Protocolli di sicurezza per Reti di sensori: panoramica, proprietà, attacchi e implementazione di LEAP

Sensori

Cosa è un sensore?

Un sensore è un dispositivo in grado di rilevare una grandezza interagendo con essa: l'energia ricevuta dal sensore modifica lo stato della grandezza variando una delle sue proprietà(lunghezza, resistenza elettrica, etc…). Il termine sensore è spesso utilizzato in luogo di trasduttore, il quale è invece più precisamente un dispositivo fisico progettato per trasformare grandezze appartenenti a un sistema energetico in grandezze equivalenti appartenenti a un diverso sistema energetico.
Poiché all'interazione con una grandezza è associata in genere anche l'operazione di conversione, si può definire sensore un dispositivo che acquisisce in ingresso una certa grandezza e ne fornisce in uscita una di natura differente, ma legata alla prima da una legge nota. La grandezza in uscita dal sensore, una volta opportunamente amplificata ed elaborata, è inviata a uno strumento di misura detto strumento terminale. Generalmente, quando un sensore fornisce una grandezza analogica, prima che venga usata in elaborazione, viene convertita in grandezza digitale di più facile trattamento.

Come vengono impiegati i sensori?

L'interesse per i sensori è in continua crescita, data l'esigenza di misurare grandezze fisiche, chimiche e biologiche nell'ambito di numerose aree applicative: monitoraggio di apparecchiature, processi industriali, analisi e diagnostica medica, robotica, controlli ambientali, costruzioni civili.

Sono moltissimi infatti i sistemi le cui prestazioni sono migliorate grazie all'impiego di sensori. Le automobili, per esempio, un tempo quasi del tutto prive di sensori, hanno ormai al loro interno vari tipi di dispositivi in grado di verificare il corretto funzionamento di parti e sottosistemi. Tutto ciò ne aumenta la sicurezza, la capacità di prevenzione dei guasti e il controllo delle condizioni di inquinamento ambientale. Sono stati inseriti, infatti, sensori di tipo fisico in ausilio all'impianto elettrico e agli apparati meccanici ma anche di tipo chimico per controllare in tempo reale le condizioni di funzionamento

dell'apparato motore e le caratteristiche dei gas di scarico. Il flusso di informazioni che ne deriva deve essere spesso analizzato in tempo reale, così da modificare in modo opportuno le condizioni di funzionamento e i parametri critici, in presenza di eventuali problemi. Si ricorre pertanto a particolari sistemi automatici in grado di acquisire, memorizzare, elaborare tali informazioni (sistemi di acquisizione dei dati).

I sensori hanno pertanto diverse applicazioni nei campi più disparati , in relazione al tipo di grandezza da essi misurata.

Classificazione dei sensori

I sensori possono essere classificati sulla base del tipo di grandezza rilevata: meccanica (posizione, velocità, accelerazione, pressione, forza, deformazione, massa, densità, momento, vibrazioni ecc.); termica (temperatura, calore specifico, conducibilità termica ecc.); acustica (livello sonoro ecc.); elettrica (carica, intensità di corrente, tensione, resistenza elettrica ecc.); magnetica (intensità di campo, permeabilità ecc.); ottica; elettromagnetica.

Per quanto riguarda le grandezze elettriche e magnetiche si parla più propriamente di sonde, mentre per le grandezze ottiche ed elettromagnetiche piuttosto che all'ampiezza o alla frequenza delle onde si fa riferimento in genere all'intensità (si ricorre, a tale scopo, a celle fotoelettriche e a celle fotoconduttive, sensibili a radiazioniluminose, infrarosse e così via). Riguardo alle grandezze acustiche si utilizzano, tenuto conto della natura del suono, particolari sensori di vibrazioni. In definitiva con sensori di grandezze fisiche ci si riferisce soprattutto a sensori di grandezze meccaniche e termiche.

Reti di sensori

Cosa è una rete di sensori?

Una rete di sensori è una rete formata da piccoli dispositivi (Smart Sensor), perfettamente autonomi che cooperano nell'esecuzione di una qualche applicazione di raccolta informazioni da un determinato fenomeno fisico. L'obbiettivo principale di una rete di sensori è quello di raccogliere dati e informazioni dall'ambiente circostante.

L'infrastruttura di rete può essere di tipo cablato oppure senza fili , quest'ultima è quella più utilizzata nella realtà tecnologica odierna , in quanto permette una migliore distribuzione dei nodi sul territorio. La necessità dell'implementazione di un'infrastruttura di rete al contempo richiede però l'utilizzo di sensori più evoluti che non sono più dei semplici trasduttori di grandezze fisiche, ma sistemi più complessi che integrano oltre alle capacità di misura anche capacità di memorizzazione, di calcolo ed ovviamente interfacce di comunicazione. Queste osservazioni portano alla definizione degli "Smart Sensor", dispositivi integrati che sono dotati di microcontrollori in grado di effettuare attività di comunicazione ed elaborazione dell'informazione.

Impiego delle reti di sensori

Le reti di sensori rappresentano la naturale, ma al contempo rivoluzionaria, evoluzione dell'impiego di sensori nell'ambito industriale. Il mercato, infatti, richiede dispositivi ed impianti dotati sempre di maggiori capacità ed elevati livelli di funzionalità, i sensori utilizzati all'interno di questi dispositivi e sistemi vengono in genere impiegati per stimare una

grandezza fisica o utilizzati per monitorare parametri di "controllo di processo". L'utilizzo di una rete di trasduttori porta innegabili vantaggi rispetto all'utilizzo di sensori tradizionali in termini di flessibilità, performance, facilità d'installazione, costi di eventuali sviluppi futuri ed attività di manutenzione.

I campi applicativi delle reti di sensori sono quindi numerosi e in continua evoluzione.

Alcuni esempi di utilizzo di reti di sensori sono:

- Monitoraggio di ambienti industriali: come strumentazioni, livello di inquinamento, allarme incendi, integrità strutturale degli edifici.

- Monitoraggio e controllo del clima.

- Gestione del consumo energetico , distribuzione dell'energia.

- Ambiti sanitari, per monitorare e assistere pazienti disabili e non.

- Ambiti commerciali: gestione degli inventari, monitoraggio della qualità dei prodotti.

- Ambito militare: sorveglianza dei campi di battaglia.

- Monitoraggio di aree colpite da disastri.

- Domotica.

- Monitoraggio ambientale.

- Controllo all'interno di veicoli.

- Geolocalizzazione e tracking.

- HCI (riconoscimento dei gesti, tracking).

Caratteristiche delle reti di sensori

Le reti di sensori hanno diverse caratteristiche che dipendono dalla topologia, e dal tipo di comunicazione utilizzato.

La caratteristica fondamentale è che tutti i nodi della rete

devono cooperare computazionalmente tra di loro per la realizzazione di un'infrastruttura sicura e affidabile.

Per la comunicazione, i nodi sensori utilizzano diversi paradigmi , come ad esempio:

- Many to One , più nodi sensori inviano i dati ad una stazione base o ad un punto di aggregazione nella rete.

- One to Many , un singolo nodo comunica via multicast una query o un'informazione di controllo a parecchi nodi sensori.

- Local Communication , per scoprirsi e coordinarsi l'uno con l'altro, i nodi vicini si scambiano dei messaggi. Un nodo può spedire i messaggi via broadcast a tutti i nodi vicini o ad un singolo nodo vicino in unicast.

Design di una rete di sensori

Nella progettazione e nella realizzazione di una rete di sensori , bisogna tener conto di diversi fattori , che in un modo o nell'altro, finiscono per influenzare in maniera più o meno forte l'implementazione della rete stessa.

Bisogna sempre tener conto che la rete di sensori deve possedere una certa tolleranza agli errori, cioè deve essere in grado di far fronte a fallimenti dovuti ad esempio a mancanza di energia, a danni fisici o interferenze ambientali. Altri elementi non trascurabili in fase di progettazione, sono la Scalabilità, cioè la densità e il numero di sensori in una rete, i costi di produzione e l'ambiente in cui la rete verrà implementata, solitamente aree impervie e remote.

Elementi cruciali in una rete di sensori sono : La topologia della rete, che attraversa diverse fasi variabili nelle quali i sensori vengono eliminati , sostituiti o aggiunti in base alle problematiche che potrebbero occorrere durante il periodo di vita della rete stessa; Il mezzo di trasmissione utilizzato dai singoli nodi ; il consumo energetico , in quanto i nodi sono dotati di energia limitata , dipendendo completamente dalla batteria che essi hanno in dotazione.

Sicurezza nelle reti di sensori Wireless (WSN)

Le più comuni reti di sensori , utilizzano tecnologie wireless per la comunicazione tra di essi e con internet , in quanto una rete cablata risulterebbe difficoltosa da realizzare e mantenere in zone remote e difficilmente raggiungibili. Per questo passeremo direttamente all'analisi delle reti di sensori wireless .

Le reti di sensori possono essere soggette ad attività malevole, la sicurezza è quindi un requisito da cui non si può prescindere. I servizi di sicurezza devono essere efficienti dal punto di vista energetico , scalabili e forti , devono inoltre garantire confidenzialità, integrità e autenticazione sia dei dati rilevati dal sensore che delle informazioni di routing . A questo punto è possibile fare una distinzione di possibili attacchi , minacce e attaccanti:

- Attacchi: cattura del nodo, manomissione fisica del nodo, spoofing, sniffing, DoS.

- Minacce:

  Attive: masquerade, replay, modifica dei messaggi, Dos.

  Passive: analisi del traffico, rilascio del contenuto del messaggio.

  Interne: nodi interni compromessi che eseguono codice malevolo.

  Esterne: L'attaccante non ha accesso speciale allarete di sensori.

- Attaccanti:

  Mote-class attackers: l'attaccante ha accesso a pochi nodi sensori con capacità simili agli altri nodi.

  Laptop-class attackers: l'attaccante può avere accesso a dispositivi più potenti, come laptop o simili. Un attaccante in questa classe ha più possibilità di attacco di un semplice

nodo.

La sfida fondamentale alla sicurezza nelle reti di sensori sta nell'impossibilità di applicare i più recenti algoritmi sicuri alle comunicazioni tra i vari nodi a causa della limitatezza computazionale , di memoria ed energetica di ogni Smart sensor. L'applicazione degli schemi di cifratura richiede la trasmissione di bit extra e quindi l'aumento di payload dei pacchetti scambiati . Ciò richiede elaborazioni extra, memoria aggiuntiva e consumo di batteria extra. Ad esempio se un nodo utilizzasse il protocollo RSA potrebbe esaurire tutta la batteria a disposizione con una singola computazione. L'obbiettivo principale nelle WSN è quindi quello di minimizzare il consumo di risorse, massimizzando la sicurezza. Passiamo adesso all'analisi dei problemi principali nelle reti di sensori wireless.

Gestione delle chiavi:

Le chiavi sono necessarie per fornire confdenzialità, integrità e autenticazione e la loro gestione è necessaria per generare, distribuire, controllare e memorizzare il materiale di codifica. La gestione delle chiavi è un problema non risolto della WSN dato che la topologia della rete non è conosciuta a priori e vi è limitatezza di risorse come energia e range di trasmissione. Il problema di gestione delle chiavi nelle WSN può essere decomposto in:

· Key pre-distribution:

Le chiavi sono installate in tutti i sensori in una fase precedente al loro dispiegamento. Possono essere distribuite in tre diversi modi: network keying , una sola chiave per tutta la rete, Pairwise keying . Chiavi condivise tra coppie di nodi, Group Keying , chiavi condivise tra gruppi di nodi.

· Neighbor discovery:

Ogni nodo deve scoprire quali sono i suoi vicini, cioè quali nodi sono nel suo raggio di comunicazione, condividendo una chiave con essi e stabilendo la topologia di rete.

· End-to-end path-key establishment:

A due nodi, che non condividono una chiave ma sono collegati da più hops, deve essere assegnata una path key per una sicura comunicazione end-to-end

· Isolating aberrant nodes:

Identificare e isolare i nodi anomali che agiscono da nodi intermedi.

· Re-keying:

Azione da effettuare quando le chiavi scadono.

· Key-establishment latency:

La latenza è un significativo impedimento durante l'inizializzazione della rete sicura. La maggior latenza `e dovuta alle comunicazioni piuttosto che alla computazione.

Per risolvere tutti questi vari problemi nello scambio di chiavi si usano *protocolli di tipo ibrido*, mix di protocolli basati su chiave pubblica, *Pebblenets* , che si basa su cifratura simmetrica e fornisce group authentication, integrità del messaggio e confidenzialità, *LEAP*.

Routing sicuro:

Il routing deve essere reso sicuro da minacce esterne e da minacce interne (nodi malevoli o malfunzionanti) atte a renderlo inefficace. Nel routing sicuro vi sono vari problemi da affrontare, esso deve garantire un meccanismo di autenticazione che comporti poca computazione e basso overhead di comunicazione, un meccanismo di route discovery, un meccanismo di Route maintenance (garantire i route error messages e prevenire che nodi non autorizzati inviino questi messaggi) , determinare se i nodi intermedi hanno inoltrato effettivamente i messaggi, essere robusto contro flooding attack atti a degradare le performance del protocollo di routing. Vari protocolli sono atti a risolvere questi problemi : *SPINS (Security Protocols for Sensor Networks), Ariadne, INSENS*.

Prevenzione DoS:

L'obiettivo di un attacco DoS è quello di rendere una macchina inaccessibile agli utenti legittimi. Un attacco DoS è un qualsiasi evento che diminuisce o elimina le capacità della rete di eseguire le proprie funzionalità. Vi sono diversi metodi per la prevenzione di attacchi DoS , anche se continuano a rimanere un problema serio per il giusto funzionamento della rete. Si usano:

Meccanismi di autenticazione. Schema di rating: i vicini di ogni singolo nodo collaborano al fine di valutare i nodi. Route dos prevention: tenta di prevenire gli attacchi DoS favorendo la cooperazione tra più nodi.

Energia:

Molta energia è consumata nelle comunicazioni piuttosto che nelle computazioni, per salvare l'overhead di comunicazione è necessario salvare i dati in qualche nodo sensore vicino. Ogni nodo sensore quindi dovrebbe condividere una chiave globale con la stazione base ma anche condividere link keys con i sensori aggregatori.

Autenticazione broadcast:

Un attaccante non dovrebbe essere in grado di dichiarare se stesso come base station e mandare via broadcast falsi comandi ai nodi sensori. Alcuni protocolli come SPINS risolvono questo problema.

Attacchi alle reti WSN

Gli attacchi alle reti WSN sono simili a quelli che è possibile osservare in una rete di computer o in Internet , ma non sono limitati ad essi, esistono possibili attacchi su ogni livello dello stack protocollare.

Livello fisico

Uno dei principali attacchi al livello fisico è quello della manomissione diretta , da parte di un attaccante, del nodo, attraverso dumping delle chiavi salvate in memoria , per risolvere attacchi del genere bisogna utilizzare schemi di gestione delle

chiavi efficienti. Un altro comune attacco è quello dei Signal Jamming , cioè la trasmissione di un segnale radio che distrugge le comunicazioni decrementando il rapporto segnale radio, esso viene considerato un attacco di tipo Dos.

Livello data link

E' vulnerabile in quanto tutti i dati vengono trasmessi attraverso un mezzo insicuro (etere) , è infatti suscettibile ad attacchi di autenticazione , integrità e confidenzialità dei dati che devono essere instradati. Alcuni possibili attacchi sono : esaurimento delle risorse, risolvibile attraverso la limitazione del rate di trasmissione, e le collisioni, risolvibili attraverso error correcting code.

Livello di rete

Gli attacchi possibili al livello di rete sono di gran lunga superiori in numero a quelli possibili sugli altri livelli.

- Spoofing, altered or replaying routing information:

Sono tutti dei metodi di corruzione del traffico di rete, è infatti possibile creare routing loops, falsi messaggi d'errore, incrementare la latenza end-to-end, aumentare o diminuire il traffico di rete … . Questi attacchi possono portare a perdita dei pacchetti con continue ritrasmissione che riducono la durata di vita di un nodo, riducendo le performance della rete. Possibili soluzioni sono implementate in protocolli quali TinySec e SPINS.

- Selective forwarding or black holes:

I sensori instradano le informazioni verso la base station passando da nodi intermedi, alcuni di questi nodi possono essere compromessi e quindi possono, inoltrare i pacchetti in modo selettivo , non inoltrare alcun pacchetto ricevuto (black hole). Una soluzione è quella del multipath routing.

- Sink holes:

L'attaccante, fa in modo che la maggior parte del traffico passi attraverso un nodo compromesso, rendendolo attraente per l'algoritmo di routing , ad esempio falsificando o replicando un advertisement per una route ad alta qualità verso la base station. In questo modo la maggior parte dei dati passano per il nodo compromesso, dando cosi' inizio ad attacchi come wormhole ,

selective forwarding o evesdropping. Possibili soluzioni stanno nell'intervenire sul routing rendendolo più robusto (Geographic routing).

- Sybil attack:

Si ha soprattutto nelle reti di sensori che usano la distribuzione di sottoprocessi e la ridondanza di informazioni, in questo attacco un nodo malizioso può assumere più identità, riducendo l'efficacia dei meccanismi di tolleranza egli errori. Possibili soluzioni stanno nella valutazione dell'identità dei nodi , i nodi fidati valutano l'identità del nodo che si unisce alla rete. Un altra soluzione sarebbe quella di identificare la posizione geografica del nodo.

- Wormhole attack:

Due nodi malevoli creano un canale nascosto tra di loro attraverso il quale possono comunicare. Se un avversario è posto vicino ad una base station può distruggere completamente il processo di routing, se i due nodi sono in differenti posizioni della rete , possono catturare i pacchetti da una parte e ripeterli nell'altra parte della rete , inoltre possono creare sinkhole per attrarre a loro i nodi vicini. Alcune soluzioni sono date da Geographic forwarding (il next hop a cui trasmettere è il nodo più vicino al nodo destinazione, anche se non esclude del tutto il wormhole), Geographic leashes (tutti i nodi sensori devono essere sincronizzati e conoscere la loro posizione corrente. La trasmissione implica un overhead extra per tempo e posizione nel pacchetto da ciascun nodo sorgente nonché le firme digitali. Il ricevente può effettuare delle computazioni predefinite per verificare e autenticare la correttezza del pacchetto) , Temporal leashes (richiede che solo il timestamp venga trasmesso nel pacchetto) .

- Flooding:

Un nodo malevolo può causare parecchio traffico di messaggi inutili nella rete, questa continua ritrasmissione (anche broadcast) di traffico inutile, causa congestione e può portare ad esaurire tutte le risorse del nodo (Attacco di tipo DoS), una soluzione sta nell'autenticare il traffico broadcast.

- Hello Flood Attack:

Un attaccante con trasmissione radio elevata invia pacchetti HELLO a un certo numero di nodi sensori che sono distribuiti in una grande aria all'interno di una WSN. I sensori pensano che l'avversario sia un loro vicino, il nodo vittima prova quindi a passare per l'attaccante, che può quindi eseguire spoofing sul nodo. Le possibili Soluzioni consistono nel verificare la bidirezionalità di un collegamento o nell'autenticare , per ogni nodo, ognuno dei suoi vicini con un protocollo di verifica delle identità usando una base station fidata.

Caso di Studio: protocollo LEAP

Il protocollo LEAP (Localized Encryption and Authentication Protocol) è un Key-Management protocol per reti di sensori. Tenta, attraverso l'utilizzo di più livelli di chiavi, di minimizzare l'impatto di un nodo compromesso solo ai suoi immediati vicini.

I meccanismi di keying multipli servono a fornire confidenzialità e autenticazione (si può essere sicuri che a parlare è un nodo preciso se si è sicuri di condividere con esso una unica chiave con la quale i messaggi scambiati vengono crittografati).

Supporta vari pattern di comunicazione e garantisce local broadcast authentication. È un protocollo Energy-efficient.

I pacchetti scambiati dai nodi possono essere classificati in diverse categorie:

· pacchetti di controllo e pacchetti di dati;

· pacchetti broadcast e pacchetti unicast;

· comandi e letture del sensore;

Non tutti i tipi di messaggi hanno gli stessi requisiti di sicurezza:

· l'autenticazione è richiesta per tutti i tipi di pacchetti.

· La confidenzialità è richiesta solo per determinati tipi di pacchetti, come le letture di un nodo sensore, le queries inviate dalla base station.

I Meccanismi di single keying non sono appropriati per tutte le comunicazioni sicure richieste.

LEAP supporta 4 tipi di chiavi:

· Individual key che ogni nodo condivide con la base station.

· Pairwise key che ogni nodo condivide con un altro nodo sensore.

· Cluster key che ogni nodo condivide con più nodi vicini.

· Group key che ogni nodo condivide con tutti gli altri nodi nella rete.

Nella simulazione proposta viene implementata la parte di keying del protocollo.

Piuttosto che una rete peer to peer , per semplicità di implementazione viene presentata una rete client-server multi-thread.

La classe BaseStation{} rappresenta il sink node della rete e si occupa semplicemente di lanciare il server con i parametri desiderati.

La classe Controller{}; (il server) funge da mezzo di trasmissione tra i nodi. Essa chiamerà tutti i metodi di invio chiavi tra i vari nodi sensori.

I thread della classe Controller{} sono implementati nella classe SocketClientHandler{}. Qui si trovano tutti i metodi di invio dati tra la base station e ogni nodo sensore (Individual Key).

A ogni thread corrisponde un client (nodo sensore) rappresentato dalla classe SensorNode{} . Questa classe contiene i metodi di ricezione e invio delle chiavi ,oltre che metodi per l'invio della proprio posizione , necessari nel processo di discovery di nodi vicini. Contiene anche un vettore delle distanze dagli altri nodi (necessario per sopperire alla mancanza del mezzo wireless).

L'ultima classe è la HardwareSensorNode{} che serve ad istanziare ogni nuovo nodo sensore.

Descriviamo di seguito il processo di scambio chiavi che avviene nella simulazione.

Individual Key:

Ogni nodo ha un'unica chiave condivisa con la base station.Questa chiave viene usata per unicamente per comunicazioni Station-Nodo.

Il protocollo prevede che la chiave sia generata e pre-caricata in ogni nodo sensore prima del suo dispiegamento. Essa è generata a partire da una master-key attraverso una funzione pseudo-casuale.

Nella simulazione , in quanto abbiamo nodi virtuali che vengono collegati direttamente in rete, è stata implementata una funzione di generazione della chiave e di invio di essa da parte della base station al nodo , non appena esso si collega alla rete (equivalentemente la chiave poteva essere calcolata indipendentemente da nodo e base station a partire da una master key e dall'id del nodo).

Il metodo d'invio da parte della base station è :

```
private void SendIndividualKey() {
            BufferedWriter writer = new BufferedWriter
            (new OutputStreamWriter(clientSocket.getOutputStream()));

            individualKey=masterKey+id;

            writer.write(individualKey+"\n");
            writer.write("finish\n");
            writer.flush();
      }
```

Per semplicità il calcolo della Individual Key avviene come banale somma della master Key e dell'id del nodo appena connesso.
Il metodo di ricezione della chiave nel nodo è :

```
public void ReadIndividualKey() {
            in=new BufferedReader
                  (new InputStreamReader(clientSocket.getInputStream()));

            while (!(tmp = in.readLine()).equals("finish")) {
                     individualKey=tmp;
             }
      }
```

Pairwise Key:
ogni nodo condivide una chiave con ognuno dei suoi vicini immediati. Usata per comunicazioni che richiedono privacy o source authentication.
Lo scambio della pairwise key avviene attraverso 4 passaggi:
-Generazione di una master Key;
-Neighbor Discovery: il nodo invia la propria posizione in broadcast con un messaggio di HELLO e attende la risposta dei nodi vicini che arriverà con un messaggio di ACK. Il nodo vicino è

autenticato con la master key e l'id.
-Pairwise Key Establishment: la pairwise key viene calcolata in base alla propria metà della pairwise key più la chiave che viene inviata da ogni nodo vicino.
-Key Erasure: viene mantenuta memoria in ogni nodo della sola pairwise key.

Nella simulazione è stato adottato il seguente procedimento:
Master key, già pre-caricata in ogni nodo.
Neighbor Discovery: per mancanza del mezzo wireles , la discovery dei nodi e il controllo di vicinanza tra i nodi, viene demandato al server, che si occuperà di inoltrare le pairwise key tra i nodi accoppiati.

Il nodo sensore inizia il processo di discovery con l'invio di un Hello contenente la propria posizione.

```
public void SendPosition() throws IOException{

            writer.write("HELLO\n");
            writer.flush();
            writer.write(posizione+"\n");
            writer.flush()
}
```

quando il thread del nodo sensore appena connesso riceve la posizione del nuovo nodo, il server (classe Controller{}) prende la posizione del nodo corrente dal client e chiama un metodo per il calcolo della distanza da ognuno degli altri nodi della rete. Le distanze vengono salvate su di un vettore che viene passato al thread ed inviato al nodo sensore, in modo tale che ogni nodo conosce la distanza dai propri vicini.

```
for(counter=0;counter<100;counter++){

      if(clientList[counter]!=null){

      clientList[counter].SetPositionVector(CalcolaDistanza(counter));
      clientList[counter].SendPairwiseKey();

      }
}
```

dove :

```
private int[] CalcolaDistanza(int id){

	int[] vettoreDis= new int[100];
	int c;

	for(c=0;c<100;c++){
		if(c==id||clientList[c]==null){
		vettoreDis[c]=0;
	}

	else {
vettoreDis[c]=Math.abs(clientList[id].GetPosizione()-
clientList[c].GetPosizione());
		}

	return vettoreDis;
}
```

A questo punto il thread (SocketClientHandler{}) chiama la funzione per l'invio della pairwise key, simulando l'invio della chiave da parte di un nodo vicino:

```
public void SendPairwiseKey() {
		int c,i;

		for(c=0;c<100;c++){
//controllo che i nodi siano effettivamente vicini
			if(vettoreDis[c]!=0 && vettoreDis[c]<200){
				BufferedWriter writer = new BufferedWriter(new
OutputStreamWriter(clientSocket.getOutputStream()));

				//metà chiave del nodo con cui accoppiarsi
				tmp=individualHalfPairwiseKey+c;

				writer.write("ACK\n");
				writer.flush();
				writer.write(id+"\n"); //id del nodo
				writer.flush();
				writer.write(tmp+"\n"); //chiave
				writer.flush();
				writer.write(c+"\n"); //id del nodo con cui accoppiarsi
				writer.flush();

				//invio del vettore delle distanze al nodo sensore
				for(i=0;i<100;i++){
					writer.write(vettoreDis[i]+"\n");
					writer.flush();
				}

				ReadResponse();
			}
		}
	}
```

Pairwise key establishment:
ogni nodo adesso accoppiato calcola la chiave a partire dall'id del nodo con cui si sta accoppiando (verifica l'identità) .

Il nodo tiene inoltre traccia solo della chiave appena calcolata garantendo anche il processo di key Erasure. Nella funzione di lettura abbiamo:

```
if(tmp1.equals("ACK")){

                        id=Integer.parseInt(in.readLine());
                        integerPairKey=Integer.parseInt(in.readLine());
                        pairId=Integer.parseInt(in.readLine());

//calcolo                   della                   pairwise                   key

pairwiseKey[pairId]=(id+integerPairKey+IntegerIndividualHalfPairwiseKey)*3478+""
;
}
```

Cluster Key:
Chiave condivisa da ogni nodo con i suoi vicini ed è usata per garantire messaggi broadcast a livello locale.
La fase di cluster establishment segue la fase di pairwise key establishment.
· u genera una random key K u c
· Poi cifra questa chiave con la pairwise key condivisa con ogni vicino immediato con cui vuole stabilire una key cluster.
· Trasmette la chiave cifrata ad ogni vicino v i .
· v i decifra la chiave, la memorizza in una tabella e trasmette a u la sua cluster key.

Nella simulazione il processo è stato mantenuto, con l'aggiunta che il forwarding delle cluster keys avviene da parte del server sempre per la mancanza del mezzo wireless e dell'architettura peer to peer.

Appena il nodo finisce di ricevere la propria pairwise key e il vettore delle distanze aggiornato, e il thread del nodo è in attesa di ricevere qualcosa, viene chiamata dal nodo sensore la funzione :

```
private void SendClusterKey(int c) throws IOException, InterruptedException{
            int tmp;

                  tmp=randomClusterKey+Integer.parseInt(pairwiseKey[c]);
                  writer.write("CLUSTERING\n");
                  writer.flush();
                  writer.write(tmp+"\n");
                  writer.flush();
                  writer.write(c+"\n");
                  writer.flush();
                  Thread.sleep(2000);

            }
      }
```

con questa funzione viene generata una chiave cifrata con la pairwise key e inviata al thread per essere forwardata al nodo specificato dalla variabile c.

A questo punto il SocketClientHandler{} legge la chiave cifrata e la salva su un vettore, pronta per l'invio:

```
if(clientInput.equals("CLUSTERING")){

                        tmpKey=Integer.parseInt(stdIn.readLine());
                        clusterId=Integer.parseInt(stdIn.readLine());
                        clusterKey[clusterId]=tmpKey;

                        break;
                }
```

dal Controller{} viene chiamata una funzione di inoltro della cluster key cifrata per ogni thread :

```
for(counter=0;counter<100;counter++){
      for(int i=0;i<100;i++){
            clientList[counter].SendClusterKey(i);
      }
}
```

dove:

```
public void SendClusterKey(int nodeId){
      BufferedWriter writer = new BufferedWriter(new
      OutputStreamWriter(clientSocket.getOutputStream()));


                if(clusterKey[nodeId]!=0){

                        writer.write("CLUSTERINGREAD\n");
                        writer.flush();
                        writer.write(clusterKey[nodeId]+"\n");
                        writer.flush();
                        writer.write(nodeId+"\n");
                        writer.flush();

                        Thread.sleep(2000);
                }
      }
}
```

A questo punto il nodo sensore è in attesa di ricevere la chiave del suo nodo accoppiato :

```
if(tmp1.equals("CLUSTERINGREAD")){
      c=Integer.parseInt(in.readLine());
      pairId=Integer.parseInt(in.readLine());

      clusterKey[pairId]= c - (Integer.parseInt(pairwiseKey[pairId]));

      Thread.sleep(2000);
      break;
}
```

la chiave viene salvata su un vettore di chiavi per ogni vicino.
Ogni nodo n possiede una stessa group key per il nodo A.
Se il nodo A volesse inviare un messaggio in broadcast basterebbe cifrarlo con la proprio group Key, cosi tutti i nodi ad esso vicino sarebbero in grado di leggerlo.

Il protocollo LEAP possiede una funzione di revoca dei nodi malevoli o danneggiati basata sul protocollo UTESLA che non viene analizzato nella presente relazione.

Il protocollo LEAP continua a non essere sicuro nel caso di cattura di un nodo. Un attaccante , prendendo possesso di un nodo, ottiene tutte le informazioni da esso contenute, potendo riutilizzarle per scopi malevoli sulla rete.