria sono notevolmente ridotti.

Per ultimo abbiamo introdotto il recupero della misure aggregate nel codice dell'External Entity per mezzo dell'algoritmo di Berlekamp-Welch (Capitolo 3), sfruttando ancora una volta le potenzialità di SAGE [21].

5.3.1 Composizione dei messaggi

I messaggi definiti dal protocollo seguono la stessa formattazione e le stesse regole descritte nella sezione precedente, riguardante l'architettura centralizzata. L'*intestazione* è leggermente diversa:

```
DAP/1.0 01 SpecifyAggregationRule
```

In questo caso si ha a che fare con la versione 1.0 del protocollo DAP (Distributed Aggregation Protocol) e il messaggio trasportato è un messaggio di SpecifyAggregationRule, univocamente definito dal codice di due cifre 01. Per completezza riportiamo la corrispondenza tra codice numerico e nome del messaggio per tutti i messaggi del protocollo:

```
01 SpecifyAggregationRule
02 GrantRule
03 SetupTree
04 SendMeasurement
05 SendAggregateShare
```

5.3.2 Descrizione dei messaggi

Di seguito riportiamo un esempio per ogni differente tipo di messaggio descrivendo le righe da cui è composto.

Il messaggio *SpecifyAggregationRule* è così fatto:

```
DAP/1.0 01 SpecifyAggregationRule
From: 1073741824
Date: Sat, 22 Sep 2012 13:05:23 GMT
M_e: 70001,80002,90003
K_e: 3
```

La riga `From: <value>` definisce l'identificativo dell'External Entity che invia la richiesta per ottenere i dati aggregati da parte dei Meter.
La riga `Date: <value>` trasporta il timestamp del messaggio.
La riga `M_e: <value>` definisce l'insieme dei Meter che l'External Entity vuole monitorare; gli identificativi devono essere scritti separati da una virgola senza spazi. Si ricorda che l'architettura prevede che gli indirizzi dei Gateway a cui sono collegati i Meter da monitorare siano univocamente ricavabili dall'indirizzo stesso dei Meter: questo permette all'External Entity di conoscere a priori quali saranno i Gateway che avranno misure aggregate da inviare.
La riga `K_e: <value>` indica la finestra di aggregazione.

Il messaggio *GrantRule* è così fatto:

```
DAP/1.0 02 GrantRule
From: 1
Date: Sat, 22 Sep 2012 13:05:23 GMT
M_e: 1,3,5
K_e: 3
T_exp: 1348405517
Sign: 48fae86781b8aa89b8254740d7d17d3d153e5b7c7f4b2473f928ead3
```

La riga `From: <value>` definisce l'identificativo del Configurator.
La riga `T_exp: <value>` trasporta l'Expiration Time relativamente all'Ex-

ternal Entity, codificato secondo l'Epoch Time e scelto dal Configurator. Esso limita temporalmente la possibilità di ottenere dati aggegati.
La riga `Sign: <value>` è la firma dei valori concatenati M_e, K_e e T_{exp} per mezzo di una funzione di firma da parte del Configurator. Ricordiamo che i campi M_e, K_e e T_{exp} e *Sign* concatenati prendono il nome di *grant*.

Il messaggio *SetupTree* è così fatto:

```
DAP/1.0 03 SetupTree
From: 129752983
Date: Sat, 22 Sep 2012 13:05:23 GMT
S: 3
M_e: 1,3,5
K_e: 3
T_exp: 1348405517
Sign: 48fae86781b8aa89b8254740d7d17d3d153e5b7c7f4b2473f928ead3
ID_min: 524768956
ID_max: 709657017
```

La riga `From: <value>` definisce l'identificativo del Gateway o dell'External Entity.
La riga `S: <value>` definisce il numero di share a cui il messaggio si riferisce e quindi a quale anello Chord il messaggio appartiene.
La riga `ID_min: <value>` indica l'ID minimo relativo agli ID dei Gateway di cui chi riceve il messaggio è responsabile: coincide con il proprio ID.
La riga `ID_max: <value>` indica l'ID massimo relativo agli ID dei Gateway di cui chi riceve il messaggio è responsabile.
ID_{min} e ID_{max} sono gli estremi dell'intervallo degli ID di cui il Gateway che riceve il messaggio è responsabile.

Il messaggio *SendMeasurement* è così fatto:

```
DAP/1.0 04 SendMeasurement
From: 1
Date: Sat, 22 Sep 2012 13:05:36 GMT
Round: 1
Measure: 457895
```

La riga `From: <value>` definisce l'identificativo del Meter che ha effettuato la misura.
La riga `Round: <value>` indica il numero di round in cui è stata effettuata la misura. Si ipotizza che la rete sia sincronizzata.
La riga `Measure: <value>` trasporta il valore numerico della misura effettuata dal Meter.

Il messaggio *SendAggregateShare* è così fatto:

```
DAP/1.0 05 SendAggregateShare
From: 581291653
Date: Sat, 22 Sep 2012 13:07:30 GMT
T_round: 6
S: 1
ID_e: 1073741824
Sigma: 68413508
```

La riga `From: <value>` definisce l'identificativo del Gateway che ha inviato la share aggregata.
La riga `T_round: <value>` indica il numero di round in cui è stata calcolata la share aggregata. Questo valore sarà sempre multiplo di k_e.

La riga `IDe: <value>` indica l'identificativo dell'External Entity per conto della quale si sta effettuando l'aggregazione.
La riga `Share: <value>` trasporta il valore numerico della share aggregata calcolata dal Gateway.

5.4 Struttura degli emulatori

Ogni nodo della rete, con tutte le sue funzionalità, è emulato da uno script Python con estensione .py (oppure .sage, nel caso si sia reso necessario includere le librerie SAGE, ad esempio per il recupero delle misure aggregate da parte del Consumer o dell'External Entity). Quando viene avviata l'emulazione, ogni nodo con il proprio ID, il proprio indirizzo IP e la porta in ascolto TCP, genera un differente processo Python. L'emulatore è stato strutturato in modo tale da poter funzionare sia in caso di connessioni locali che in caso di connessioni remote. Nel primo caso tutti i nodi sono simulati sulla stessa macchina e tutti i messaggi sono scambiati sull'interfaccia *localhost*: in questo modo è stato semplice testare il funzionamento corretto delle funzionalità dei nodi durante la fase di sviluppo del software. E' importante specificare che in un contesto simile, a causa della condivisione dell'interfaccia per tutti i messaggi, ogni nodo deve porsi in ascolto su una porta TCP differente. Nel secondo caso, invece, i nodi possono essere simulati su differenti computer ed è necessario impostare correttamente gli indirizzi IP delle macchine. Per eseguire le misure prestazionali del protocollo abbiamo deciso di seguire un approccio ibrido tra i due descritti sopra: la maggior parte dei nodi sono simulati su una macchina, mentre gli altri sono simulati su un altro computer (ad esempio un Gateway e l'External Entity) in modo tale da poter testare le

Parametro	Valore	Descrizione
T_{round}	20 s	Durata di ogni singolo round
q	15000017	Numero primo del sistema
k	30 bit	Lunghezza degli ID chord
K_e	$3 \cdot T_{round}$	Finestra di aggregazione
w	4	Numero di share

Tabella 5.1: Parametri di emulazione

prestazioni di rete. E' importante notare che tutte le connessioni sono autenticate e cifrate per mezzo del protocollo SSL/TLS: per far ciò abbiamo creato una Certification Authority (CA) e un certificato X.509 per ogni nodo con il software open source OpenSSL [24], quindi abbiamo fornito ad ogni nodo la propria chiave privata, il proprio certificato X.509 e il certificato X.509 della CA.

5.5 Risultati sperimentali

In questa sezione vengono riportate le misure sperimentali ottenute dagli emulatori. Ci siamo concentrati principalmente sull'emulatore relativo all'architettura distribuita, poiché tale architettura, come più volte sottolineato, ha un interesse molto maggiore rispetto all'architettura centralizzata e poiché l'architettura centralizzata offre misure per lo più deterministiche: ad esempio, fissato il numero di PPN, il *fan-in* (numero messaggi in ingresso) di Gateway e Consumer e il *fan-out* (numero di messaggi in uscita) di Producer e PPN sono calcolabili analiticamente.

Nella Tabella 5.1 sono illustrati i parametri di emulazione. L'emulatore

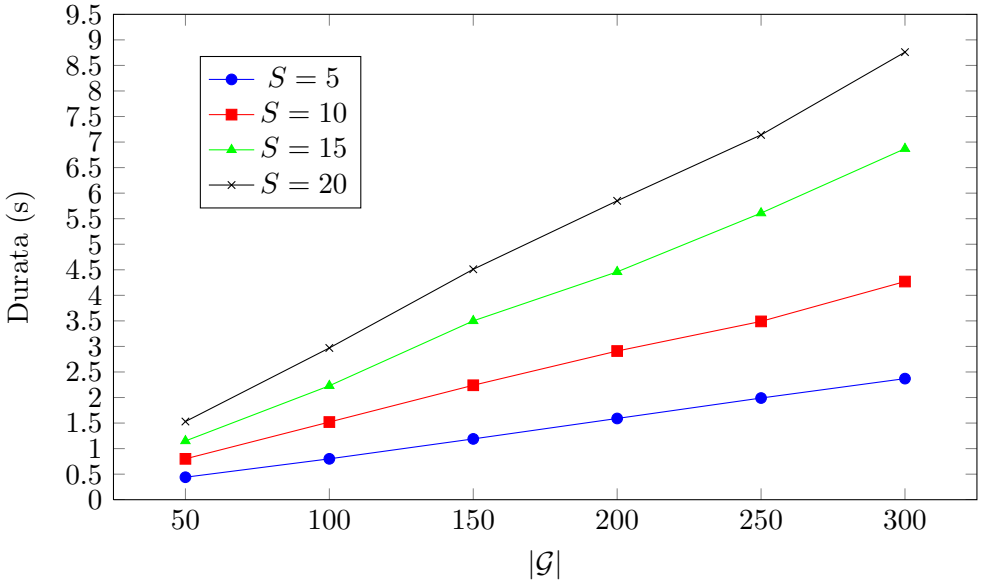

Figura 5.1: Durata della fase di generazione dell'albero al variare del numero di Gateway e fissato il numero di anelli Chord

è stato avviato, in base alle differenti misurazioni effettuate, su una o due macchine del laboratorio Bonsai del Politecnico di Milano. Le macchine sono entrambe dotate di 8 Gb di memoria RAM e di un processore Intel quad-core con Core i5-2400 a frequenza di clock pari a 3.10 GHz.

5.5.1 Durata della fase di SetupTree

Per ottenere le misure di seguito riportate abbiamo avviato l'emulatore su due macchine differenti: sulla prima macchina abbiamo collocato l'External Entity e un Gateway foglia per un anello, mentre sulla seconda macchina abbiamo collocato il resto della rete, ovvero tutti gli altri Gateway e il Configurator. Abbiamo quindi valutato l'intervallo temporale tra il primo messaggio di *SetupTree* inviato dall'External Entity e l'ultimo messaggio di *SetupTree* ricevuto dal Gateway foglia. In questo modo è possibile valutare la durata complessiva della fase di generazione dell'albero per mezzo dei messaggi di

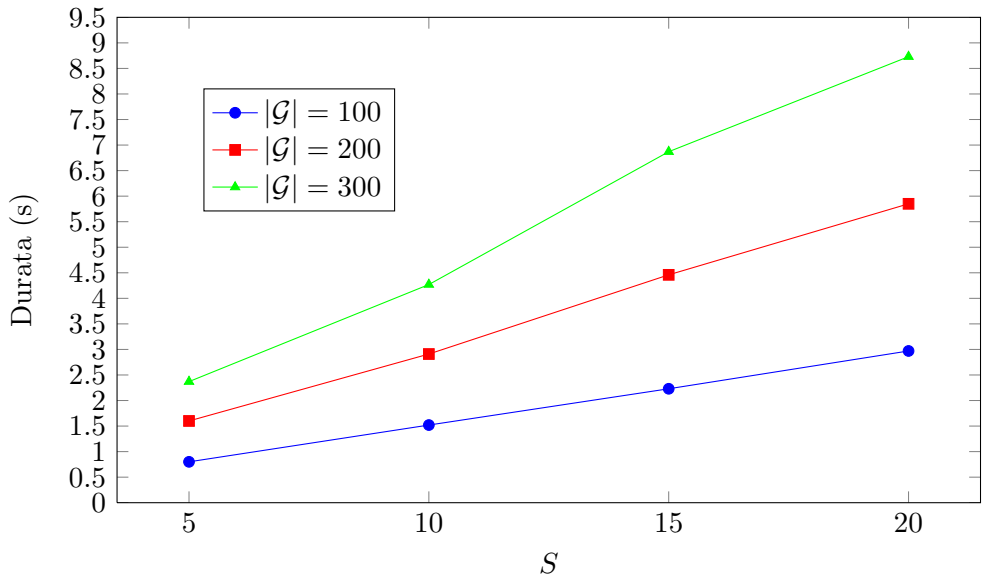

Figura 5.2: Durata della fase di generazione dell'albero al variare del numero di anelli Chord e fissato il numero di Gateway

SetupTree. Ogni misura è stata effettuata considerando un adeguato numero di istanze, in modo tale da ottenere intervalli di confidenza inferiori al 10%. Abbiamo misurato la durata della fase di generazione dell'albero al variare del numero di Gateway ($|\mathcal{G}|$) mantenendo fisso il numero di anelli Chord S e al variare del numero di anelli Chord S mantenendo fisso il numero di Gateway ($|\mathcal{G}|$). I risultati sono illustrati nelle Figure 5.1 e 5.2. Si nota subito un andamento lineare crescente in entrambe le Figure. Questo significa che la durata di creazione dell'alberi è lineare sia all'aumentare del numero di Gateway che all'aumentare del numero di anelli Chord. La crescita lineare in dipendenza al numero di Gateway è dovuta all'aumento della profondità dell'albero. La presenza di un maggior numero di Gateway sull'anello, infatti, porta ad un partizionamento dell'anello stesso, previsto dall'instradamento Chord, in intervalli più piccoli. La crescita lineare in dipendenza al numero di anelli Chord, invece, è dovuta al maggior carico computazionale a cui deve

far fronte ogni singolo Gateway, il quale deve elaborare un maggior numero di messaggi di *SetupTree*. Ogni anello Chord, infatti, porta alla creazione di un differente albero, rallentando la durata complessiva della creazione di ogni singolo albero.

5.5.2 SendAggregateShare ricevute

Per ogni singola istanza abbiamo considerato quale Gateway ha ricevuto il numero maggiore di messaggi di *SendAggregateShare* ed abbiamo misurato tale numero. Questo equivale a considerare il Gateway con il maggior numero di nodi figli complessivamente, ossia su tutti gli anelli Chord. Successivamente abbiamo mediato tale valore su un numero elevato di istanze in modo tale da ottenere intervalli di confidenza inferiori al 10%. Il valore ottenuto è la media del *fan-in* massimo dei Gateway. Abbiamo considerato come ipotesi il caso peggiore, ovvero che ogni Gateway sia collegato a un Meter che l'External Entity vuole monitorare.

Abbiamo misurato il numero di messaggi di *SendAggregateShare* ricevuti al variare del numero di Gateway con il numero di anelli Chord S fissato. I risultati, per valori di S differenti, sono illustrati in Figura 5.3. Come si può vedere, si ottiene un andamento pressochè costante. Considerando le curve relative a differenti valori di S si nota come, al crescere di S, aumenta il numero di SendAggregateShare ricevute. L'aumento del numero di messaggi di *SendAggregateShare* ricevute al crescere del numero degli anelli Chord è dovuto al fatto che, complessivamente, se il sistema prevede un numero maggiore di anelli Chord, ogni Gateway deve elaborare un numero di messaggi in ingresso proporzionale al numero di anelli.

Al contrario, il numero di messaggi di *SendAggregateShare* ricevuti da un Gateway, in linea di massima, non dipende dal numero di Gateway presenti

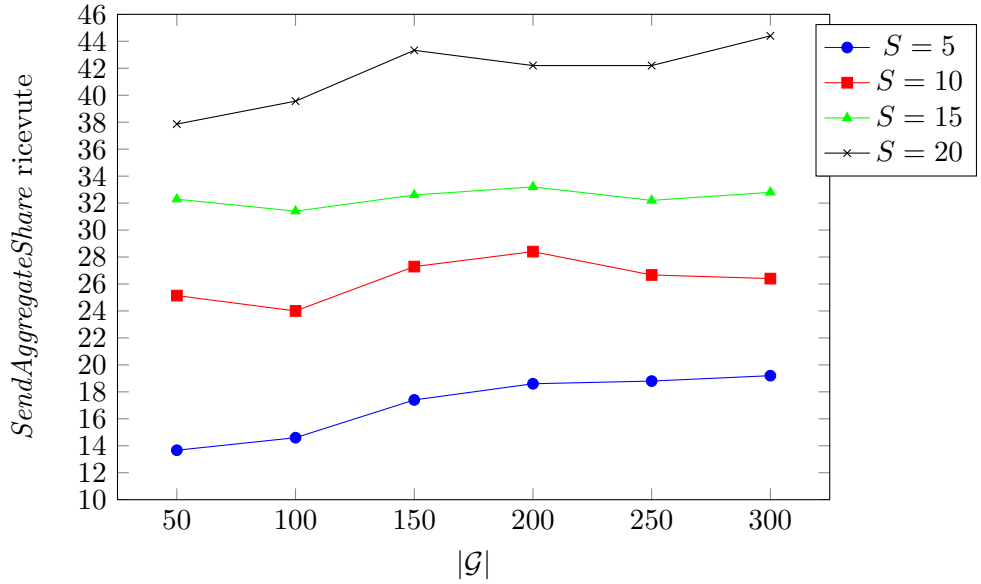

Figura 5.3: Numero di *SendAggregateShare* ricevute al variare del numero di Gateway e fissato il numero di anelli Chord

in rete. Questo perché, in generale, i nodi tendono ad equidistribuirsi sugli anelli, soprattutto per un numero di Gateway elevato. Infatti, nel caso di un numero di Gateway ridotto, la probabilità di avere uno sbilanciamento dei nodi sull'anello è superiore.

E' interessante notare come l'architettura distribuita si comporti, in termini di *fan-in* dei nodi aggregatori, molto meglio rispetto a quella centralizzata. Infatti, in quest'ultima, fissato $w = |\Omega_c|$, il fan-in dei PPN risulta deterministicamente uguale al numero di Producer $|\Pi_c|$.

5.5.3 Lunghezza dei messaggi

Abbiamo valutato la lunghezza dei messaggi del protocollo da noi implementati secondo lo standard testuale `Name: <value>`, sia per l'architettura centralizzata che per l'architettura distribuita, in modo tale da poter effettuare un confronto. I risultati sono ottenuti nelle Tabelle 5.2 e 5.3.

$\lvert\mathcal{M}_e\rvert$	SpecAggrRule	GrantRule	SetupTree	SendMeas	SendAggr
10	206	256	315	130	169
50	486	545	595	130	169
100	836	895	945	130	169
150	1186	1186	1295	130	169
200	1536	1536	1645	130	169
250	1886	1886	1995	130	169
300	2236	2236	2345	130	169
350	2586	2586	2695	130	169

Tabella 5.2: Lunghezza dei messaggi del protocollo (in byte) per l'architettura distribuita al variare del numero dei Meter

Nell'architettura distribuita (Tabella 5.2) la lunghezza dei messaggi di *SpecifyAggregationRule*, *GrantRule* e *SetupTree* cresce all'aumentare del numero di Meter $|\mathcal{M}_e|$ che sono monitorati dall'External Entity. Questo è dovuto al fatto che il messaggio di *SpecifyAggregationRule* trasporta l'identificatore di tutti i Meter che l'External Entity vuole monitorare. Lo stesso vale per i messaggi di *GrantRule* e di *SetupTree*: essi infatti trasportano il grant, ed uno dei campi che definisce il grant è proprio l'insieme degli identificatori dei Meter monitorati. La crescita delle dimensioni dei messaggi è lineare. I messaggi di *SendMeasurement* e *SendAggregateShare*, invece, hanno dimensione costante e indipendente dal numero di Meter $|\mathcal{M}_e|$. Si può notare che nessun messaggio dipende dal numero di Gateway, poiché i Gateway responsabili dell'aggregazione delle misure non sono definiti a priori dal Configurator ma vengono definiti durante la fase di creazione degli alberi, avviata dall'External Entity.

Nell'architettura centralizzata (Tabella 5.3), invece, la lunghezza dei mes-

$\lvert\Pi_c\rvert/\lvert\Omega_c\rvert$	SpecAggrRule	ConfPPN	ConfProd	SendShare	SendAggr
10	140	141	129	124	210
50	300	301	289	124	210
100	501	502	490	124	210
150	751	752	740	124	210
200	1001	1002	990	124	210
250	1251	1252	1240	124	210
300	1501	1502	1490	124	210
350	1751	1752	1740	124	210

Tabella 5.3: Lunghezza dei messaggi del protocollo (in byte) per l'architettura centralizzata al variare del numero dei Producer e dei PPN

saggi dipende sia dal numero di Producer $|\Pi_c|$ monitorati dal Consumer che dal numero di PPN $|\Omega_c|$ scelto a priori dal Configurator nella fase di setup e comunicato ai Producer. La lunghezza dei messaggi di *SpecifyAggregationRule* e di *ConfigurePPN* cresce con il numero di Producer monitorati dal Consumer, poiché essi trasportano l'identificatore di tutti i Producer che il Consumer vuole monitorare. Le dimensioni dei messaggi di *ConfigureProducer*, invece, crescono con il numero di PPN scelti dal Configurator per l'aggregazione, in quanto l'identificatore di ogni PPN viene comunicato ad ogni Producer per mezzo di questi messaggi. Per comodità, in Tabella, si è riportata solo una colonna per quanto riguarda il numero di Producer e il numero di PPN. A seconda del diverso messaggio considerato, in base a quanto appena detto, varia o il numero di Producer o il numero di PPN. Non si verifica mai contemporaneamente che la lunghezza dei messaggi dipenda da entrambi. La crescita delle dimensioni dei messaggi è lineare. I messaggi di *SendShare* e *SendAggregateShare*, invece, hanno dimensione costante e

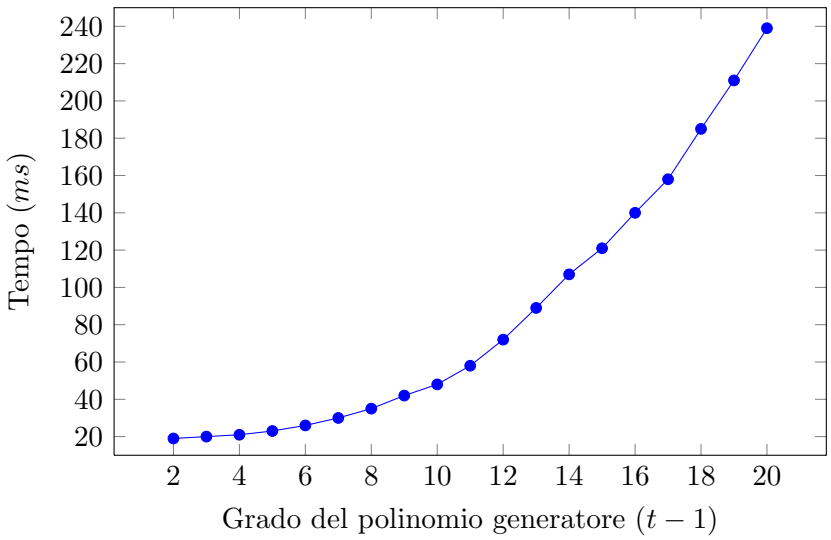

Figura 5.4: Tempo computazionale necessario per il recupero delle misure aggregate, utilizzando l'algoritmo di recupero robusto di Berlekamp-Welch, al variare del grado del polinomio generatore.

indipendente sia dal numero di Producer che dal numero di PPN.

Paragonando i risultati riportati nelle Tabelle 5.2 e 5.3 si nota come, in generale, la lunghezza dei messaggi nell'architettura distribuita sia superiore alla lunghezza dei messaggi nell'architettura centralizzata. Questo perché i messaggi per l'architettura distribuita trasportano più campi rispetto ai messaggi per l'architettura centralizzata, essendo la sua gestione più complessa.

5.5.4 Tempo di recupero delle misure aggregate

Negli emulatori, sia per l'architettura centralizzata che per l'architettura distribuita, abbiamo implementato l'algoritmo di Berlekamp-Welch (vedere Sezione 3.1.1) per il recupero robusto delle misure aggregate. Questo algoritmo prevede operazioni tra polinomi in aritmetica modulare e la risoluzione di un sistema lineare al fine di ottenere il polinomio generatore delle share

(Equazione 3.1), dal quale è possibile recuperare il valore delle share aggregate, essendo questo il termine noto di tale polinomio. Abbiamo quindi fatto variare il grado del polinomio, pari a $t-1$. Al variare del grado del polinomio varia anche il parametro $v = 3t - 2$, dipendendo esso da t, il quale indica il numero minimo di share necessarie per poter effettuare il recupero robusto per mezzo dell'algoritmo di Berlekamp-Welch. Il parametro v determina la dimensione del sistema lineare da risolvere.

I risultati sono illustrati nella Figura 5.4. Come si può osservare, si ha un andamento superlineare all'aumentare del grado del polinomio. Questo significa che, per garantire una maggiore robustezza, la quale permette il recupero in presenza di un maggior numero di errori, si ha un onere computazionale maggiore e crescente in modo superlineare.

Per effettuare queste misure, a differenza delle altre Sezioni, è stato utilizzato un computer con processore dual-core Intel Core 2 Duo U7600 con frequenza di clock per singolo core pari a 1.2 GHz e 2 Gb di memoria RAM.

5.5.5 Errori di link

In ultima analisi abbiamo trattato il caso, relativamente all'architettura centralizzata, in cui sono previsti errori di link tra Producer e PPN (Sezione 4.1.1): questo significa che con una certa probabilità la comunicazione tra un Producer e un PPN fallisce a causa di malfunzionamenti del link o a causa di eccessivi ritardi di trasmissione. Abbiamo considerato, per il recupero delle misure aggregate, l'algoritmo di recupero robusto di Berlekamp-Welch, considerando un numero w di share, e quindi di PPN, pari a $v = 3t - 2$ (Sezione 3.1.1). Il numero di PPN considerati è quindi il numero minimo che consente di effettuare il recupero per mezzo dell'algoritmo di Berlekamp-Welch. Per ottenere risultati significativi in tempi brevi e per un maggior numero di Pro-

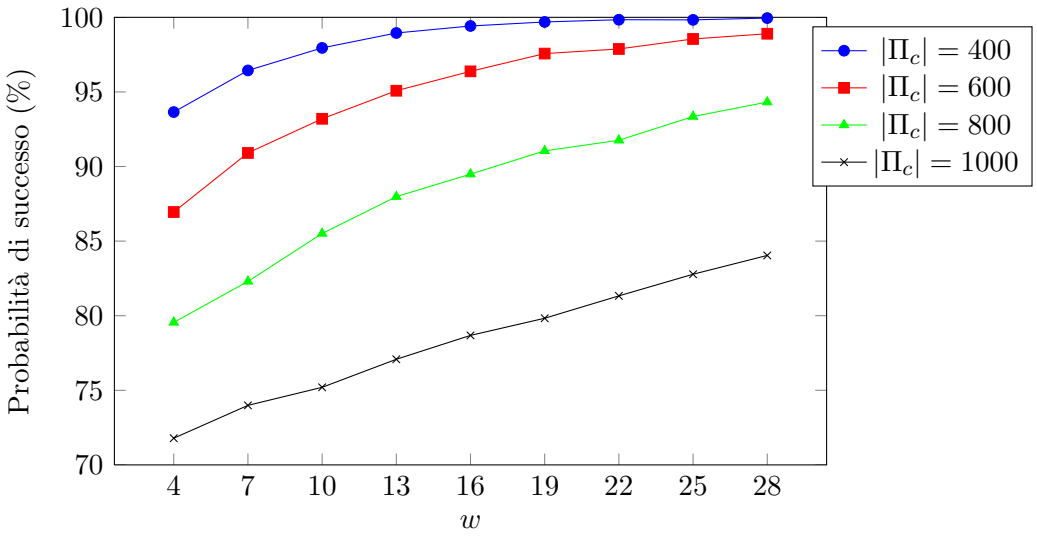

Figura 5.5: Probabilità di successo nel recupero delle misure aggregate, fissata una probabilità di guasto sul link pari a 10^{-4}.

ducer in rete, abbiamo estrapolato dall'emulatore il procedimento di invio e recupero robusto delle share, quindi lo abbiamo implementato in Matlab. I risultati ottenuti sono illustrati nelle Figure 5.5 e 5.6. La Figura 5.5 mostra la probabilità di successo in termini percentuali del recupero delle misure aggregate al variare del numero di share $w = v$ fissati il numero di Producer monitorati $|\Pi_c|$ e la probabilità di errore sul link $p = 10^{-4}$. Sul grafico vengono considerati i risultati per differenti valori di $|\Pi_c|$. Come si può notare, la probabilità di successo aumenta all'aumentare del numero di share w; a parità di w, invece, tale probabilità diminuisce all'aumentare del numero di Producer $|\Pi_c|$. L'aumento della probabilità di successo all'aumentare del numero di share w è dovuta all'aumento del numero di link in uscita dal singolo Producer, che comporta un aumento della probabilità che un sottoinsieme t di PPN esegua l'aggregazione sullo stesso insieme di Producer. Al contrario, la diminuzione della probabilità di successo all'aumentare del numero di Producer è causata dal fatto che diminuisce la probabilità che un numero t

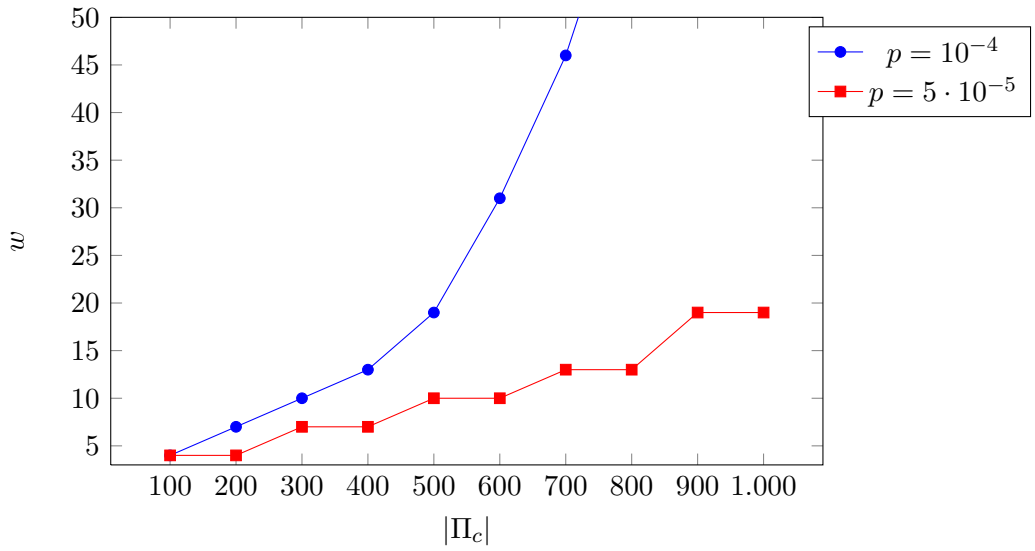

Figura 5.6: Numero di share necessario a garantire una probabilità di successo nel recupero delle misure aggregate superiore al 99% per differenti valori di probabilità di errore del link.

di PPN, soglia minima imposta dallo schema a soglia di Shamir per il recupero delle misure aggregate, esegua l'aggregazione sullo stesso insieme di Producer. La Figura 5.6, invece, mostra il numero $w = v$ di share necessario a garantire una probabilità di successo nel recupero delle misure aggregate superiore al 99% al variare del numero di Producer $|\Pi_c|$. Nella Figura sono illustrati i risultati per due differenti valori di probabilità p di errore sul singolo link. Come si può osservare, una diretta conseguenza di quanto detto prima è che a parità di p il numero di share w aumenta all'aumentare del numero di Producer. Inoltre, al diminuire di p, come è logico aspettarsi, a parità di $|\Pi_c|$ diminuisce il valore di w. In particolare si osserva che, anche solo dimezzando la probabilità di errore sul link, si ottengono scostamenti di w molto significativi per un numero di Producer elevato. Abbiamo emulato anche il caso in cui $p = 10^{-5}$ ma, avendo ottenuto $w = 4$ per ogni valore di $|\Pi_c| \leq 1000$, non l'abbiamo riportato sul grafico.

CAPITOLO 6

ATTACCO ALLA PRIVACY DIFFERENZIALE

6.1 Modello gaussiano

Nel Capitolo 3 e più precisamente nella Sezione 3.3 sono stati illustrati i principi chiave della privacy differenziale, la quale permette ad un utente di partecipare con le proprie misure ad un aggregato di misure senza che la propria presenza influenzi troppo la misura aggregata stessa, rendendo quindi impossibile ad un attaccante che dispone di informazione aggiuntiva (ausiliaria) oltre all'aggregato di comprendere se la misura dell'utente è presente o meno nella misura aggregata. A tal fine, le misure degli utenti vengono randomizzate per mezzo dell'aggiunta di un rumore opportunamente dimensionato (Sezione 3.3.2), il quale da un lato permette di garantire privacy differenziale per gli utenti ma dall'altro rischia di compromettere l'u-

tilità statistica della misura aggregata. Lo schema illustrato nella Sezione 3.3 consente di garantire privacy differenziale e di avere un errore tra la misura aggregata rumorosa e la misura aggregata nominale ridotto, come illustrato in [12].

Lo schema descritto nel Capitolo 3 può essere adottato senza alcun problema dalle architetture di rete descritte nel Capitolo 4. Infatti, la suddivisione delle misure di contatori in share per mezzo dello schema a soglia di Shamir e la presenza di nodi intermedi aggregatori di share come i PPN o i Gateway è assolutamente trasparente rispetto allo schema sopra citato. Ciò significa che una volta recuperata la misura aggregata dal Consumer o dall'External Entity è possibile ricondursi a tale schema, secondo cui i dati provenienti da più partecipanti vengono aggregati da un nodo aggregatore. Le architetture viste in precedenza hanno il solo scopo di evitare che il Consumer o l'External Entity (nodo aggregatore) possa risalire alle misure del singolo utente che compete all'aggregazione, in quanto è assunto essere non affidabile. Questa informazione, però, può essere disponibile al Consumer o all'External Entity come informazione ausiliaria proveniente da fonti esterne all'architettura.

Supponiamo per l'aggregatore il modello di attaccante *onesto-ma-curioso*. Un attaccante di questo genere segue alla lettera il protocollo, ma cerca di ottenere informazioni dalla traccia delle misure aggregate rumorose ottenute: ipotizziamo siano rese disponibili all'aggregatore T misure aggregate rumorose ottenute in altrettanti slot temporali consecutivi. Se l'aggregatore, pur non conoscendo nello specifico l'andamento nel tempo delle misure di ogni singolo utente, ha a disposizione informazioni sulle statistiche dell'insieme di misure per un singolo utente come il valor medio, la varianza e la correlazione tra i campioni, può sfruttarle per cercare di ridurre o addirittura rimuovere il rumore dalla misura aggregata al fine di abbassare il livello di privacy diffe-

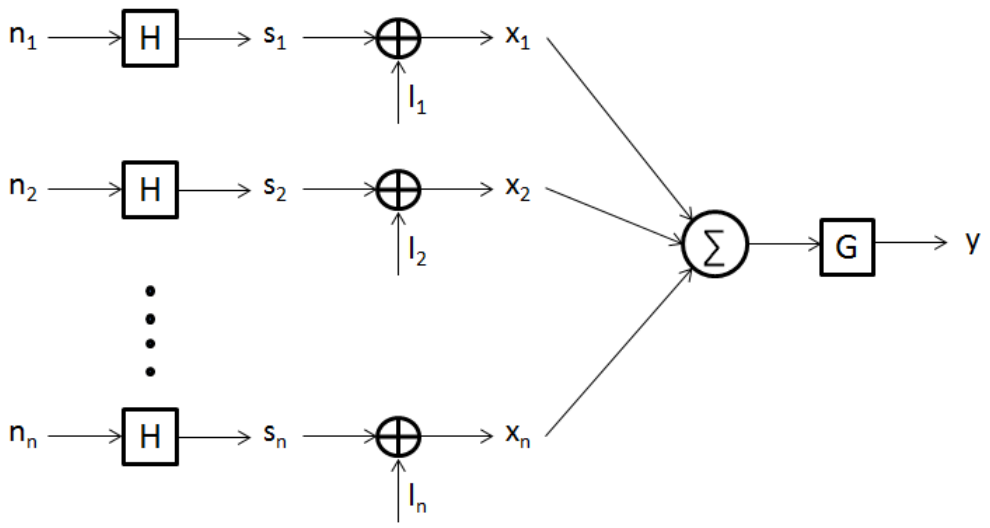

Figura 6.1: Modello gaussiano

renziale degli utenti. Si ipotizza che l'attaccante conosca anche le statistiche proprie del rumore additivo (Sezione 3.3.2).

Da questo momento in poi faremo riferimento alla terminologia comunemente utilizzata nella teoria dei segnali: un singolo valore (misura) prende il nome di *campione*, più campioni prendono il nome di *segnale*.

6.1.1 Modello e attacco

Il modello è illustrato nella figura 6.1. Supponiamo la presenza di N partecipanti e che il segnale proveniente da ogni singolo contatore i sia modellato da un processo casuale $s_i[t]$ ottenuto come output di un filtro digitale lineare e tempoinvariante (LTI) H: t è l'indice temporale con $t \in Z$. L'input $n_i[t]$ di tale filtro è un processo casuale gaussiano con autocorrelazione impulsiva (ovvero con i campioni tutti scorrelati tra di loro) con valor medio μ_n e varianza σ_n^2. Si noti che anche $s_i[t]$ sarà allora un processo casuale gaussiano, in quanto un processo casuale gaussiano filtrato da un filtro LTI

produce in output un processo casuale ancora gaussiano [18]. Supponiamo le stesse statistiche per ogni singolo contatore i e indipendenza spaziale tra di essi. Ciò significa che ogni campione in un determinato istante di tempo per un partecipante è indipendente dal campione di ogni altro partecipante nel medesimo istante di tempo.

Ad $s_i[t]$ viene sommato un rumore geometrico simmetrico a media nulla $l_i[t]$ come descritto nella Sezione 3.3.2. Si ottiene quindi il processo casuale $x_i[t]$ che rappresenta il segnale rumoroso. Per ogni istante di tempo $\bar{t}$, i campioni rumorosi $x_1[\bar{t}], \ldots, x_N[\bar{t}]$ sono sommati e forniti all'attaccante. L'attaccante, una volta ottenuti T valori aggregati per gli istanti di tempo $1, \ldots, T$, cerca di ridurre il rumore per ogni singolo valore aggregato dando in input tali valori a un filtro digitale LTI G. Il processo casuale output del filtro G vale

$$y[t] = \left(\sum_{i=1}^{N} x_i[t] \right) * g[t]$$

dove $g[t]$ è la risposta all'impulso del filtro G e il simbolo $*$ indica l'operazione di *convoluzione*. Le proprietà LTI del filtro permettono di scrivere:

$$y[t] = \left(\sum_{i=1}^{N} x_i[t] \right) * g[t] = \sum_{i=1}^{N} (x_i[t] * g[t]) = \sum_{i=1}^{N} ((s_i[t] + l_i[t]) * g[t])$$
$$= \sum_{i=1}^{N} (s_i[t] * g[t] + l_i[t] * g[t]) = \sum_{i=1}^{N} (n_i[t] * h[t] * g[t] + l_i[t] * g[t])$$

Chiamiamo quindi $y_{s_i}[t] = n_i[t] * h[t] * g[t]$ e $y_{l_i}[t] = l_i[t] * g[t]$ per esplicitare la parte di segnale e la parte di rumore del contributo i-simo in $y[t]$, ottenendo:

$$y[t] = \sum_{i=1}^{N} (y_{s_i}[t] + y_{l_i}[t])$$

Analizziamo quindi in termini statistici il processo casuale $y[t]$, determinando il suo valor medio $\mathrm{E}[y[t]]$ e la sua varianza $\mathrm{var}[y[t]]$. Essendo gli i

partecipanti caratterizzati tutti allo stesso modo, il valor medio vale:

$$\mathrm{E}[y[t]] = \mathrm{E}\left[\sum_{i=1}^{N}(y_{s_i}[t] + y_{l_i}[t])\right] = \sum_{i=1}^{N}\mathrm{E}[y_{s_i}[t] + y_{l_i}[t]] = N \cdot \mathrm{E}[y_s[t] + y_l[t]] =$$
$$= N \cdot \mathrm{E}[y_s[t]] + N \cdot \mathrm{E}[y_l[t]] = N \cdot \mu_{y_s} + N \cdot \mu_{y_l}$$

Grazie alle proprietà dei filtri LTI attraversati da processi casuali, si possono facilmente calcolare μ_{y_s} e μ_{y_l}:

$$\mu_{y_s} = \mathrm{E}[n_i[t] * h[t] * g[t]] = \mu_n \cdot \mathcal{H}(0) \cdot \mathcal{G}(0)$$

$$\mu_{y_l} = \mathrm{E}[l_i[t] * g[t]] = \mu_l \cdot \mathcal{G}(0) = 0$$

Ricapitolando:

$$\mathrm{E}[y[t]] = N \cdot \mu_n \cdot \mathcal{H}(0) \cdot \mathcal{G}(0)$$

dove $\mathcal{G}(0)$ e $\mathcal{H}(0)$ sono le risposte in frequenza dei filtri G e H in $f = 0$. Il valor medio della parte di rumore, essendo $\mu_l = 0$ (rumore a media nulla), rimarrà nullo.

Per quanto riguarda la varianza, grazie all'ipotesi di indipendenza spaziale tra i contatori formulata in precedenza, alla mutua indipendenza statistica tra i processi $s_i[t]$ ed $l_i[t]$ e alla medesima caratterizzazione statistica per tutti i partecipanti, possiamo scrivere:

$$\mathrm{var}[y[t]] = \mathrm{var}\left[\sum_{i=1}^{N}(y_{s_i}[t] + y_{l_i}[t])\right] = \sum_{i=1}^{N}\mathrm{var}[y_{s_i}[t] + y_{l_i}[t]] = N \cdot \mathrm{var}[y_s[t] + y_l[t]]$$
$$= N \cdot \mathrm{var}[y_s[t]] + N \cdot \mathrm{var}[y_l[t]] = N \cdot \sigma^2_{y_s} + N \cdot \sigma^2_{y_l}$$

Calcoliamo $\sigma^2_{y_s}$ come:

$$\sigma^2_{y_s} = R_{y_s}(0) - \mu^2_{y_s}$$

E' necessario calcolare $R_{y_s}(\tau)$, ovvero l'autocorrelazione di $y_s[t]$ e considerarne in valore in $\tau = 0$. Per le proprietà dei sistemi LTI vale:

$$R_{y_s}(\tau) = R_s(\tau) * \mathcal{F}^{-1}\left\{|\mathcal{G}(f)|^2\right\}$$

Allo stesso modo si può scrivere l'espressione per $R_s(\tau)$, infatti:

$$R_s(\tau) = R_n(\tau) * \mathcal{F}^{-1}\left\{|\mathcal{H}(f)|^2\right\}$$

Il simbolo $\mathcal{F}^{-1}\{\cdot\}$ indica che è stata effettuata sull'argomento la Trasformata di Fourier Inversa. Si possono combinare le due precedenti espressioni ottenendo:

$$\begin{aligned} R_{y_s}(\tau) &= R_n(\tau) * \mathcal{F}^{-1}\left\{|\mathcal{H}(f)|^2\right\} * \mathcal{F}^{-1}\left\{|\mathcal{G}(f)|^2\right\} \\ &= R_n(\tau) * \mathcal{F}^{-1}\left\{|\mathcal{H}(f)|^2 \cdot |\mathcal{G}(f)|^2\right\} \end{aligned}$$

Data l'ipotesi di autocorrelazione impulsiva per $n_i[t]$, ovvero:

$$R_n(\tau) = \sigma_n^2 \cdot \delta(\tau) + \mu_n^2$$

è possibile scrivere:

$$\begin{aligned} R_{y_s}(\tau) &= (\sigma_n^2 \cdot \delta(\tau) + \mu_n^2) * \mathcal{F}^{-1}\left\{|\mathcal{H}(f)|^2 \cdot |\mathcal{G}(f)|^2\right\} \\ &= (\sigma_n^2 \cdot \delta(\tau)) * \mathcal{F}^{-1}\left\{|\mathcal{H}(f)|^2 \cdot |\mathcal{G}(f)|^2\right\} + \mu_n^2 * \mathcal{F}^{-1}\left\{|\mathcal{H}(f)|^2 \cdot |\mathcal{G}(f)|^2\right\} \\ &= \sigma_n^2 \cdot \mathcal{F}^{-1}\left\{|\mathcal{H}(f)|^2 \cdot |\mathcal{G}(f)|^2\right\} + \mu_n^2 \cdot |\mathcal{H}(0)|^2 \cdot |\mathcal{G}(0)|^2 \end{aligned}$$

Si ottiene allora:

$$R_{y_s}(0) = \sigma_n^2 \cdot \mathcal{F}^{-1}\left\{|\mathcal{H}(f)|^2 \cdot |\mathcal{G}(f)|^2\right\} \mid_{\tau=0} + \mu_n^2 \cdot |\mathcal{H}(0)|^2 \cdot |\mathcal{G}(0)|^2$$

E' allora possibile calcolare:

$$\begin{aligned} \sigma_{y_s}^2 &= R_{y_s}(0) - \mu_{y_s}^2 \\ &= \sigma_n^2 \cdot \mathcal{F}^{-1}\left\{|\mathcal{H}(f)|^2 \cdot |\mathcal{G}(f)|^2\right\} \mid_{\tau=0} + \\ &+ \mu_n^2 \cdot |\mathcal{H}(0)|^2 \cdot |\mathcal{G}(0)|^2 - \mu_n^2 \cdot \mathcal{H}^2(0) \cdot \mathcal{G}^2(0) \end{aligned}$$

Se si ipotizza che la risposta all'impulso dei filtri digitali H e G sia caratterizzata esclusivamente da campioni reali, per il Teorema di Parseval $|\mathcal{H}(0)| = \mathcal{H}(0)$ e $|\mathcal{G}(0)| = \mathcal{G}(0)$ e l'espressione si riduce a:

$$\sigma_{y_s}^2 = \sigma_n^2 \cdot \mathcal{F}^{-1}\left\{|\mathcal{H}(f)|^2 \cdot |\mathcal{G}(f)|^2\right\} \mid_{\tau=0}$$

Calcoliamo $\sigma^2_{y_l}$ come:

$$\sigma^2_{y_l} = R_{y_l}(0) - \mu^2_{y_l} = R_{y_l}(0)$$

Otteniamo quindi l'autocorrelazione del rumore filtrato:

$$R_{y_l}(\tau) = R_l(\tau) * \mathcal{F}^{-1}\left\{|\mathcal{G}(f)|^2\right\}$$

Data l'ipotesi di autocorrelazione impulsiva per $l_i[t]$ e di valor medio nullo si può scrivere:

$$R_l(\tau) = \sigma^2_l \cdot \delta(\tau)$$

Quindi:

$$R_{y_l}(\tau) = \sigma^2_l \cdot \delta(\tau) * \mathcal{F}^{-1}\left\{|\mathcal{G}(f)|^2\right\} = \sigma^2_{y_l} \cdot \mathcal{F}^{-1}\left\{|\mathcal{G}(f)|^2\right\}$$

Si ottiene allora, per il rumore filtrato, la varianza:

$$\sigma^2_{y_l} = R_{y_l}(0) = \sigma^2_{y_l} \cdot \mathcal{F}^{-1}\left\{|\mathcal{G}(f)|^2\right\} \mid_{\tau=0}$$

Esprimiamo quindi il rapporto tra le varianze $\sigma^2_{y_s}$ e $\sigma^2_{y_l}$ e lo chiamiamo SNR:

$$SNR = \frac{\sigma^2_{y_s}}{\sigma^2_{y_l}} = \frac{\sigma^2_n \cdot \mathcal{F}^{-1}\left\{|\mathcal{H}(f)|^2 \cdot |\mathcal{G}(f)|^2\right\} \mid_{\tau=0}}{\sigma^2_l \cdot \mathcal{F}^{-1}\left\{|\mathcal{G}(f)|^2\right\} \mid_{\tau=0}}$$

In questo modo abbiamo espresso il rapporto tra le varianze della parte di segnale e della parte di rumore di $y[t]$.

E' necessario quindi definire quale filtro G verrà scelto dall'attaccante per ridurre il più possibile il rumore $l_i[t]$. Se si suppone che l'attaccante conosca il filtro H e quindi la sua risposta all'impulso, la scelta potrebbe ricadere sul *Filtro Inverso*:

$$\mathcal{G}(f) = \frac{1}{\mathcal{H}(f)}$$

Questo filtro, però, presenta problemi in caso di componenti di rumore ad alta frequenza, in quanto esse vengono amplificate e la stima del segnale peggiora notevolmente.

Il filtro ottimo che viene utilizzato è il *Filtro di Wiener* [25]. Tale filtro ottimizza le proprie prestazioni in termini di errore quadratico medio (MSE, mean square error) della stima ed ha la seguente espressione:

$$\mathcal{G}(f) = \frac{1}{\mathcal{H}(f)} \cdot \frac{|\mathcal{H}(f)|^2}{|\mathcal{H}(f)|^2 + \frac{S_l(f)}{S_n(f)}}$$

dove $S_l(f)$ e $S_n(f)$ sono rispettivamente la densità spettrale di potenza (DSP) del rumore $l_i[t]$ e la densità spettrale di potenza del segnale $n_i[t]$ in input ad H. $S_l(f)$ è supposta nota in quanto, come ipotizzato in precedenza, l'attaccante conosce la descrizione statistica di $l_i[t]$. La DSP è facilmente ottenibile in quanto è la Trasformata di Fourier dell'autocorrelazione $R_l(\tau)$, quindi:

$$S_l(f) = \mathcal{F}\{R_l(\tau)\} = \mathcal{F}\left\{\sigma_l^2 \cdot \delta(\tau)\right\} = \sigma_l^2$$

E' interessante notare come, in assenza di rumore $l_i[t]$, ovvero nel caso in cui $S_l(f) = 0$, l'espressione del Filtro di Wiener si riduca ad essere equivalente all'espressione del Filtro Inverso.

Viene ipotizzato anche che l'attaccante conosca una descrizione statistica di $n_i[t]$, ricavata da informazione ausiliaria. Questo significa che esso possiede una stima di media, varianza ed autocorrelazione di $n_i[t]$ e da queste informazioni può ricavare una stima della DSP di $n_i[t]$, che vale:

$$S_n(f) = \mathcal{F}\{R_n(\tau)\} = \mathcal{F}\left\{\sigma_n^2 \cdot \delta(\tau) + \mu_n^2\right\} = \sigma_n^2 + \mu_n^2 \cdot \delta(f)$$

In base a queste informazioni, l'attaccante è in grado di ottenere una stima di $\sum_{i=1}^{N} n_i[t]$ che, filtrata per mezzo del filtro H, fornisce una stima del segnale $\sum_{i=1}^{N} s_i[t]$ (Figura 6.2).

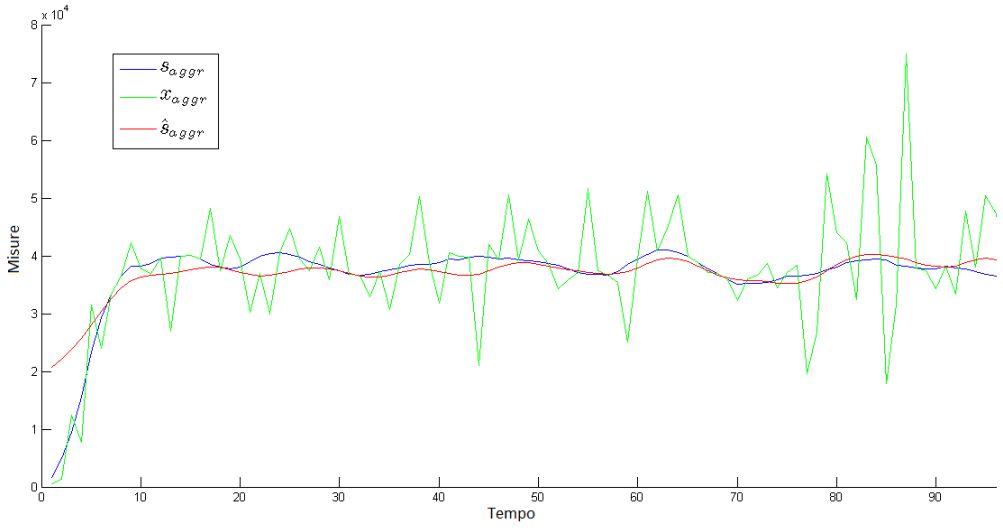

Figura 6.2: Stima del segnale aggregato per mezzo del Filtro di Wiener

6.1.2 Difesa dall'attacco

Un modello di attacco come quello descritto nella Sezione 6.1.1 permette all'attaccante di ridurre notevolmente il livello di privacy differenziale delle misure aggregate, dato che è in grado di ridurre il livello di rumore da esse. Abbiamo quindi pensato ad una possibile difesa che non consenta all'attaccante, anche nel caso in cui venga utilizzato il Filtro di Wiener, di ridurre il livello di privacy differenziale degli utenti. Una possibile soluzione, semplice ma efficace, è di filtrare il rumore $l_i'[t]$ per mezzo del filtro H (Figura 6.3) prima di sommarlo al segnale $s_i[t]$. In questo modo si crea un rumore $l_i[t]$ con campioni correlati che non può essere rimosso dal Filtro di Wiener, in quanto l'utilizzo di tale filtro permette di ottenere al meglio una stima del processo $s_i[t] + l_i[t]$. Tale concetto può essere facilmente compreso se si ragiona sulle DSP dei processi in gioco, ovvero:

$$S_n(f) = \mathcal{F}\{R_n(\tau)\}$$

$$S_s(f) = \mathcal{F}\{R_s(\tau)\}$$

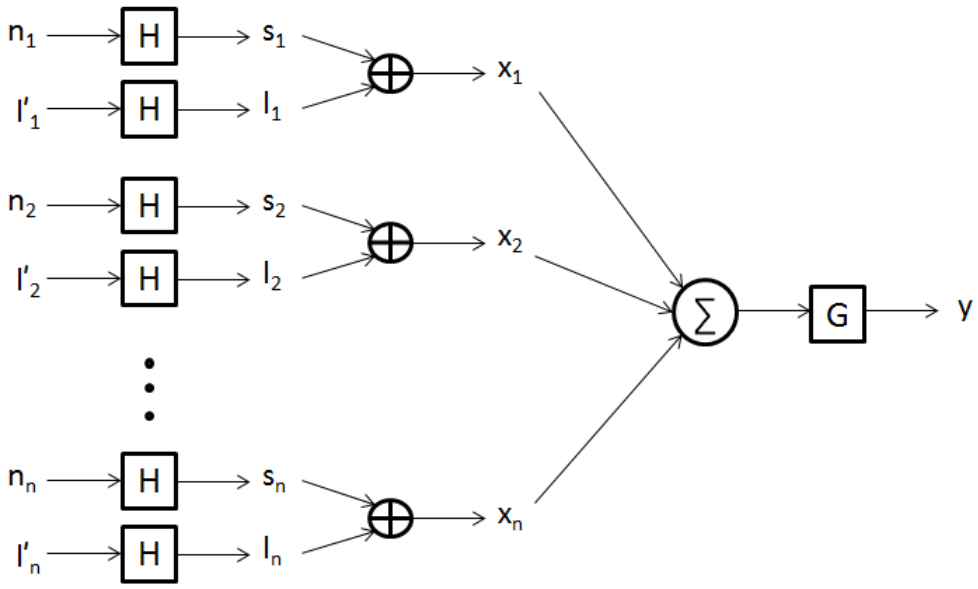

Figura 6.3: Difesa per il modello gaussiano con rumore prefiltrato per mezzo del filtro H

$$S_l(f) = \mathcal{F}\{R_l(\tau)\}$$

$$S_y(f) = \mathcal{F}\{R_y(\tau)\}$$

Vogliamo vedere come è fatta $S_y(f)$. Per le proprietà dei sistemi LTI attraversati da processi casuali si può scrivere:

$$S_y(f) = S_x(f) \cdot |\mathcal{G}(f)|^2$$

$$S_x(f) = S_n(f) \cdot |\mathcal{H}(f)|^2 + S_l(f)$$

Combinando le due espressioni si ottiene:

$$S_y(f) = \underbrace{S_n(f) \cdot |\mathcal{H}(f)|^2 \cdot |\mathcal{G}(f)|^2}_{Segnale} + \underbrace{S_l(f) \cdot |\mathcal{G}(f)|^2}_{Rumore}$$

Da tale espressione per $S_y(f)$ si può facilmente osservare che la DSP della parte di rumore, una volta che il processo aggregato delle $x_i[t]$ viene fatto passare dal filtro G, è notevolmente ridotta rispetto alla DSP della parte di segnale.

Nel caso in cui anche il rumore $l_i'[t]$ venga filtrato per mezzo di H prima di essere sommato al processo $s_i[t]$, l'espressione di $S_x(f)$ cambia e diventa:

$$S_x(f) = S_n(f) \cdot |\mathcal{H}(f)|^2 + S_{l'}(f) \cdot |\mathcal{H}(f)|^2$$

Quindi:

$$S_y(f) = \underbrace{S_n(f) \cdot |\mathcal{H}(f)|^2 \cdot |\mathcal{G}(f)|^2}_{Segnale} + \underbrace{S_{l'}(f) \cdot |\mathcal{H}(f)|^2 |\mathcal{G}(f)|^2}_{Rumore}$$

In questo caso le DSP della parte di rumore e della parte di segnale sono scalate dello stesso valore $|\mathcal{H}(f)|^2 \cdot |\mathcal{G}(f)|^2$ e l'attacco è reso inefficiente.

Lo stesso risultato può essere mostrato valutando il rapporto tra le varianze come definito nella Sezione 6.1.1. Senza svolgere nuovamente tutti i calcoli, è semplice verificare che si ottiene:

$$SNR = \frac{\sigma_{y_s}^2}{\sigma_{y_l}^2} = \frac{\sigma_n^2 \cdot \mathcal{F}^{-1}\left\{|\mathcal{H}(f)|^2 \cdot |\mathcal{G}(f)|^2\right\}|_{\tau=0}}{\sigma_{l'}^2 \cdot \mathcal{F}^{-1}\left\{|\mathcal{H}(f)|^2 \cdot |\mathcal{G}(f)|^2\right\}|_{\tau=0}} = \frac{\sigma_n^2}{\sigma_{l'}^2}$$

Questo significa che il rapporto tra le varianze di segnale e rumore prima del filtraggio per mezzo dei filtri H e G è invariato rispetto al rapporto tra le varianze della parte di segnale e della parte di rumore dopo il filtraggio con i filtri H e G, rendendo inefficace l'attacco per mezzo del Filtro di Wiener.

6.1.3 La Sensitività: nuova definizione

La sensitività Δ è stata definita nella Sezione 3.3.1. Nel modello da noi considerato abbiamo a che fare, relativamente alla componente di segnale, con processi casuali gaussiani. E' quindi impossibile riferirsi alla sensitività per come definita in letteratura, in quanto il segnale può assumere tutti i valori dell'asse reale: è necessario modificarne la definizione facendo leva sulle caratteristiche statistiche dei segnali in gioco.

Partiamo dalla definizione di sensitività data in [12], ovvero:

$$\Delta = \max_{\boldsymbol{x},\boldsymbol{y}} |\,\mathrm{sum}(\boldsymbol{x}) - \mathrm{sum}(\boldsymbol{y})|$$

dove $\mathrm{sum}(\boldsymbol{x}) = \sum_{i=1}^{n} x_i[\bar{t}]$, fissato un istante temporale $\bar{t}$. Ricordiamo che $\boldsymbol{x}$ e $\boldsymbol{y}$ sono due vettori *vicini*, ovvero che differiscono di un solo elemento (campione). Tale definizione di sensitività indica il massimo cambiamento che un singolo elemento può portare sulla somma aggregata, e quindi coincide con il massimo valore di scostamento che un singolo elemento può avere.

Se consideriamo processi casuali con variabili distribuite secondo una distribuzione gaussiana, in linea teorica questo scostamento è infinito, in quanto la densità di probabilità gaussiana è illimitata e un singolo campione può assumere un qualsiasi valore reale. In termini pratici, però, la probabilità che lo scostamento Δ assuma valori superiori ad una determinata soglia è trascurabile. Ci occupiamo quindi di come definire tale soglia. Chiamiamo $s_1[\bar{t}]$ ed $s_2[\bar{t}]$ gli elementi differenti in $\boldsymbol{x}$ e $\boldsymbol{y}$, dove $s_1[\bar{t}]$ è presente nel vettore $\boldsymbol{x}$ e $s_2[\bar{t}]$ è presente nel vettore $\boldsymbol{y}$. Si può allora scrivere:

$$\mathrm{sum}(\boldsymbol{x}) - \mathrm{sum}(\boldsymbol{y}) = s_1[\bar{t}] - s_2[\bar{t}] = s_\Delta[\bar{t}]$$

A questo punto calcoliamo il valor medio $\mathrm{E}[s_\Delta[\bar{t}]]$ e la varianza $\mathrm{var}[s_\Delta[\bar{t}]]$, ottenute come differenza tra due variabili gaussiane entrambe con media $\mu_s = \mu_n \cdot \mathcal{H}(0)$ e varianza $\sigma_s^2 = \sigma_n^2 \cdot \mathcal{F}^{-1}\left\{|\mathcal{H}(f)|^2\right\}\,|_{\tau=0}$, in quanto le due variabili presentano la medesima caratterizzazione statistica. Inoltre, poiché le due variabili sono indipendenti, si ottiene:

$$\mathrm{E}[s_\Delta[\bar{t}]] = \mathrm{E}[s_1[\bar{t}] - s_2[\bar{t}]] = \mu_s - \mu_s = 0$$

$$\mathrm{var}[s_\Delta[\bar{t}]] = \mathrm{var}[s_1[\bar{t}] - s_2[\bar{t}]] = \sigma_s^2 + \sigma_s^2 = 2\sigma_s^2$$

Questo significa che $s_1[\bar{t}] - s_2[\bar{t}]$ ha distribuzione gaussiana a valor medio nullo e deviazione standard doppia rispetto al singolo campione.

Una volta definita $s_\Delta[\bar{t}]$ in questo modo, è possibile definire la sensitività Δ come:

$$\Delta = \gamma : \Pr\left\{\max_{\boldsymbol{x},\boldsymbol{y}} |\,\mathrm{sum}(\boldsymbol{x}) - \mathrm{sum}(\boldsymbol{y})| > \gamma\right\} < \alpha$$

con γ numero reale positivo e α definito piccolo a piacere.

Ciò significa che si sceglie la sensitività Δ in modo tale per cui la probabilità di avere uno scostamento per un elemento superiore a $\Delta = \gamma$ è inferiore a una probabilità fissata α. Questo porta ad una diminuzione di Δ all'aumentare di α. Ovviamente la probabilità α deve essere mantenuta piccola altrimenti la sensitività Δ, che è il parametro che lega il segnale all'ampiezza del rumore da sommare al segnale stesso per garantire privacy differenziale (vedere Sezione 3.3.2), non è più un parametro significativo e porta ad un'alterazione dei risultati ottenuti dalla teoria come elaborata in [11, 12].

6.1.4 Problema decisionale

Dopo aver discusso il modello di attacco e una possibile difesa per preservare la privacy differenziale anche in presenza di attacco è necessario valutare le prestazioni e l'efficienza di attacco e difesa. Il più probabile scenario di attacco è un aggregatore che cerca di ricostruire la sequenza di misure di un singolo utente. Tale problema è un problema computazionale. Stabilire una metrica di efficacia di un attacco a un problema computazionale è, in generale, più difficile e meno intuitivo che stabilire una metrica di successo di un attacco a un problema decisionale. Pertanto ci siamo concentrati su un *problema decisionale*, nell'ipotesi che, se il problema decisionale è di difficile soluzione, lo sia anche il corrispondente problema computazionale.

Definizione 6.1. Problema decisionale. *L'attaccante ha a disposizione T campioni dei due aggregati rumorosi*

$$X_a[t] = \sum_{i=1}^{N-1} x_i[t] + x_a[t] = \sum_{i=1}^{N-1} x_i[t] + (s_a[t] + l_a[t])$$

$$X_b[t] = \sum_{i=1}^{N-1} x_i[t] + x_b[t] = \sum_{i=1}^{N-1} x_i[t] + (s_b[t] + l_b[t])$$

Tali aggregati rumorosi, entrambi relativi a N partecipanti, differiscono per un solo partecipante: nel primo è presente il partecipante a e nel secondo è presente il partecipante b. All'attaccante vengono forniti anche come informazione ausiliaria i T campioni di $s_a[t]$ sui quali è stata effettuata l'aggregazione nel primo gruppo. Viene richiesto all'attaccante di decidere, in base a queste informazioni, in quale delle due misure aggregate è presente il segnale $s_a[t]$.

Un problema decisionale di questo tipo valuta il livello di sicurezza per un singolo utente che partecipa all'aggregazione. L'obiettivo della privacy differenziale è non consentire all'attaccante di sapere se un segnale relativo a un singolo utente, da esso recuperato in qualche modo che esula dalla nostra trattazione, è presente o meno in un insieme di dati rumorosi aggregati. In questo modo la privacy dell'utente è preservata anche se l'attaccante possiede, rispetto a quell'utente, l'informazione ausiliaria più significativa, ovvero l'insieme stesso di misure non rumorose.

Dopo aver definito il problema decisionale è necessario definire il *decisore* che porta l'attaccante a fare la sua scelta. Viene scelto un decisore molto semplice che sfrutta la *cross-correlazione* (vedere Sezione 3.3.3) tra i segnali. L'attaccante effettua la cross-correlazione tra $s_a[t]$ e $X_a[t]$ e tra $s_a[t]$ e $X_b[t]$, ovvero calcola

$$R_{s_a,X_a}[m] = \sum_{m=-t}^{m=+t} s_a[t+m] \cdot X_a[t]$$

$$R_{s_a,X_b}[m] = \sum_{m=-t}^{m=+t} s_a[t+m] \cdot X_b[t]$$

Viene effettuata questa operazione perché, per la linearità del sistema, effettuare la cross-correlazione tra un segnale e un segnale aggregato relativo a più partecipanti è equivalente ad effettuare la cross-correlazione tra il segnale ed ogni singolo segnale rumoroso che compete all'aggregazione e successivamente sommare i contributi di cross-correlazione ottenuti. Per preservare l'utilità del segnale aggregato, il rumore generico $l_i[t]$ sommato al segnale generico $s_i[t]$ non sarà tale da offuscare completamente il segnale aggregato e l'errore sulla misura, come definito in [12], sarà mantenuto limitato. L'attaccante si aspetta quindi che la cross-correlazione abbia un picco più elevato in corrispondenza di $m = 0$ quando viene effettuata tra $s_a[t]$ e il segnale aggregato in cui è presente il segnale $s_a[t]$ reso rumoroso (ovvero $x_a[t]$). Questo perché, per quanto detto sulla linearità del sistema e sulla definizione del rumore aggregato, il contributo dato alla cross-correlazione complessiva da parte della cross-correlazione parziale tra $s_a[t]$ e $x_a[t]$ in $m = 0$ sarà più elevato rispetto al contributo dato da tutte le altre cross-correlazioni parziali tra $s_a[t]$ e $x_i[t]$, in quanto le due serie di dati $s_a[t]$ ed $x_a[t]$ sono simili, essendo una la versione rumorosa dell'altra.

In base a questo ragionamento l'attaccante decide tra $X_a[t]$ e $X_b[t]$ valutando quale tra le cross-correlazioni $R_{s_a,X_a}[m]$ e $R_{s_a,X_b}[m]$ presenta un valore in $m = 0$ superiore.

Finora abbiamo solamente impostato il *problema decisionale* e definito il *decisore* che permette all'attaccante di fare la sua scelta, ma non abbiamo considerato i modelli di attacco e difesa illustrati nelle Sezioni 6.1.1 e 6.1.2. Ciò significa che non abbiamo tenuto in considerazione quali operazioni può effettuare l'attaccante su $X_a[t]$ e $X_b[t]$ per rendere più efficiente il metodo

delle cross-correlazioni e nemmeno quali contromisure può prendere un partecipante per evitare che l'attaccante abbia successo nel manipolare $X_a[t]$ e $X_b[t]$. Per valutare la qualità della decisione effettuata dall'attaccante e se l'utilizzo del metodo della cross-correlazione è efficiente o meno consideriamo cinque differenti scenari:

S1. Gli aggregati $X_a[t]$ e $X_b[t]$ non presentano rumore, quindi non è garantita in alcun modo la privacy differenziale;

S2. Gli aggregati $X_a[t]$ e $X_b[t]$ sono rumorosi e il rumore è definito come da teoria, illustrata nel Capitolo 3 e in [12], per garantire la privacy differenziale;

S3. Gli aggregati $X_a[t]$ e $X_b[t]$ sono filtrati entrambi per mezzo del Filtro di Wiener, al fine di ridurne il rumore e abbassare il livello di privacy differenziale, come definito nella Sezione 6.1.1;

S4. Gli aggregati $X_a[t]$ e $X_b[t]$ sono rumorosi e il rumore è correlato, come definito nella Sezione 6.1.2;

S5. Gli aggregati $X_a[t]$ e $X_b[t]$, ottenuti sommando ai singoli contributi un rumore correlato come definito nella Sezione 6.1.2, sono filtrati entrambi per mezzo del Filtro di Wiener, al fine di ridurne il rumore e di abbassare il livello di privacy differenziale.

E' opportuno ricordare che il modello a cui ci riferiamo è illustrato in Figura 6.1. Si suppone, come già detto in precedenza, che l'attaccante conosca il filtro H e le DSP di $n_i[t]$ e $l_i[t]$, ovvero $S_n(f)$ ed $S_l(f)$, necessarie per definire correttamente il Filtro di Wiener, ovvero il filtro ottimo di rimozione del rumore che consente di minimizzare l'errore quadratico medio della stima. Senza queste informazioni l'attaccante potrebbe focalizzarsi, per la rimozione

del rumore, su filtri meno efficienti come ad esempio il *filtro a media mobile* (filtro MA). Ciò che ci interessa per la valutazione delle prestazioni, però, è la situazione ottimale per l'attaccante e peggiore per i partecipanti che vogliono preservare la propria privacy.

In base alla teoria definita finora, risulta evidente che nello scenario S1 l'attaccante non dovrebbe avere problemi nell'effettuare la scelta corretta. Non essendo presente rumore, quando si considera la cross-correlazione tra $s_a[t]$ e $X_a[t]$, si verifica che la cross-correlazione parziale tra $s_a[t]$ e $x_a[t]$, essendo $x_a[t] = s_a[t]$, non è altro che l'autocorrelazione di $s_a[t]$ e che, in base a quanto detto nella Sezione 3.3.3, essa presenta un picco molto pronunciato in $m = 0$. Questo non si verifica se nell'aggregato non è presente $x_a[t] = s_a[t]$.

Nello scenario S2 l'attaccante dovrebbe avere più difficoltà nella scelta a causa dell'introduzione del rumore. Il rumore, infatti, altera il segnale ed ha ripercussioni sui picchi delle cross-correlazioni. Ovviamente l'alterazione dei picchi è maggiore tanto più è potente il rumore sommato al segnale, poiché in questo caso l'alterazione del segnale è più marcata e l'andamento temporale del segnale rumoroso si discosta maggiormente dall'andamento del segnale pulito.

Nello scenario S3 l'attaccante riduce il rumore dai segnali rumorosi aggregati. In questo caso le prestazioni dell'attaccante dovrebbero migliorare rispetto allo scenario S2 in quanto i segnali aggregati rumorosi $X_a[t]$ e $X_b[t]$ vengono parzialmente ripuliti dal rumore e ci si avvicina nuovamente allo scenario S1.

Nello scenario S4 le prestazioni dell'attaccante dovrebbero peggiorare notevolmente: il rumore correlato dovrebbe impedire all'attaccante di effettuare la decisione corretta in quanto altera molto maggiormente i picchi di cross-correlazione rispetto allo scenario S2.

Parametro	Valore
μ_n	700
σ_n	350
$h[t]$	$\frac{1}{9} \cdot \text{tri}[\frac{t}{5}]$
γ	1
δ	0.3
Δ	730
Iterazioni	4000

Tabella 6.1: Parametri di simulazione

Nello scenario S5 le prestazioni dell'attaccante dovrebbero migliorare rispetto allo scenario S4, grazie al filtraggio degli aggregati rumorosi con rumore correlato per mezzo del Filtro di Wiener, ma il filtraggio dovrebbe risultare molto meno efficiente rispetto allo scenario S3. Inoltre, l'attaccante dovrebbe avere più difficoltà nella scelta rispetto allo scenario S2, in quanto il massimo che può ottenere per mezzo del filtraggio è una stima dell'aggregato rumoroso.

6.1.5 Risultati sperimentali

Per verificare il comportamento del decisore in base ai differenti scenari considerati abbiamo scritto un simulatore in Matlab che simula il modello descritto finora.

La simulazione viene iterata per un numero elevato di volte in modo tale da definire in termini percentuali quante volte il decisore utilizzato dall'attaccante ha effettuato la scelta corretta e quante volte invece ha scelto il segnale aggregato rumoroso sbagliato. Questa percentuale è indice della qualità con cui il problema decisionale viene risolto.

ϵ	$\sigma_{l,tot}$
0.01	$1.29 \cdot 10^5$
0.05	$2.62 \cdot 10^4$
0.1	$1.29 \cdot 10^4$
0.15	$8.37 \cdot 10^3$
0.2	$6.68 \cdot 10^3$
0.25	$5.24 \cdot 10^3$
0.3	$4.29 \cdot 10^3$
0.4	$3.20 \cdot 10^3$
0.5	$2.62 \cdot 10^3$
0.6	$2.15 \cdot 10^3$
0.7	$1.85 \cdot 10^3$

Tabella 6.2: Esempio di corrispondenza tra ϵ e $\sigma_{l,tot}$ nel caso $N = 20$, $T = 100$ per gli scenari S2 e S3, ovvero con rumore scorrelato

I risultati sono ottenuti con i parametri definiti nella Tabella 6.1. In Figura 6.4 sono illustrati i risultati ottenuti scegliendo un numero di partecipanti N pari a 20 ed un numero di campioni T per $s_a[t]$, $X_a[t]$ e $X_b[t]$ pari a 100. Abbiamo considerato varianze di rumore complessivo $\sigma^2_{l,tot}$ decrescenti, che determinano un livello di privacy differenziale inferiore. Per il legame che esiste tra la varianza di rumore $\sigma^2_{l,tot}$ e il parametro di sicurezza ϵ si rimanda alla Sezione 3.3.

S2, S3, S4, S5 definiscono lo scenario al quale la curva sul grafico si riferisce. S1 non viene considerato in Figura perché è lo scenario che non prevede privacy differenziale e, non essendoci aggiunta di rumore, ovviamente non dipende dal valore di $\sigma^2_{l,tot}$. In $S1$, infatti, la probabilità di successo per l'attaccante è ottenuta sempre pari al 100%, ovvero il decisore si comporta

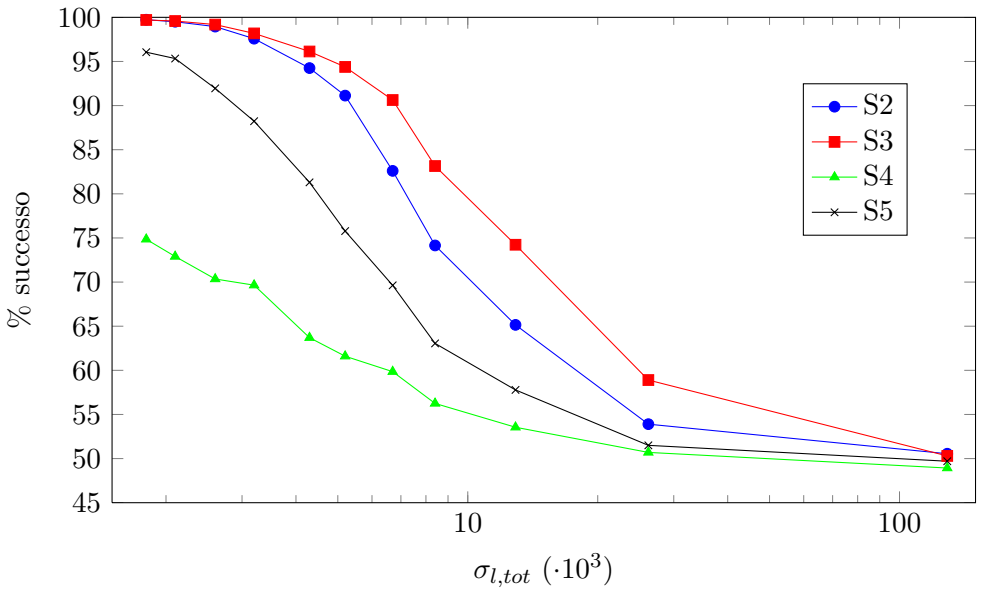

Figura 6.4: Percentuale di successo al variare della deviazione standard del rumore per $N = 20$ e $T = 100$ senza attacco e con attacco nel caso di rumore scorrelato (S2 e S3) e nel caso di rumore correlato (S4 e S5)

come un *decisore ideale*. Nella Tabella 6.2 viene illustrata, per S2 e S3, la corrispondenza tra ϵ e $\sigma_{l,tot}$ (consideriamo la deviazione standard invece della varianza per semplicità di notazione).

Bisogna specificare che negli scenari S4 e S5 si ha a che fare con del rumore filtrato per mezzo del filtro H. In questo caso, per poter paragonare i risultati ottenuti con quelli degli scenari S2 e S3, è necessario impostare una varianza di rumore per il rumore geometrico simmetrico tale per cui, dopo il filtraggio per mezzo di H, si ottiene la stessa varianza di rumore che si ha in tali scenari. Ne risulta un valore di ϵ differente, motivo per cui sul grafico lungo l'asse delle ascisse è riportata la deviazione standard del rumore totale $\sigma_{l,tot}$ (per la caratterizzazione statistica del rumore totale all'aggregato si rimanda alla Sezione 3.3.2). Si ha infatti, per quanto detto nella Sezione

6.1.2, che

$$\sigma_l^2 = \sigma_{l'}^2 \cdot \mathcal{F}^{-1}\left\{|\mathcal{H}(f)|^2\right\}|_{\tau=0}$$

dove $\sigma_{l'}^2$ è la varianza di rumore prima del filtro. In questo modo, impostando correttamente la varianza del rumore geometrico $\sigma_{l'}^2$ prima del filtro, è possibile ottenere la varianza di rumore cercata σ_l^2.

Il parametro di sensitività Δ è stato calcolato seguendo la nuova definizione data nella Sezione 6.1.3.

Si può vedere che per valori di $\sigma_{l,tot}$ elevati, nell'ordine di $130 \cdot 10^3$, il decisore si comporta come un *decisore casuale* in tutti e quattro gli scenari, ovvero non è in grado di risolvere il problema decisionale e la probabilità di successo si attesta intorno al 50%: ciò significa che il criterio di scelta tra $X_a[t]$ e $X_b[t]$ è del tutto casuale.

Al diminuire di $\sigma_{l,tot}$ è visibile uno scostamento tra le probabilità di successo nei quattro differenti scenari. Come ipotizzato, l'attacco per mezzo del Filtro di Wiener (S3) porta miglioramenti rispetto allo scenario in cui l'attacco non viene effettuato (S2). Dai risultati ottenuti si può vedere che per valori di $\sigma_{l,tot}$ intorno a $10 \cdot 10^3$ il Filtro di Wiener permette di ottenere i risultati migliori e la probabilità di successo cresce circa del 10%. Il miglioramento ottenuto in S3 viene drasticamente contrastato in S4, in cui il rumore aggiunto al segnale è filtrato per mezzo del filtro H. Sempre relativamente ai valori di $\sigma_{l,tot}$ intorno a $10 \cdot 10^3$, S4 abbatte la probabilità di successo di circa il 20% rispetto a S3. L'utilizzo del Filtro di Wiener in presenza di rumore correlato (S5) migliora la situazione rispetto a S4 ma le prestazioni si attestano ben al di sotto rispetto a S2 e, ovviamente, S3. Nell'intorno dei valori citati poco sopra le prestazioni rispetto a S2 peggiorano dell'$8 - 9\%$, rispetto a S3 addirittura del $17 - 20\%$. Questo significa che l'aggiunta di rumore filtrato per mezzo del filtro H piuttosto che di rumore geometrico

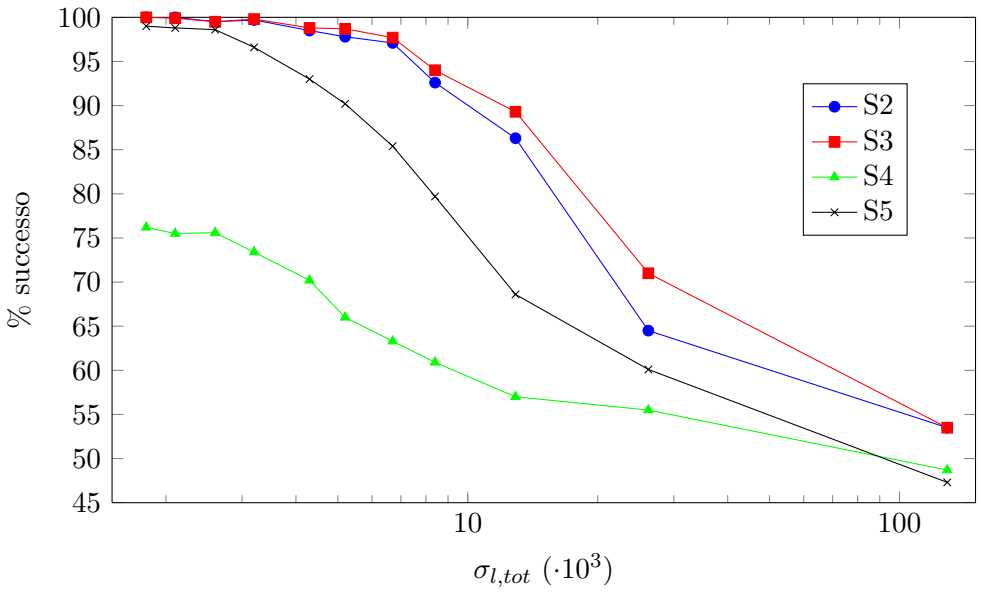

Figura 6.5: Percentuale di successo al variare della deviazione standard del rumore per $N = 50$ e $T = 100$ senza attacco e con attacco nel caso di rumore scorrelato (S2 e S3) e nel caso di rumore correlato (S4 e S5)

simmetrico puro permette di contrastare efficacemente l'attacco per mezzo del Filtro di Wiener.

Se si considerano poi valori per $\sigma_{l,tot}$ sempre minori, ovvero si riduce il livello di privacy differenziale che si vuole mantenere, la probabilità di successo per l'attaccante cresce in tutti gli scenari. Per S2 ed S3, per valori di $\sigma_{l,tot}$ inferiori a $3.2 \cdot 10^3$ l'attaccante ha una probabilità di successo intorno al 100% e il decisore si comporta come un *decisore ideale*. Per questi valori la privacy differenziale non è garantita poiché la varianza di rumore è eccessivamente ridotta ed il rumore è ininfluente. Per S4, invece, la probabilità di successo è notevolmente ridotta (intorno al 70%), ma netti miglioramenti si possono ottenere per mezzo di S5, anche se, per valori di $\sigma_{l,tot}$ inferiori a $3.2 \cdot 10^3$, i risultati sono peggiori del $5 - 10\%$ rispetto ai risultati ottenuti con rumore geometrico puro a parità di varianza di rumore.

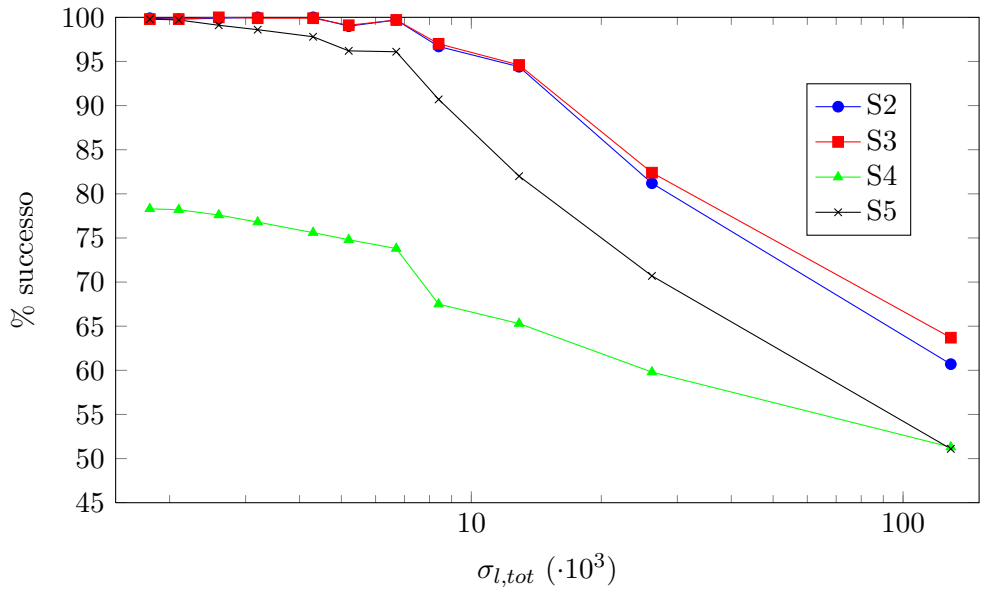

Figura 6.6: Percentuale di successo al variare della deviazione standard del rumore per $N = 100$ e $T = 100$ senza attacco e con attacco nel caso di rumore scorrelato (S2 e S3) e nel caso di rumore correlato (S4 e S5)

Ovviamente se si fosse diminuito ulteriormente il valore di $\sigma_{l,tot}$ si sarebbe ottenuta una convergenza verso la probabilità di successo del 100% in tutti gli scenari.

Abbiamo quindi considerato come variano i risultati mantenendo $T = 100$ ma aumentando il numero di partecipanti a $N = 50$ (Figura 6.5). Tutti gli altri parametri sono illustrati nella Tabella 6.1. Come si può osservare, per valori di $\sigma_{l,tot}$ intorno a $26 \cdot 10^3$ si hanno i risultati migliori per S2 rispetto ad S1, dell'ordine del 7%. Il miglioramento ottenuto è inferiore rispetto al caso con $N = 20$, in cui al meglio si otteneva un miglioramento del 10%. Questo perché l'influenza di $s_a[t]$ all'interno del segnale aggregato rumoroso è inferiore, in base alle proprietà di linearità già discusse all'inizio di questa Sezione. In generale si può notare un avvicinamento tra le curve S2 e S3,

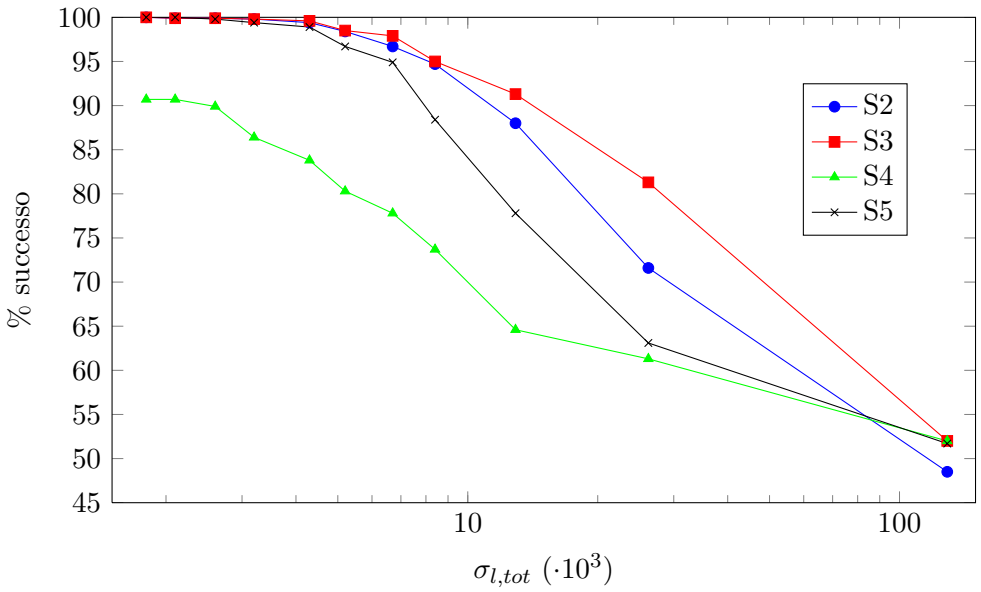

Figura 6.7: Percentuale di successo al variare della deviazione standard del rumore per $N = 50$ e $T = 200$ senza attacco e con attacco nel caso di rumore scorrelato (S2 e S3) e nel caso di rumore correlato (S4 e S5)

che indica come il metodo con filtraggio di Wiener (S3) sia meno efficiente all'aumentare del numero degli utenti a causa della minor efficienza del metodo della cross-correlazione. Negli scenari con rumore correlato S3 e S4 si vede come la situazione peggiora notevolmente, con percentuali di circa il 30% per S4 paragonato a S2 e di circa il 15% per S5 paragonato a S2. Il filtraggio di Wiener nel caso di rumore correlato (S5) migliora la situazione rispetto al caso in cui il filtraggio non viene effettuato (S4) ma la percentuale di successo rimane comunque notevolente inferiore al caso in cui si considera rumore geometrico simmetrico puro (S2 e S3). Aumentando il numero di partecipanti a $N = 100$ (Figura 6.6) quanto detto per $N = 50$ risulta ancora più evidente, con una quasi sovrapposizione tra le curve S1 ed S2 e con peggioramenti significativi se si condiderano gli scenari S4 e S5.

Infine abbiamo considerato, per $N = 50$, un aumento di campioni a $T =$

200 (Figura 6.7). Con un numero di campioni superiore l'attaccante ha a disposizione maggiore informazione e la differenza tra gli scenari S1 ed S2 viene accentuata rispetto al caso con $T = 100$, in quanto il Filtro di Wiener è più efficiente. Si ha, per valori di $\sigma_{l,tot}$ intorno a $26 \cdot 10^3$, un miglioramento di circa il 10%. Per quanto riguarda gli scenari con rumore correlato S4 e S5 si ha, come al solito, un peggioramento consistente per S4 ed un peggioramento ridotto per S5. Si può notare, confrontando i grafici in Figura 6.5 e in Figura 6.7 che all'aumentare del numero di campioni T il decisore inizia a comportarsi come un decisore ideale per valori di $\sigma_{l,tot}$ superiori.

6.2 Modello empirico

Il modello gaussiano illustrato nella Sezione 6.1 è un modello molto utile per comprendere i principi fondamentali della privacy differenziale e di come un attaccante può sfruttare l'informazione da lui conosciuta in un contesto di attaccante *onesto-ma-curioso*. I risultati ottenuti sono soddisfacenti ma risulta chiaro che tale modello è un modello che si discosta parecchio dalla realtà, in quando prevede l'esistenza di un filtro H uguale per tutti i partecipanti che crea l'andamento temporale delle misure di un utente a partire da un processo gaussiano $n_i[t]$ che viene filtrato per mezzo di questo filtro. Questo modello di generazione delle misure degli utenti, che dovrebbero rappresentare quanto un contatore effettivamente registra ed invia al Gateway adibito all'aggregazione, è poco realistico, in quanto non considera l'andamento effettivo nel tempo dei vari dispositivi ed elettrodomestici in ogni singola abitazione.

Per questo motivo abbiamo pensato ad un nuovo modello empirico che

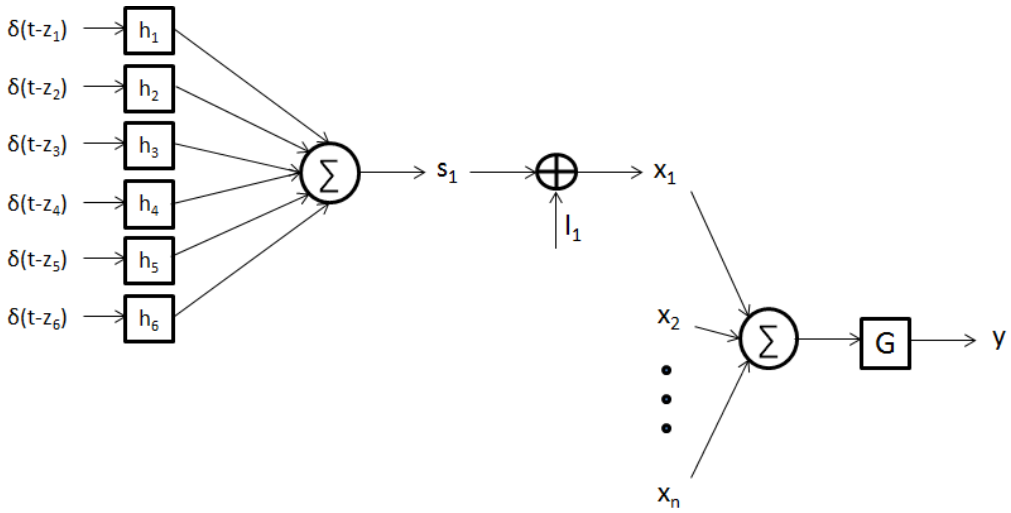

Figura 6.8: Modello empirico

possa generare delle curve di consumo più aderenti alla realtà (Figura 6.9) ed abbiamo riflettuto su quale attacco e quale difesa è possibile effettuare in questo nuovo modello. Per la messa a punto del modello e per lo studio del suo comportamento abbiamo sfruttato le potenzialità di Matlab relativamente all'elaborazione numerica dei segnali.

In generale l'approccio seguito in questa Sezione segue l'approccio seguito nella Sezione 6.1 e si rifà ai concetti espressi nella Sezione 3.3. Non si ha, però, a differenza della Sezione 6.1, una trattazione teorica rigorosa, in quanto il modello è stato elaborato esclusivamente su base empirica.

6.2.1 Modello e attacco

Il modello è illustrato in Figura 6.8. Ipotizziamo la presenza di sei differenti categorie di apparecchiature elettroniche. Per ognuna delle categorie abbiamo considerato i consumi reali nell'arco delle 24 ore, a partire dalle 00.15 fino alle 00.00 del giorno successivo, misurate ad intervalli di 15 minuti: si hanno a disposizione, quindi, $T = 96$ campioni. I consumi sono

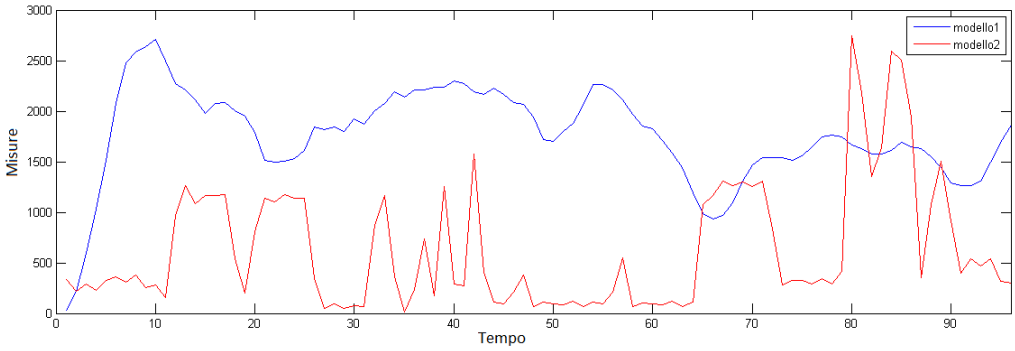

Figura 6.9: Esempio di realizzazione per il singolo utente nei due modelli

stati ricavati da [26], considerando i dati relativi a un'abitazione con potenza contrattuale pari a 3kW (come la maggior parte delle abitazioni su territorio italiano) e configurazione invernale. Ogni curva dei consumi descritta sopra viene considerata come la risposta all'impulso di un filtro $h_j[t]$. Le diverse categorie riguardano le seguenti apparecchiature domestiche:

1. Luci ($h_1[t]$);
2. Forno e forno a microonde ($h_2[t]$);
3. Televisore e Personal Computer ($h_3[t]$);
4. Frigorifero ($h_4[t]$);
5. Boiler ($h_5[t]$);
6. Lavatrice e lavastoviglie ($h_6[t]$).

Le risposte all'impulso $h_j[t]$ rappresentano la base per la realizzazione delle differenti curve di consumo $s_i[t]$ per ognuno degli N utenti. Ognuno degli utenti, infatti, genera la propria realizzazione estraendo un numero naturale $0 \leq z_j \leq 48$ per ognuna delle sei curve e dando in ingresso al filtro $h_j[t]$ l'impulso $\delta(t - z_j)$. Viene quindi effettuata una *convoluzione circolare*

tra l'impulso $\delta(t - z_j)$ e il filtro $h_j[t]$, che in output risulta in una *traslazione circolare* di z_j campioni. Effettuare una traslazione circolare di al massimo 48 campioni significa permettere che le curve di consumo $h_j[t]$ possano portare a cambiamenti nelle abitudini di due utenti differenti di al massimo 12 ore. La curva di consumo complessiva del singolo utente viene allora generata sommando gli output di ogni singolo filtro. Il segnale del generico utente $s_i[t]$ risulta quindi

$$s_i[t] = \sum_{j=1}^{6} (\delta[t - z_j] \circledast h_j[t])$$

Una volta ottenuti i processi $s_i[t]$ per ogni singolo utente si procede con l'aggiunta di rumore geometrico simmetrico $l_i[t]$, come descritto nella Sezione 3.3 e nella Sezione 6.1, ottenendo $x_i[t] = s_i[t] + l_i[t]$. Si procede quindi con l'aggregazione calcolando $x_{aggr}[t] = \sum_{i=1}^{N} x_i[t]$ e si forniscono le misure aggregate rumorose all'attaccante.

L'attaccante, dopo aver collezionato le $T = 96$ misure aggregate, cerca di rimuovere il rumore facendo passare i T campioni del processo $x_{aggr}[t]$ nel filtro G, ottenendo il processo $y[t]$. A differenza del modello illustrato nella Sezione 6.1 non è possibile per l'attaccante costruire un filtro ottimo come il Filtro di Wiener. Un modello del genere, infatti, rende differente la generazione della curva di consumo per ogni singolo utente. Ciò significa che non esiste un unico filtro H che genera i consumi per ogni utente a partire da un determinato processo gaussiano.

L'attaccante deve allora scegliere un filtro G di rimozione del rumore non ottimo. La scelta più semplice ricade sul *filtro a media mobile* (filtro MA), che permette di ottenere

$$y[t] = \frac{1}{M} \sum_{j=0}^{M-1} x_{aggr}[t + j]$$

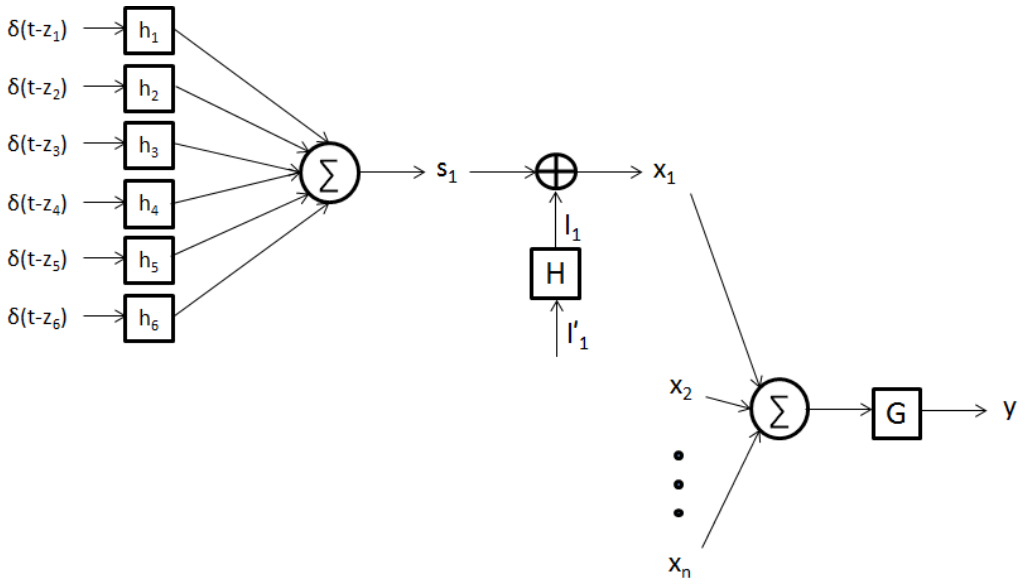

Figura 6.10: Difesa per il modello empirico con rumore prefiltrato per mezzo del filtro H

Il parametro M è la finestra di campioni su cui viene effettuata la media: in questo modo si ha rimozione del rumore in quanto vengono ridotte le fluttuazioni sul segnale dovute alla presenza del rumore geometrico simmetrico. L'utilizzo del filtro G così definito permette quindi di ottenere, grazie alle proprietà di linearità del sistema descritte nella Sezione 6.1 e valide anche in questo contesto, una stima del segnale aggregato $\sum_{i=1}^{N} s_i[t]$, con conseguente riduzione del livello di privacy differenziale.

6.2.2 Difesa dall'attacco

Dopo aver ipotizzato il possibile attacco per effettuare una riduzione del rumore sulle misure aggregate per ridurre il livello di privacy differenziale, ci siamo soffermati su una possibile difesa a tale attacco. Abbiamo mantenuto l'idea evidenziata nella Sezione 6.1.2 di sommare del rumore correlato, dando in input ad un filtro H il rumore simmetrico geometrico e sommando l'out-

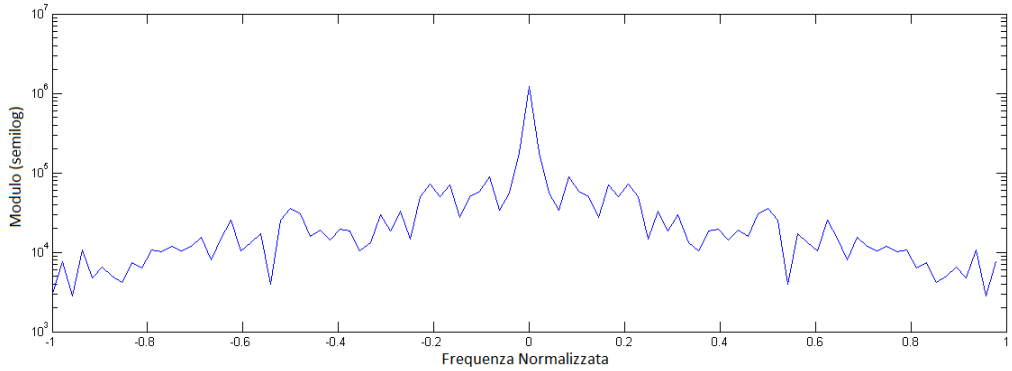

Figura 6.11: Modulo della Trasformata di Fourier di una realizzazione del segnale aggregato non rumoroso

put $l_i[t]$ al processo $s_i[t]$. In questo modello, però, il filtro H per il filtraggio del rumore deve essere in qualche modo stimato. Infatti le risposte impulsive $h_i[t]$, traslate circolarmente in modo differente per ogni singolo utente e sommate, non permettono di definire un filtro univoco per la generazione delle misure. Per definire quale può essere un filtro che contrasti in modo efficiente l'attacco per mezzo del filtro G a media mobile abbiamo analizzato la Trasformata di Fourier di alcune realizzazioni del processo $\sum_{i=1}^{N} s_i[t]$, ovvero l'aggregato delle misure non rumorose, notando un andamento decrescente in frequenza (Figura 6.11). Il nostro obiettivo è di colorare il rumore geometrico simmetrico in modo tale da conferirgli un andamento temporale e una caratterizzazione in frequenza simile a quella del processo aggregato non rumoroso. A questo fine abbiamo effettuato una *stima spettrale parametrica autoregressiva (AR) a singolo polo* [27] del segnale $\sum_{i=1}^{N} s_i[t]$. Il singolo polo stimato è risultato essere un polo reale positivo di valore all'incirca pari a 0.95. In termini di *trasformata zeta*, quindi, abbiamo considerato un filtro H così fatto

$$H(z) = \frac{1}{1 - \eta \cdot z^{-1}} \quad (\eta = 0.95)$$

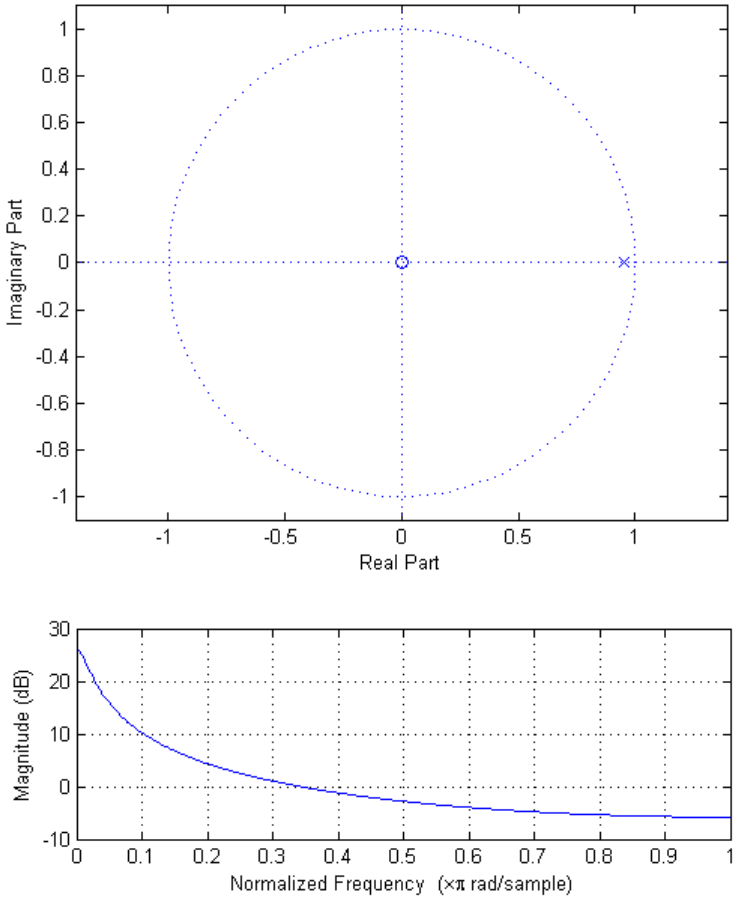

Figura 6.12: Posizione del polo stimato sul piano di Gauss e modulo della risposta in frequenza

Tale filtro presenta un andamento in frequenza esponenziale decrescente (Figura 6.12) che approssima l'andamento in frequenza di $\sum_{i=1}^{N} s_i[t]$. Dare in input il rumore geometrico simmetrico, il quale presenta campioni tutti scorrelati tra loro e, per questo motivo, Trasformata di Fourier costante in frequenza, al filtro H, significa sagomare l'andamento in frequenza del rumore in modo simile all'andamento in frequenza dell'aggregato non rumoroso ed

Parametro	Valore
T	100 campioni
γ	1
δ	0.3
Δ	3000
$g[t]$	$\frac{1}{5} \cdot \text{rect}[\frac{t}{5}]$
Iterazioni	4000

Tabella 6.3: Parametri di simulazione

introdurre una correlazione tra i suoi campioni.

Grazie a questa idea il filtro G MA, che ha un comportamento passa-basso ed ha un buon funzionamento in caso di rumore con componenti ad alta frenquenza, viene reso inefficiente in quanto le componenti ad alta frequenza del rumore sono state eliminate dai partecipanti effettuando il filtraggio di tale rumore per mezzo del filtro H.

6.2.3 Risultati sperimentali

Il Problema Decisionale formalizzato per valutare le prestazioni di attacco e difesa è il medesimo definito nella Sezione 6.1.4, come il metodo utilizzato per effettuare la scelta, ovvero il metodo della cross-correlazione. All'attaccante vengono forniti gli aggregati rumorosi $X_a[t]$ e $X_b[t]$, che differiscono solamente per la realizzazione di un singolo utente (a nel primo aggregato e b nel secondo aggregato), e la realizzazione singola non rumorosa $s_a[t]$ dell'utente a. Viene valutata, su un numero elevato di iterazioni, la percentuale di successo per l'attaccante che aumenta all'aumentare della qualità del decisore. La qualità della decisione viene valutata per cinque differenti

scenari:

S1. Gli aggregati $X_a[t]$ e $X_b[t]$ non presentano rumore, quindi non è garantita in alcun modo la privacy differenziale;

S2. Gli aggregati $X_a[t]$ e $X_b[t]$ sono rumorosi e il rumore è definito come da teoria, illustrata nel Capitolo 3 e in [12], per garantire la privacy differenziale;

S3. Gli aggregati $X_a[t]$ e $X_b[t]$ sono filtrati entrambi per mezzo del filtro G a media mobile, al fine di ridurne il rumore e abbassare il livello di privacy differenziale, come definito nella Sezione 6.2.1;

S4. Gli aggregati $X_a[t]$ e $X_b[t]$ sono rumorosi e il rumore è correlato per mezzo del filtro H, come definito nella Sezione 6.2.2;

S5. Gli aggregati $X_a[t]$ e $X_b[t]$, ottenuti sommando ai singoli contributi un rumore correlato per mezzo del filtro H come definito nella Sezione 6.2.2, sono filtrati entrambi per mezzo del filtro G a media mobile, al fine di ridurne il rumore e di abbassare il livello di privacy differenziale.

In questo modello, in quanto non si ha a che fare con processi casuali gaussiani, non è prossibile utilizzare per il parametro di sensitività Δ la nuova definizione da noi ideata nella Sezione 6.1.3. E' necessario quindi rifarsi alla definizione che è stata data in [11] e riportata nella Sezione 3.3.1. Poiché, come definito nella Sezione 6.2.1, le curve di consumo considerate per le apparecchiature elettroniche si riferiscono ad un'abitazione con fornitura 3kW, consideriamo $\Delta = 3000$.

I risultati sono ottenuti con i parametri definiti nella Tabella 6.3. In Figura 6.13 sono illustrati i risultati ottenuti scegliendo N pari a 20. Anche in questo modello lo scenario S1, in cui non è presente rumore, porta a

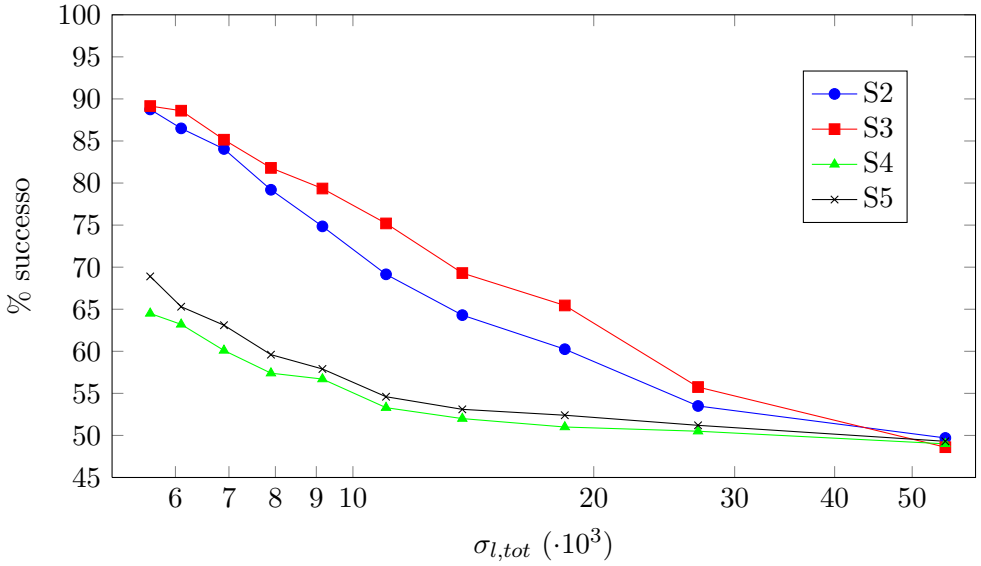

Figura 6.13: Percentuale di successo al variare della deviazione standard del rumore per $N = 20$ senza attacco e con attacco nel caso di rumore scorrelato (S2 e S3) e nel caso di rumore correlato (S4 e S5)

ϵ	$\sigma_{l,tot}$
0.1	$55 \cdot 10^3$
0.2	$27 \cdot 10^3$
0.3	$18.4 \cdot 10^3$
0.4	$13.7 \cdot 10^3$
0.5	$11 \cdot 10^3$
0.6	$9.16 \cdot 10^3$
0.7	$7.90 \cdot 10^3$
0.8	$6.90 \cdot 10^3$
0.9	$6.10 \cdot 10^3$
1	$5.58 \cdot 10^3$

Tabella 6.4: Esempio di corrispondenza tra ϵ e $\sigma_{l,tot}$ nel caso $N = 20$ per gli scenari S2 e S3, ovvero con rumore scorrelato

percentuali di successo del 100%. Il decisore è sempre in grado di risolvere il Problema Decisionale e si comporta da *decisore ideale*. S2, S3, S4, S5 definiscono lo scenario al quale la curva sul grafico si riferisce. La relazione tra il parametro di sicurezza ϵ e la deviazione standard $\sigma_{l,tot}$ per S2 e S3, come definita nella Sezione 3.3, è illustrata in Tabella 6.4. Dalla Figura si vede che, per S2 e S3 e per deviazioni standard di rumore $\sigma_{l,tot}$ tra $18.4 \cdot 10^3$ e $9.16 \cdot 10^3$ il miglioramento portato dal filtro MA è di circa il 5%, mentre nel modello precedente il Filtro di Wiener portava ad un miglioramento massimo del 10%. Da questo si può notare come il filtro MA sia meno efficiente, a parità di parametro di sicurezza ϵ, rispetto al Filtro di Wiener. Il comportamento complessivo del sistema è quello che ci aspettavamo, con il decisore che si comporta da *decisore casuale* per deviazioni standard di rumore $\sigma_{l,tot}$ molto elevate e all'incirca da *decisore ideale* per deviazioni standard di rumore $\sigma_{l,tot}$ molto ridotte. Considerando poi S4 e S5, in cui è stato colorato il rumore geometrico simmetrico per mezzo del filtro H, si nota un netto peggioramento delle prestazioni del decisore. La differenza tra S4 e S5 è piuttosto ridotta, nell'ordine di qualche punto percentuale: anche filtrando con il filtro MA le misure aggregate rumorose con rumore correlato (S5) le prestazioni del decisore non migliorano di molto rispetto al caso in cui non si effettua questa operazione (S4). Anche in questo caso per S4 e S5 si effettua il paragone con S2 e S3 a parità di deviazione standard di rumore $\sigma_{l,tot}$. Rispetto a S2 e S3 il peggioramento in S4 e S5 è consistente, anche nell'ordine del $20 - 25\%$. Anche in questo caso la difesa ideata per contrastare l'attacco con filtro MA è efficace.

Abbiamo quindi considerato $N = 50$. Con l'aumentare del numero di utenti risulta che S2 porta un miglioramento di circa il 5% rispetto a S1 come nel caso con $N = 20$, ma questo miglioramento si verifica per un intervallo

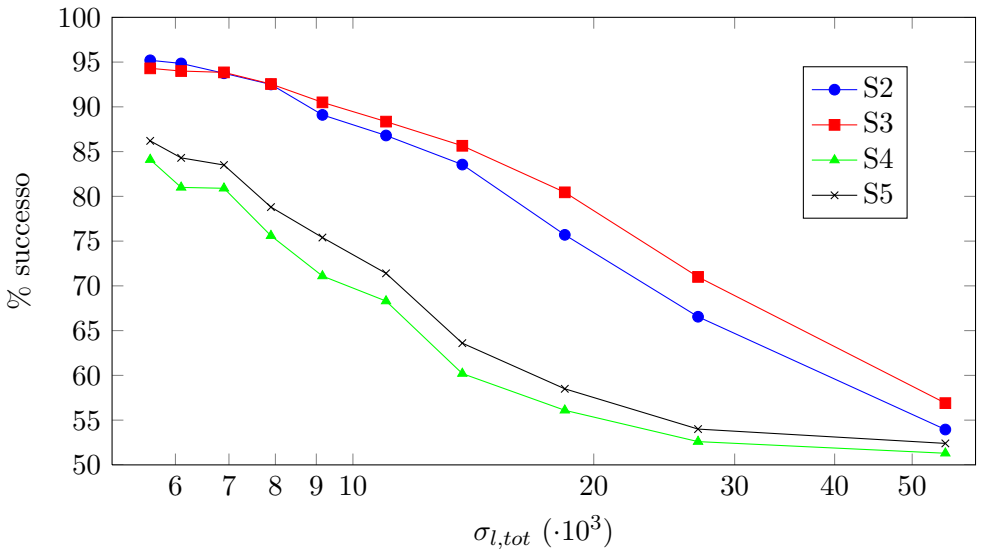

Figura 6.14: Percentuale di successo al variare della deviazione standard di rumore per $N = 50$ senza attacco e con attacco nel caso di rumore scorrelato (S2 e S3) e nel caso di rumore correlato (S4 e S5)

di $\sigma_{l,tot}$ inferiore, tra $27 \cdot 10^3$ e $18.4 \cdot 10^3$. In generale le curve S2 e S3 tendono ad avvicinarsi. S4 e S5, poi, peggiorano anche in questo caso notevolmente le prestazioni del decisore, dell'ordine del $20 - 25\%$, ma tra di esse non si ha una differenza significativa, nell'ordine del $2 - 3\%$. Anche in questo caso, allora, il filtraggio MA delle misure aggregate con rumore correlato non porta a miglioramenti significativi.

In ultima analisi abbiamo considerato il caso $N = 100$. Non abbiamo riportato il grafico in quanto le curve di S2 e S3 risultano esattamente sovrapposte. Ciò significa che il filtraggio MA non porta alcun miglioramento e un attacco di questo genere risulta inefficiente in partenza. Chiaramente risulta comunque conveniente filtrare il rumore geometrico simmetrico (S4) in quanto si abbassa ulteriormente la percentuale di successo del decisore e si contrasta ulteriormente la capacità dell'attaccante di effettuare la scelta corretta.

CAPITOLO 7

CONCLUSIONI

In questo lavoro abbiamo trattato due aspetti di privacy differenti ma strettamente connessi nel contesto delle Smart Grid.

In primo luogo abbiamo illustrato due possibili architetture per l'aggregazione spaziale e/o temporale delle misure di consumo degli utenti al fine di garantire che la curva di consumo di un singolo utente non sia resa disponibile all'entità che ha richiesto i dati aggregati. Sulla base di queste architetture abbiamo realizzato due emulatori per poter valutarne le prestazioni. Ne è risultato che la seconda architettura considerata, ovvero l'architettura distribuita, è più scalabile e più semplice da implementare in uno scenario realistico: questo va a discapito di una maggiore complessità del protocollo di comunicazione tra i nodi dell'architettura. Abbiamo realizzato gli emulatori con lo scopo di creare un prototipo funzionante e facilmente implementabile su una reale rete di contatori, anche su larga scala. L'unico requisito richiesto per contatori e nodi della rete è che siano dotati del framework Twisted.

Questo comporta che, per ottenere le misure volute, abbiamo sfruttato solo in parte le potenzialità del software da noi scritto, poiché, ovviamente, non ci è stato possibile distribuirlo su un numero elevato di macchine. I risultati ottenuti ci hanno comunque permesso di mostrare in generale un miglior comportamento dell'architettura distribuita, soprattutto in termini di onere computazionale dei nodi destinati all'aggregazione di share.

Durante questa fase, in cui ci siamo essenzialmente concentrati sull'implementazione di architetture e protocolli già esistenti in letteratura, è emersa una problematica mai trattata prima, ovvero l'esistenza in rete di nodi malevoli in grado di alterare le share aggregate. Per questo motivo abbiamo introdotto un nuovo metodo di recupero robusto delle misure aggregate, che sfrutta l'algoritmo di Berlekamp-Welch. Supponendo intervalli di aggregazione nell'ordine dei minuti, abbiamo mostrato che il tempo computazionale per il recupero delle misure aggregate per mezzo di questo algoritmo è compatibile con essi.

Successivamente abbiamo focalizzato la nostra attenzione su un ulteriore aspetto: la privacy differenziale. Negli ultimi anni, la ricerca scientifica su tale argomento si è soffermata soprattutto sull'ideazione di differenti tecniche che consentano di garantire privacy differenziale all'utente: da questo punto di vista si sono avuti grandi sviluppi. Mancava però, in letteratura, la trattazione di possibili attacchi volti a ridurre la privacy differenziale degli utenti e possibili contromisure: questo lavoro cerca di sopperire, almeno in parte, a questa mancanza. A tal fine abbiamo elaborato due modelli che consentono di definire due possibili attacchi alla privacy differenziale e due possibili contromisure: abbiamo fatto ampio riferimento ad alcuni concetti chiave della teoria e dell'elaborazione numerica dei segnali. I risultati ottenuti confermano come, facendo ipotesi sulle statistiche dei segnali che rappresentano

l'andamento delle misure, sia effetivamente possibile eseguire un attacco per la riduzione del rumore e quindi della privacy differenziale. I risultati mostrano inoltre l'efficienza delle nostre contromisure che permettono di ottenere livelli di privacy differenziale maggiore a parità di potenza di rumore.

In conclusione, questo lavoro ha trattato aspetti sia implementativi che teorici in relazione alla necessità di privacy degli utenti in un contesto di Smart Grid, evidenziando problematiche e possibili soluzioni osservate da differenti prospettive. Future ricerche potrebbero prendere spunto da questo lavoro di tesi, soprattutto per il contributo offerto al concetto di privacy differenziale. Ulteriori ricerche potrebbero, ad esempio, soffermarsi sull'elaborazione di nuovi modelli più realistici o di nuove tipologie di attacco più efficienti.

RINGRAZIAMENTI

Un sincero ringraziamento va a tutti coloro che, in momenti diversi e in vari modi, ci hanno aiutato nella stesura di questo lavoro.

Prima di tutto, un *immenso* grazie ai nostri genitori che ci hanno sostenuto sotto ogni punto di vista e ci hanno permesso di raggiungere questo importante traguardo.

Vogliamo poi ringraziare il nostro relatore, prof. Giacomo Verticale, e la nostra correlatrice Cristina per averci guidato ed aiutato tutte le volte in cui ne abbiamo avuto bisogno.

Grazie alle nostre sorelle, Daniela e Lisa, che ci hanno sopportato anche nei momenti più difficili.

Grazie a tutti i nostri compagni di corso, che hanno reso piacevoli e, per certi versi, divertenti questi cinque anni insieme.

Grazie ai nostri amici, che continueranno a volerci bene nonostante il nostro essere ingegneri!

E, infine, uno speciale ringraziamento a due importantissime donne della nostra vita: Annina e Marta.

BIBLIOGRAFIA

[1] Smart grid.
URL http://www.smartgrid.gov/the_smart_grid

[2] C. Rottondi, G. Verticale, A. Capone, Privacy-preserving smart metering with multiple data consumers, 2012, submitted.

[3] C. Rottondi, G. Verticale, C. Krauss, Distributed privacy-preserving aggregation of metering data in smart grids, 2012, submitted.

[4] F. Li, B. Luo, P. Liu, Secure information aggregation for smart grids using homomorphic encryption, in: Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on, 2010, pp. 327 –332. doi:10.1109/SMARTGRID.2010.5622064.

[5] F. D. Garcia, B. Jacobs, Privacy-friendly energy-metering via homomorphic encryption, in: Proceedings of the 6th international conference on Security and trust management, STM'10, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 226–238.
URL http://dl.acm.org/citation.cfm?id=2050149.2050164

[6] K. Kursawe, G. Danezis, M. Kohlweiss, Privacy-friendly aggregation for the smart-grid, in: Proceedings of the 11th international conference on Privacy enhancing technologies, PETS'11, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 175–191.
URL http://dl.acm.org/citation.cfm?id=2032162.2032172

[7] G. Ács, C. Castelluccia, I have a dream!: differentially private smart metering, in: Proceedings of the 13th international conference on Information hiding, IH'11, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 118–132.
URL http://dl.acm.org/citation.cfm?id=2042445.2042457

[8] M. Burkhart, M. Strasser, D. Many, X. Dimitropoulos, Sepia: Privacy-preserving aggregation of multi-domain network events and statistics, in: USENIX SECURITY SYMPOSIUM, USENIX, 2010.

[9] A. Shamir, How to share a secret, Commun. ACM 22 (11) (1979) 612–613. doi:10.1145/359168.359176.
URL http://doi.acm.org/10.1145/359168.359176

[10] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. Kaashoek, F. Dabek, H. Balakrishnan, Chord: a scalable peer-to-peer lookup protocol for internet applications, Networking, IEEE/ACM Transactions on 11 (1) (2003) 17 – 32. doi:10.1109/TNET.2002.808407.

[11] C. Dwork, Differential privacy, in: M. Bugliesi, B. Preneel, V. Sassone, I. Wegener (Eds.), Automata, Languages and Programming, Vol. 4052 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2006, pp. 1–12. doi:10.1007/11787006_1.
URL http://dx.doi.org/10.1007/11787006_1

[12] E. Shi, T.-H. H. Chan, E. G. Rieffel, R. Chow, D. Song, Privacy-preserving aggregation of time-series data., in: NDSS, The Internet Society, 2011.
URL `http://dblp.uni-trier.de/db/conf/ndss/ndss2011.html#ShiCRCS11`

[13] S. Rajagopalan, L. Sankar, S. Mohajer, H. Poor, Smart meter privacy: A utility-privacy framework, in: Smart Grid Communications (SmartGridComm), 2011 IEEE International Conference on, 2011, pp. 190 –195. doi:10.1109/SmartGridComm.2011.6102315.

[14] F. Zhang, L. He, W. He, X. Liu, Data perturbation with state-dependent noise for participatory sensing., in: A. G. Greenberg, K. Sohraby (Eds.), INFOCOM, IEEE, pp. 2246–2254.

[15] A. Pagnoni, G. Verticale, E. Munarini, Crittografia: con elementi di teoria dei codici, Pearson Italia, 2009, traduzione italiana di W. Trappe, L. Washington, Introduction to cryptography with coding theory, 2nd Edition, Prentice Hall, 2006.
URL `http://hdl.handle.net/2434/73125`

[16] N. Smart, Cryptography: an introduction, Mcgraw-hill education, McGraw-Hill, 2003.
URL `http://books.google.it/books?id=keNGAQAAIAAJ`

[17] I. Mironov, O. Pandey, O. Reingold, S. Vadhan, Computational differential privacy, in: Proceedings of the 29th Annual International Cryptology Conference on Advances in Cryptology, CRYPTO '09, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 126–142. doi:10.1007/978-3-642-

03356-8_8.
URL http://dx.doi.org/10.1007/978-3-642-03356-8_8

[18] C. Prati, Segnali e sistemi per le telecomunicazioni, McGraw-Hill Companies, 2010.

[19] Python.
URL http://www.python.org/

[20] Twisted.
URL http://twistedmatrix.com/

[21] Sage.
URL http://www.sagemath.org/

[22] D. H. Crocker, Standard for the format of ARPA Internet text messages, IETF, 1982.
URL http://www.ietf.org/rfc/rfc0822.txt

[23] P. Garcia Lopez, Implementazione locale di Chord in Python.
URL http://deim.urv.cat/~pgarcia/P2P/soft/chordfinal.zip

[24] Openssl.
URL http://www.openssl.org/

[25] R. C. Gonzalez, R. E. Woods, Digital Image Processing (3rd Edition), Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.

[26] Micene Project, http://www.eerg.it/index.php?p=Progetti_-_MICENE (apr 2012).

[27] F. Rocca, Elaborazione numerica dei segnali, 2010.
URL http://risorse.dei.polimi.it/dsp/courses/ens_l1/