Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Informatica per il Management

Design e implementazione di un protocollo basato su Blind Signatures per il Mobile Crowdsensing

Relatore:
Prof.
FEDERICO MONTORI

Presentata da:
FRANCESCA FOLLI

Sessione II
Anno Accademico 2023/2024

Sommario

Oggigiorno, il Mobile Crowdsensing sta acquisendo sempre più interesse grazie alla sua sostenibilità e ai suoi molteplici utilizzi, resi possibili dalle capacità avanzate di rilevamento e comunicazione dei dispositivi mobili moderni. Tuttavia, la partecipazione a campagne di crowdsensing, specialmente se incentivate tramite ricompense, comporta rischi significativi per la privacy degli utenti. In questa tesi, viene proposto un protocollo di Mobile Crowdsensing basato sulle Blind Signatures, che ha l'obiettivo di proteggere la privacy degli utenti, garantendo al contempo ricompense differenziate, senza compromettere le prestazioni del sistema. Lo schema proposto dimostra come questa tecnologia possa garantire l'anonimato degli utenti, introducendo un overhead computazionale limitato e sostenibile, rivelandosi una soluzione efficace per affrontare le crescenti sfide legate alla sicurezza.

Introduzione

Negli ultimi anni, il Mobile Crowdsensing (MCS) ha acquisito una crescente popolarità grazie alla diffusione capillare di dispositivi mobili dotati di sensori avanzati. Gli smartphone, i tablet e altri dispositivi mobili moderni, grazie alle loro capacità di rilevamento ambientale, consentono di raccogliere grandi quantità di dati in tempo reale, provenienti da un'ampia base di utenti distribuiti su vaste aree geografiche. Questa tecnologia ha trasformato settori come il monitoraggio ambientale, la gestione del traffico e la sanità, offrendo nuovi strumenti per migliorare l'efficienza e la qualità dei servizi.

Il Mobile Crowdsensing si basa sull'idea di utilizzare la partecipazione attiva degli utenti per raccogliere dati, consentendo alle persone di contribuire con informazioni utili tramite i sensori integrati nei loro dispositivi. Tuttavia, con l'aumento della raccolta di dati personali, emergono sempre più sfide riguardo alla sicurezza e alla privacy degli utenti, soprattutto quando i dati raccolti possono essere tracciati o associati a un individuo specifico.

L'obiettivo di questa tesi è quello di proporre un nuovo sistema di Mobile Crowdsensing che sia in grado di garantire alti livelli di privacy attraverso l'uso di meccanismi crittografici avanzati, come le blind signatures. Questo sistema permette di raccogliere dati in modo anonimo e sicuro, proteggendo l'identità degli utenti che partecipano al crowdsensing.

Attraverso una serie di test, verranno misurati e analizzati i tempi di risposta sia per le comunicazioni baseline (non crittografate) che per quelle crittografate, con l'obiettivo di dimostrare che l'overhead introdotto dalle operazioni crittografiche è gestibile e che il sistema proposto può garantire

privacy senza compromettere le prestazioni.

Nel primo capitolo viene presentato lo stato dell'arte, offrendo una panoramica sul Mobile Crowdsensing e sul suo sviluppo nel corso degli anni. Viene analizzata l'evoluzione di questa tecnologia. Particolare attenzione è rivolta al problema cruciale che emerge con l'adozione del MCS: la protezione della privacy degli utenti. Il capitolo affronta le principali sfide legate alla salvaguardia dei dati personali e illustra le soluzioni proposte dalla letteratura per mitigare i rischi associati alla condivisione di informazioni sensibili. Nel secondo capitolo viene approfondita la tecnologia delle blind signatures, elemento chiave della soluzione proposta, illustrandone i principi di funzionamento e le ragioni per cui è stata scelta per garantire la privacy degli utenti nel sistema di Mobile Crowdsensing. Il terzo capitolo è dedicato alla descrizione dell'architettura della soluzione proposta, fornendo una visione dettagliata dei componenti e dei flussi di comunicazione che costituiscono il sistema. Successivamente, nel quarto capitolo, viene esposta l'implementazione pratica della soluzione, illustrando le tecnologie e i meccanismi discussi nei capitoli precedenti. Il quinto capitolo descrive i test effettuati per valutare le prestazioni e l'efficacia della soluzione proposta, analizzando i risultati ottenuti e le relative considerazioni. Infine, nel sesto e ultimo capitolo, viene presentata l'applicazione mobile nativa per Android progettata e sviluppata per dimostrare l'applicabilità pratica del sistema proposto.

Indice

Elenco delle figure

Capitolo 1

Stato dell'arte

1.1 Mobile Crowdsensing

Il Mobile Crowdsensing (MCS) è un paradigma innovativo che sfrutta la diffusione capillare di dispositivi mobili, come smartphone e wearable, per raccogliere dati attraverso la partecipazione di un'ampia rete di utenti [1]. Questi dispositivi, equipaggiati con una vasta gamma di sensori integrati, come GPS, accelerometri, microfoni e fotocamere, permettono di raccogliere dati su larga scala in modo distribuito e dinamico, rendendo il MCS una soluzione potente ed economica per molteplici applicazioni.

Il termine "Mobile Crowdsensing" è stato introdotto da Ganti et al. [2], che lo definisce come un'evoluzione del "mobile phone sensing", con l'obiettivo di coinvolgere attivamente i cittadini nel monitoraggio di fenomeni ambientali e sociali. Il paradigma combina l'intelligenza umana con il machine learning, sfruttando la partecipazione attiva degli utenti per migliorare la copertura spaziale e la consapevolezza del contesto. Questo approccio consente di raccogliere dati di qualità superiore rispetto ai tradizionali network di sensori statici, poiché i dispositivi mobili permettono appunto una maggiore flessibilità e quindi un accesso più capillare alle aree urbane.

Il MCS si dimostra particolarmente adatto per il monitoraggio urbano e ambientale, poiché le città moderne stanno affrontando sfide significative

nella gestione e manutenzione delle infrastrutture. Coinvolgere direttamente i cittadini in questo processo può migliorare il monitoraggio e l'efficienza dei servizi. Inoltre, il MCS rappresenta una soluzione fondamentale per lo sviluppo delle smart cities del futuro, con l'obiettivo di migliorare la qualità della vita dei cittadini e introdurre nuove prospettive per le società urbane [1].

Le applicazioni tradizionali di monitoraggio urbano si affidano a infrastrutture di rilevamento specializzate, come le stazioni di monitoraggio della qualità dell'aria, che sono costose da implementare e mantenere. Inoltre, tali infrastrutture presentano il limite della scarsa riusabilità per diversi scopi applicativi, ostacolando così la crescita rapida e diversificata delle applicazioni di rilevamento su larga scala. Il MCS, invece, offre un modello complementare al paradigma tradizionale, sfruttando la mobilità degli utenti, i sensori già presenti nei dispositivi mobili e l'infrastruttura wireless esistente per raccogliere dati ambientali in modo flessibile e a basso costo. Questo consente di monitorare aree urbane che non sono coperte dalle infrastrutture di rilevamento specializzate, riducendo così i costi operativi e ampliando l'accessibilità ai dati [4].

L'architettura di un sistema MCS [1] può essere suddivisa in quattro livelli principali:

- Livello Applicativo: Coinvolge l'interfaccia con l'utente e include funzioni come il reclutamento dei partecipanti, la raccolta di dati e la visualizzazione dei risultati. Questo livello è essenziale per organizzare campagne di crowdsensing efficienti.

- Livello Dati: Responsabile della raccolta, gestione e analisi dei dati. I dati raccolti dai partecipanti vengono aggregati e analizzati per fornire informazioni utili. In questo livello, l'integrità dei dati e la loro accuratezza sono sfide importanti, soprattutto quando i dati provengono da sensori a basso costo.

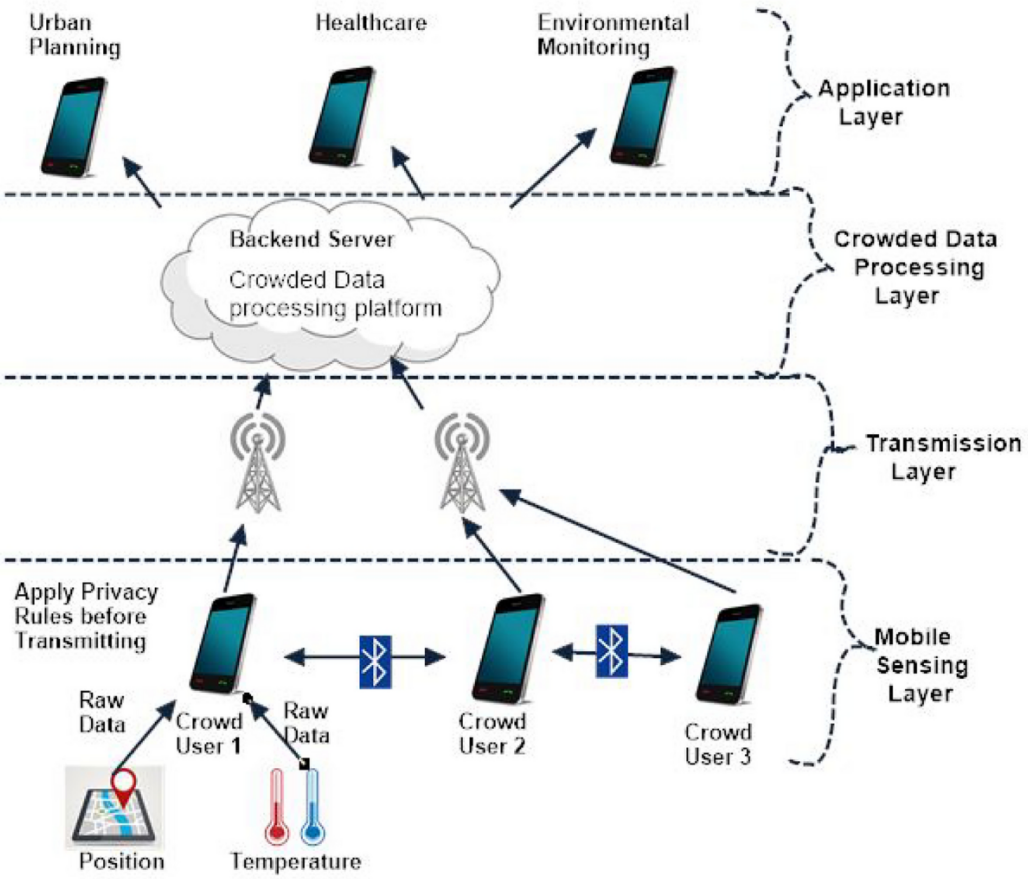

Figura 1.1: Architettura di un sistema MCS

- Livello di Trasmissione: Riguarda le tecnologie utilizzate per trasmettere i dati dai dispositivi mobili ai server di destinazione. Le comunicazioni avvengono principalmente tramite reti cellulari, Wi-Fi o tecnologie Bluetooth. La scelta del canale di comunicazione influisce significativamente sull'efficienza energetica del sistema.

- Livello di Rilevamento: Include tutti i sensori fisici utilizzati dai dispositivi mobili per raccogliere dati. I sensori presenti nei dispositivi mobili possono essere affetti da problemi di precisione, come il rumore nei segnali raccolti o la mancanza di accuratezza temporale.

1.2 Privacy nel Mobile Crowdsensing

Nonostante la convenienza e i bassi costi, per operare in modo efficiente i sistemi di crowdsensing necessitano di un numero sufficiente di partecipanti. I dati raccolti dai sensori di un dispositivo personale possono rivelare informazioni sensibili sull'utente, e metterne a rischio la privacy.

La privacy è diventata quindi un problema cruciale, poiché i dispositivi raccolgono dati sensibili degli individui e la loro divulgazione può avere serie implicazioni. L'identità dell'utente viene tipicamente registrata per valutare ad esempio la credibilità dei dati e l'affidabilità dell'utente, tuttavia queste informazioni possono portare a un profilo dell'utente. Inoltre, alcuni framework di reclutamento acquisiscono dati, come il registro delle chiamate telefoniche, che nella maggior parte dei casi non sarebbe considerato accettabile in termini di privacy [3].

Alcuni dati possono direttamente rivelare informazioni personali, come la posizione o lo stato di salute. Anche dati apparentemente non correlati possono essere analizzati per dedurre informazioni private. Ad esempio, i dati dell'accelerometro, che di per sé difficilmente sono considerati sensibili, possono essere utilizzati per classificare le attività dell'utente; le tracce GPS possono essere usate per identificare l'identità dell'utente; e i dati dei sensori di movimento possono essere utilizzati per dedurre le password dell'utente [5].

Per alleviare questo ostacolo, negli ultimi anni la comunità di ricerca ha dedicato molti sforzi all'indagine sulle questioni di privacy e sono stati proposti sistemi di crowdsensing che preservano la privacy per affrontare la riluttanza alla partecipazione da parte degli utenti preoccupati.

Tra le strategie messe in atto [10] quella di Kapadia et al. utilizza la tecnica della "Tessellation Map" per rappresentare le località come regioni invece di punti per migliorare la tutela della privacy. Il problema della privacy può, in alternativa, essere affrontato dall'aggregazione dei dati, inviando questi nella forma di un data set multidimensionale, ma questo impedirebbe di processare i dati singolarmente. Zhu et al. ha inoltre proposto uno sche-

ma di aggregazione dei dati che tuteli la privacy basato sul teorema cinese del resto, che impiegava tecniche di cifratura omomorfica e fattore cieco di Paillier. Questo schema aumenta la robustezza del sistema garantendo allo stesso tempo la privacy dei dati sensoristici; tuttavia, non riesce ancora a raggiungere una gestione granulare dei dati. Alcuni schemi assicurano l'integrità dei messaggi inviati grazie a firme digitali, ma questi hanno elevati overhead computazionali poichè utilizzano funzioni molto costose.

Molti altri sistemi sono stati proposti [5]: PoolView perturba i dati prima di immetterli per preservare la privacy, in Prisense i dati vengono inoltrati tra gli utenti prima di essere inviati al server per nascondere l'origine dei dati.

La tutela della privacy, tuttavia, non è sufficiente per incentivare la partecipazione. Sono stati fatti diversi sforzi per proporre meccanismi di incentivi per attirare gli utenti. Ad esempio, alcuni hanno proposto meccanismi di incentivi basati su aste, altri propongono che i partecipanti possano guadagnare crediti in cambio dei dati dei loro sensori. Anche micropagamenti per i dati condivisi possono essere efficaci.

Anche gli sforzi fatti per attrarre gli utenti tramite meccanismi di incentivo può però esporre gli stessi a rischi per la privacy. Un tipico processo di ricompensa può, infatti, rivelare l'identità dei beneficiari. In primo luogo, gli utenti devono dimostrare il loro contributo alla raccolta dei dati, il che potrebbe rivelare chi sono, quando e dove sono stati raccolti i dati e persino quali fossero i dati. In secondo luogo, dopo aver verificato il contributo, la procedura di ricompensa potrebbe anche rivelare l'identità del beneficiario a seconda della forma di ricompensa (ad esempio, crediti sul conto, bonifico bancario e così via) [5].

Tra i pochi studi che già propongono l'utilizzo della tecnologia delle firme cieche per preservare la privacy nel campo del MCS ci sono [10] e [11], che non diversificano però in alcun modo le ricompense all'utente. Un'altra soluzione proposta in questo senso è [5], il sistema funziona separando fisicamente il processo di raccolta dati dal processo di erogazione delle ricompense.

Nella soluzione proposta di seguito, viene mantenuto un livello di elasticità maggiore: i due processi vengono separati solo temporalmente e viene mantenuta la possibilità di offrire ricompense differenziate sia a livello quantitativo che qualitativo. Le ricompense variano in base alla tipologia dei dati raccolti e alla quantità di dati già forniti dagli utenti per quel particolare tipo di dato sensoristico, garantendo così una flessibilità nelle ricompense che rispecchia il valore del contributo offerto.

Capitolo 2

Le Blind Signatures

Le blind signatures, o firme cieche, sono una variante delle firme digitali tradizionali in cui il firmatario appone una firma su un messaggio senza conoscerne il contenuto. Questo processo viene utilizzato per garantire che il messaggio possa essere verificato come autentico in seguito, ma senza rivelare al firmatario il contenuto del messaggio durante il processo di firma.

Le firme cieche sono state inizialmente proposte da David Chaum [6] nel 1982 nel contesto del denaro elettronico. Chaum desiderava creare un sistema di pagamento elettronico che garantisse sia l'anonimato del cliente sia la non tracciabilità delle transazioni.

Il suo lavoro pionieristico ha introdotto i concetti di non rintracciabilità - la banca non può collegare una moneta spesa da un utente alla moneta originariamente firmata - e non collegabilità - non è possibile collegare tra loro due transazioni effettuate dallo stesso utente. In questo schema, un utente ottiene una moneta elettronica firmata dalla banca, la spende in un negozio e il negozio la restituisce alla banca per la verifica. La banca può verificare la validità della moneta, ma non può collegarla all'utente originario che l'ha richiesta, preservando così l'anonimato.

Un esempio specifico di firma cieca è la firma cieca RSA [7], che si basa sul classico algoritmo RSA. Nel processo [8] una firma RSA tradizionale viene calcolata elevando il messaggio M all'esponente segreto d modulo il modulo

pubblico N. La versione cieca utilizza un valore casuale r, tale che r sia relativamente primo rispetto a N, ovvero $\gcd(r, N) = 1$. Il valore r viene elevato all'esponente pubblico e modulo N, e il valore risultante $r^e \mod N$ viene utilizzato come fattore di offuscamento.

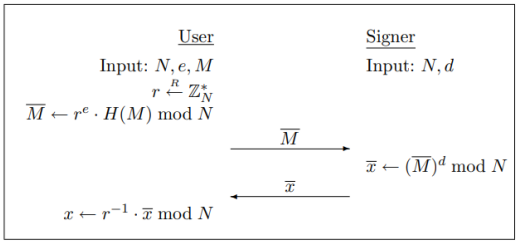

Figura 2.1: Protocollo di firma cieca RSA

L'autore del messaggio calcola il prodotto del messaggio e del fattore di offuscamento, cioè:

$$\overline{M} \equiv M \cdot r^e \mod N$$

e invia il valore risultante $\overline{M}$ all'autorità di firma.

L'autorità di firma quindi calcola la firma offuscata $\overline{x}$ come:

$$\overline{x} \equiv (\overline{M})^d \mod N$$

Il valore $\overline{x}$ viene inviato all'autore del messaggio, che può poi rimuovere il fattore di offuscamento per rivelare x, la firma RSA valida di M:

$$x \equiv \overline{x} \cdot r^{-1} \mod N$$

Questo funziona poiché le chiavi RSA soddisfano l'equazione:

$$r^{ed} \equiv r \mod N$$

e quindi:

$$x \equiv \overline{x} \cdot r^{-1} \equiv (M')^d \cdot r^{-1} \equiv M^d \cdot r^{ed} \cdot r^{-1} \equiv M^d \cdot r \cdot r^{-1} \equiv M^d \mod N$$

Pertanto, x è effettivamente la firma di M.

Una firma cieca garantisce quindi perfetta riservatezza al contenuto del messaggio. A causa di questa caratteristica però, il firmatario non può assicurarsi che ciò che firma contenga informazioni corrette. Questo che potrebbe rivelarsi uno svantaggio, viene risolto nella soluzione proposta al capitolo 3, stabilendo che il messaggio debba essere un valore random di 16 cifre, da utilizzare in fase di verifica per controllare la veridicità e la correttezza della firma stessa.

Capitolo 3

Architettura

La soluzione proposta in ambito di Mobile Crowdsensing per la protezione della privacy degli utenti che contribuiscono alla campagna di crowdsensing è basata sulla tecnologia delle blind signatures.

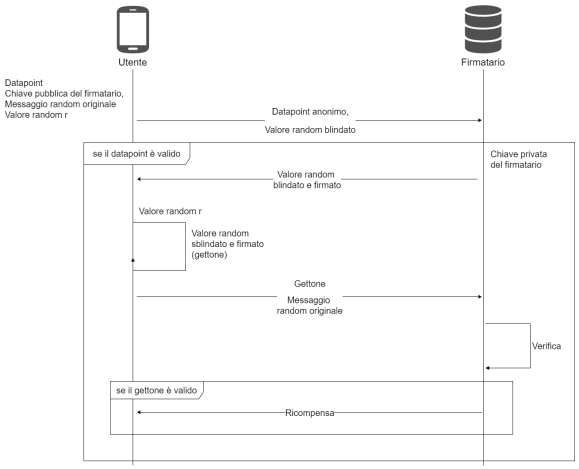

Figura 3.1: Modello della soluzione proposta

L'idea di base è la seguente 3.1:

1. *Raccolta dato*: l'utente raccoglie un dato sensoristico e impacchetta il relativo datapoint anonimo, composto dal dato stesso, la tipologia di dato raccolto, le coordinate del punto in cui il dato è stato raccolto e il timestamp.

2. *Generazione messaggio random*: l'utente genera un valore random di 16 cifre.

3. *Offuscamento messaggio random*: il messaggio precedentemente generato viene offuscato utilizzando la chiave pubblica del firmatario e un valore casuale (r), scelto in modo che sia coprimo con il modulo pubblico della chiave, come previsto dalla tecnologia delle blind signatures. Successivamente, il datapoint anonimo e il messaggio offuscato vengono inviati al firmatario per essere firmati.

4. *Firma del messaggio random*: il datapoint inviato viene controllato e se il contenuto è utile, il firmatario procede a firmare il messaggio offuscato ricevuto utilizzando la propria chiave privata e lo invia indietro all'utente.

5. *Rimozione offuscamento*: l'utente riceve il messaggio offuscato e firmato dal firmatario e procede a rimuovere l'offuscamento utilizzando il valore casuale r, scelto precedentemente durante il processo di blindaggio, sfruttando le proprietà delle blind signatures. Il messaggio risultante, ovvero il messaggio firmato e non offuscato, diventa il gettone che l'utente potrà utilizzare in qualsiasi momento futuro per ottenere il compenso spettante in base al contributo fornito.

6. *Invio gettone e verifica*: in un secondo momento, a discrezione dell'utente, viene inviato al firmatario il gettone (cioè il messaggio firmato e non offuscato) insieme al messaggio random originale. Egli verifica che il gettone sia valido. In caso affermativo, controlla che il gettone non sia

già stato utilizzato in precedenza. Se il gettone è valido e non è ancora stato utilizzato, viene restituita all'utente la ricompensa dovuta.

Nell'implementazione della soluzione proposta, una delle principali innovazioni risiede nella differenziazione delle ricompense sia a livello quantitativo che qualitativo. In particolare, per la discriminazione quantitativa, sono state definite due fasce, con un valore soglia di 50. Questo significa che le ricompense variano in base alla quantità di dati già raccolti. Per quanto riguarda la discriminazione qualitativa, sono state stabilite tre fasce, determinate dal tipo di sensore utilizzato per la raccolta dei dati.

Per implementare questo meccanismo di differenziazione, il server utilizza 6 coppie di chiavi crittografiche, anziché una sola. Ogni coppia di chiavi è associata a una diversa combinazione tra la fascia di quantità di dati raccolti e la tipologia di sensore utilizzato, garantendo così una gestione delle ricompense dinamica e adeguata al contributo fornito dall'utente.

Di conseguenza, al punto 3 dell'architettura della soluzione, il messaggio random generato al punto 2 viene offuscato non con una, ma con tutte e sei le chiavi pubbliche del firmatario. Il datapoint anonimo viene quindi inviato al server insieme a tutti e sei i messaggi offuscati.

Al punto 4, il firmatario, una volta ricevuti il datapoint e i sei messaggi offuscati, deve determinare quale dei sei token firmare utilizzando la chiave privata corrispondente, sulla base delle discriminazioni quantitative e qualitative precedentemente descritte. Dopo aver firmato uno dei messaggi offuscati, il firmatario restituisce al client sia il messaggio firmato sia un'indicazione della chiave utilizzata per la firma.

Questo passaggio è cruciale, poiché consente all'utente, al punto 5, di rimuovere correttamente l'offuscamento utilizzando il valore casuale (r) associato alla chiave indicata, ottenendo così il gettone firmato e valido per la successiva verifica e riscossione della ricompensa.

Questa soluzione garantisce l'anonimato dei dati sensoristici separando il processo di raccolta dei dati dall'erogazione delle ricompense. Questo approccio è sicuro perché utilizza le blind signatures, che permettono di firmare

i token inviati dagli utenti senza che il server conosca il contenuto originale del messaggio o l'identità dell'utente.

La sicurezza del sistema deriva dal fatto che i dati raccolti vengono gestiti in modo anonimo e indipendente rispetto all'erogazione delle ricompense. Questo significa che, anche se un utente invia un dato sensoriale, il server non può associarlo direttamente alla ricompensa erogata. Non esiste infatti alcun legame tra un valore offuscato e un valore firmato ma non offuscato, impedendo così qualsiasi collegamento tra un gettone inviato da un utente e un messaggio offuscato che sarebbe riconducibile a un datapoint specifico.

Ciò che viene offuscato è un semplice messaggio random, il cui contenuto non ha importanza intrinseca, poiché serve unicamente come prova del contributo fornito dall'utente. Questo elimina il problema tradizionalmente legato all'uso delle blind signatures, in cui il firmatario potrebbe accidentalmente firmare qualcosa di sconveniente o sconosciuto. Nel sistema proposto, il messaggio random è usato soltanto per la verifica finale e, nel caso in cui il messaggio non fosse corretto, la ricompensa non viene erogata. Di conseguenza, la sicurezza è garantita senza compromettere l'anonimato o l'integrità del processo di firma e verifica.

Capitolo 4

Implementazione

In questo capitolo viene presentata l'implementazione della soluzione proposta, disponibile al seguente link:

`https://github.com/frafo227295/TESI_FOLLI.git`

Tale soluzione è costituita da un Server Flask, che è il firmatario, e un Client, che rappresenta l'utente che raccoglie i dati sensoristici con il proprio dispositivo mobile. Fondamentale è l'utilizzo della libreria RSA per poter integrare la tecnologia delle blind signatures:

```
from Crypto.PublicKey import RSA
```

4.1 Server

Il server Flask stabilisce innanzitutto una connessione a un database non relazionale MongoDB, dove vengono memorizzati i datapoint inviati dagli utenti, insieme ai messaggi casuali di 16 cifre utilizzati per la verifica dei token e la gestione delle ricompense.

Utilizzando la libreria RSA, il server genera sei coppie di chiavi:

```
privkey = RSA.generate(2048)
pubkey = privkey.publickey()
```

ciascuna delle quali viene impiegata per firmare i messaggi relativi a differenti tipologie di contributi e quindi diverse categorie di ricompense, come presentato in precedenza:

high_many_key firma fotografie, audio o registrazioni del rumore ambientale, quando ne sono stati già registrati più di 50 dati;

high_few_key firma fotografie, audio o registrazioni del rumore ambientale, quando ne sono stati registrati meno di 50 dati;

medium_high_key firma dati da giroscopio o sensore della luce, quando ne sono stati già registrati più di 50 dati;

medium_few_key firma dati da giroscopio o sensore della luce, quando ne sono stati registrati meno di 50 dati;

low_high_key firma dati da accelerometro, quando ne sono stati già registrati più di 50 dati;

low_few_key firma dati da accelerometro, quando ne sono stati registrati meno di 50 dati.

Ogni volta che il server riceve un datapoint anonimo insieme ai 6 token firmati con le rispettive chiavi pubbliche, determina quale token selezionare in base alla discriminazione tra le diverse categorie di contributi di cui sopra. Successivamente, il server firma uno solo tra i token ricevuti e lo restituisce al client, accompagnato dalla tipologia di chiave utilizzata per la firma.

Funzione per la firma del token:

```
def sign(blinded, server_priv_key):
    N = server_priv_key.n
    D = server_priv_key.d
    signed = pow(blinded, D, N)
    return signed
```

Quando il server riceve una richiesta di verifica e compenso da parte di un utente, determina nuovamente quale sia la chiave corretta da utilizzare

per la verifica del token. Una volta identificata la chiave appropriata, esegue la verifica e, se il token è valido, procede con l'assegnazione del compenso all'utente.

Funzione per la verifica del token:

```
def verify(unblinded, server_pub_key, message):
    N = server_pub_key.n
    E = server_pub_key.e
    msg_verify = pow(unblinded, E, N)
    return message == msg_verify
```

Se la funzione restituisce *true*, il server verifica che il *message* non sia stato già utilizzato, controllando che non sia presente nel database MongoDB. In questo modo, si evita che lo stesso messaggio possa essere utilizzato più volte per richiedere una ricompensa.

```
already_used = 1 if messagerandom.find_one({ 'random':
    message }) else 0
```

Se la verifica ha esito positivo e la ricerca del messaggio random nel database risulta negativa, ovvero il messaggio non è stato già utilizzato, viene calcolata la ricompensa dovuta in base al contributo fornito. La ricompensa viene quindi restituita al Client insieme alla conferma della validità del token.

4.2 Client

Il client è responsabile della raccolta dei dati sensoristici, dell'oscuramento dei token utilizzando le firme cieche, e della comunicazione con il server per richiedere la firma dei token, verificarne la validità e ricevere le relative ricompense.

Dopo aver ricevuto le 6 chiavi pubbliche dal server e generato un valore random di 16 cifre, il client offusca (blinda) tale messaggio random utilizzando ciascuna delle 6 chiavi pubbliche e un valore casuale r, coprimo con N

(il modulo della chiave pubblica RSA). Questo processo genera 6 messaggi offuscati, uno per ogni chiave pubblica.

Funzione per l'offuscamento del messaggio random:

```
def blind(message, server_pub_key):
    N = int(server_pub_key['n'])
    E = int(server_pub_key['e'])

    while True:
        r = random.randint(1, N-1)
        if gcd(r, N) == 1:
            break

    blinded = (message * pow(r, E, N)) % N
    return blinded , r
```

Il datapoint e i token vengono inviati al server tramite una richiesta HTTP POST. In questa richiesta, i dati sensoristici raccolti e i token offuscati generati vengono trasmessi al server per essere elaborati e firmati.

Esempio di richiesta inviata dal client al server con dati fittizi:

```
requestBody = {
    'datapoint' : {
        'geolocation' : '0.0 , 0.0',
        'datatype' : 'light',
        'value' : '0.0',
        'timestamp' : '01/01/1990'
    },
    'tokens' : tokens
    }
}
```

Una volta ricevuto questo messaggio, il client rimuove l'offuscamento utilizzando il valore casuale r che aveva generato in precedenza. Questo pro-

cesso di rimozione dell'offuscamento permette al client di ottenere il gettone firmato, che può quindi essere collezionato immediatamente.

Funzione per togliere l'offuscamento:

```
def unblind(signature, r, N):
    r_inv = inverse(r, N)
    unblinded = (signature * r_inv) % N
    return unblinded
```

Il messaggio *unblinded* viene utilizzato idealmente in un secondo momento come gettone per ottenere la ricompensa dovuta. Una volta sbloccato, il client può inviare questo gettone al server in una richiesta successiva per verificare la validità del token e ricevere la ricompensa associata al contributo fornito.

4.3 Comunicazione Client-Server

La comunicazione tra Client e Server in questo sistema di mobile crowdsensing avviene utilizzando il protocollo HTTP per l'invio e la ricezione di messaggi. Il sistema adotta un'architettura Client-Server, in cui il client raccoglie i dati dai sensori e invia richieste al server tramite vari endpoint dedicati, finalizzati all'ottenimento di firme digitali, alla verifica dei token e alla ricezione delle ricompense associate.

La sicurezza della comunicazione è garantita dall'uso della crittografia RSA e dal meccanismo delle blind signatures, che consente di firmare i token preservando l'anonimato dell'utente durante le fasi di firma e verifica, impedendo al server di associare al messaggio random originale il datapoint inviato corrispondente.

Il server offre una serie di endpoint che gestiscono sia i datapoint anonimi raccolti tramite blind signatures, sia i datapoint baseline, ovvero quelli inviati con un identificativo utente e senza l'utilizzo di firme cieche, permettendo così diverse modalità di interazione con il sistema, in base al livello di privacy richiesto.

Esempio di datapoint baseline con dati fittizi:

```
requestBaselineBody = {'datapoint' : {
        'geolocation' : '0.0 , 0.0',
        'datatype' : 'light',
        'value' : '0.0',
        'timestamp' : '01/01/1990'
},
'identifier' : 'Francesca Folli'
}
```

Nel caso di datapoint baseline, la richiesta HTTP POST restituisce all'utente un gettone che rappresenta direttamente il valore stesso della ricompensa dovuta.

L'uso delle firme cieche permette al client di ottenere una firma dal server senza rivelare il contenuto del messaggio originale. Inoltre, la verifica del token e l'assegnazione delle ricompense avvengono in modo sicuro e tracciabile, prevenendo attacchi di tipo replay grazie al controllo dei token già utilizzati.

Capitolo 5

Risultati ottenuti

In questo capitolo vengono illustrati i test effettuati sulla soluzione proposta e analizzati i risultati ottenuti. I test sono stati progettati per valutare l'efficacia del sistema in termini di prestazioni, sicurezza e gestione delle ricompense, confrontando l'uso delle firme cieche con l'approccio baseline, e misurando l'overhead introdotto dalla crittografia.

Oltre a questi aspetti, è stata calcolata la differenza nominale tra le richieste e le risposte baseline e quelle crittografate, per determinare quanto la crittografia aumenti le dimensioni dei dati trasmessi. Le metriche di interesse includono i tempi di risposta, l'overhead computazionale, la differenza nominale e la corretta gestione della privacy attraverso i meccanismi crittografici.

5.1 Test svolti

Per valutare l'efficienza del sistema e l'eventuale overhead introdotto dalla crittografia, sono stati eseguiti una serie di test attraverso l'interfaccia di loopback (127.0.0.1). L'utilizzo di questa interfaccia permette di simulare una comunicazione client-server locale senza coinvolgere una rete esterna, eliminando così la variabile della latenza di rete e consentendo di misurare con precisione le prestazioni del sistema. Tutti i test sono stati svolti su un

computer dotato di processore AMD Ryzen 5 5500U e RAM installata di 8,00 GB.

I test sono stati organizzati in due benchmark principali, con l'obiettivo di misurare e confrontare l'overhead tra la comunicazione baseline (non crittografata) e quella crittografata.

Il primo benchmark aveva l'obiettivo di misurare il tempo di risposta per la trasmissione di dati baseline rispetto a quelli crittografati, con particolare attenzione alla ricezione del gettone. Nel caso di dati crittografati, il processo includeva il blindaggio del messaggio, la richiesta di firma al server e lo sblindaggio del token firmato. Per la trasmissione di dati baseline (non crittografati), il processo consisteva nell'invio del datapoint con identificativo e la ricezione di una risposta diretta. L'obiettivo era quantificare l'impatto della crittografia e delle operazioni di firma sul tempo complessivo di ricezione del gettone.

Il secondo benchmark ha valutato l'overhead legato alla verifica delle richieste e alla ricezione della ricompensa. Nel caso di un datapoint crittografato, è stato misurato il tempo necessario per la verifica del gettone, la determinazione della ricompensa corretta e l'invio della stessa al client. Per il datapoint baseline, è stato misurato soltanto il tempo di restituzione della ricompensa, poiché non sono presenti operazioni di verifica crittografica.

Per ciascun benchmark, è stato eseguito un test su quattro set di dati, inviando lo stesso dato al server per 50, 100, 200 e 400 iterazioni. Dopo ogni serie di invii, è stato calcolato il tempo totale di elaborazione per ciascun set di iterazioni, sia per la comunicazione baseline che per quella crittografata. Questo approccio ha permesso di osservare come l'overhead varia all'aumentare del numero di richieste inviate al server.

Ciascun benchmark è stato eseguito 20 volte.

Inoltre, è stata calcolata la differenza nominale tra una richiesta baseline (non crittografata) e una crittografata, così come tra una risposta baseline e una crittografata. Questa differenza misura l'aumento della dimensione dei dati trasmessi causato dall'applicazione della crittografia. La crittogra-

fia introduce un overhead aggiuntivo nelle dimensioni delle richieste e delle risposte, che può influire sul traffico di rete complessivo e sulle prestazioni. La misurazione della differenza nominale permette di quantificare quanto la crittografia aumenta la dimensione dei dati scambiati, fornendo un'indicazione chiara dell'impatto che la crittografia ha sul sistema in termini di utilizzo delle risorse di rete e tempi di risposta.

5.2 Analisi risultati

Poiché i test sono stati eseguiti utilizzando l'interfaccia di loopback (127.0.0.1), possiamo assumere che il delay di rete sia praticamente nullo. Questo significa che le variazioni nei tempi di esecuzione sono esclusivamente dovute all'overhead computazionale introdotto dalla crittografia, e non dall'eventuale latenza di trasmissione su rete. Di conseguenza, possiamo affermare che l'incremento nei tempi di esecuzione osservato per le richieste blindate è direttamente correlato all'utilizzo delle blind signatures.

Il grafico 5.1 mostra il tempo totale (in secondi) richiesto per completare diverse iterazioni (50, 100, 200, 400) sia per la comunicazione baseline (non crittografata) sia per quella crittografata (blindata) relativamente alla ricezione del gettone, con barre di errore che indicano la variabilità nei tempi misurati. I risultati dimostrano come l'overhead introdotto dalle blind signatures rimanga contenuto e sostenibile, anche in condizioni di carico crescente.

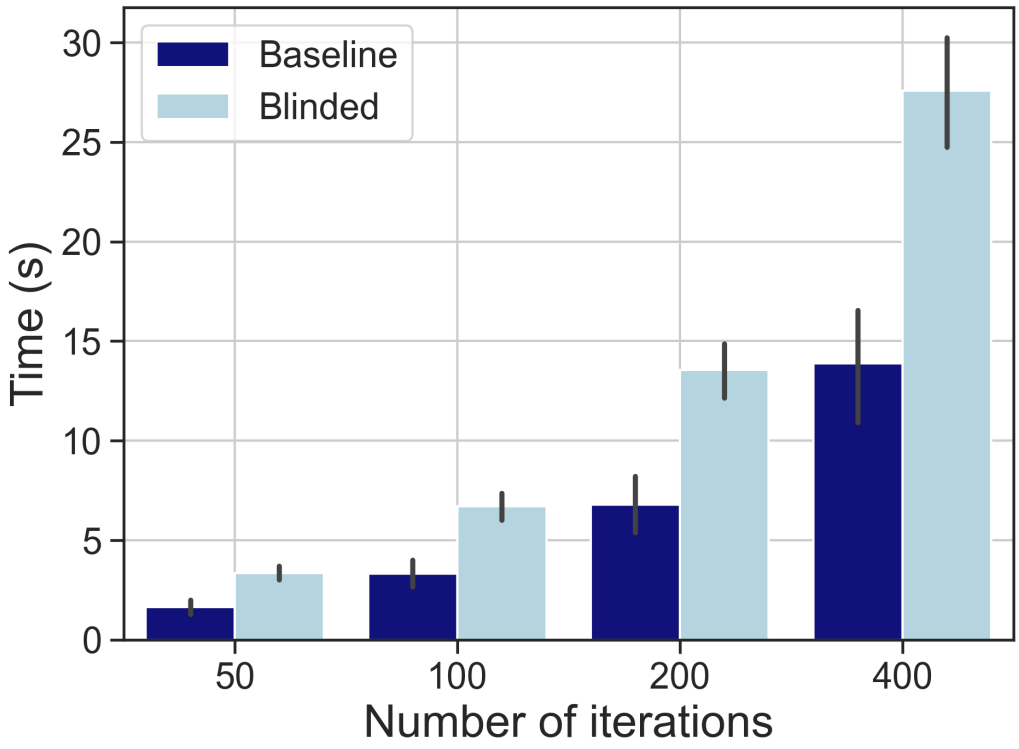

Figura 5.1: Tempo totale impiegato da ogni set di dati per ottenere i gettoni

Le comunicazioni baseline, rappresentate dalle barre blu scure, mostrano tempi di esecuzione relativamente bassi e con una crescita lineare. Questo è atteso, poiché l'assenza di crittografia rende il processo di elaborazione meno intensivo in termini di risorse computazionali.

Anche le comunicazioni blindate, rappresentate dalle barre azzurre chiare, presentano un aumento dei tempi di esecuzione all'aumentare delle iterazioni. Tuttavia, il confronto con la baseline mostra che, nonostante l'uso delle blind signatures comporti operazioni crittografiche aggiuntive, l'overhead rimane relativamente basso. Anche per 400 iterazioni, il tempo necessario per completare le richieste blindate non supera i 30 secondi, un valore accettabile in molti contesti operativi. Per numeri inferiori di iterazioni (50 e 100), il divario tra baseline e blindata è piuttosto ridotto, con differenze di pochi secondi. Questo dimostra che l'impatto della crittografia è minimizzato per

quantità moderate di richieste, il che rende il sistema efficiente e scalabile anche quando si adottano meccanismi di sicurezza avanzati.

Il secondo grafico 5.2 rappresenta il tempo totale impiegato da ogni set di dati per ottenere le ricompense.

I tempi di esecuzione per la comunicazione baseline (barre blu scuro) mostrano una crescita graduale e lineare. Anche in questo caso, la crescita è proporzionale al numero di iterazioni, indicando che il tempo di risposta senza crittografia rimane costante.

I tempi di esecuzione per la comunicazione crittografata (barre azzurro chiaro) sono più alti rispetto alla baseline. Tuttavia, anche in questo caso, l'incremento è contenuto e segue un andamento prevedibile all'aumentare delle iterazioni.

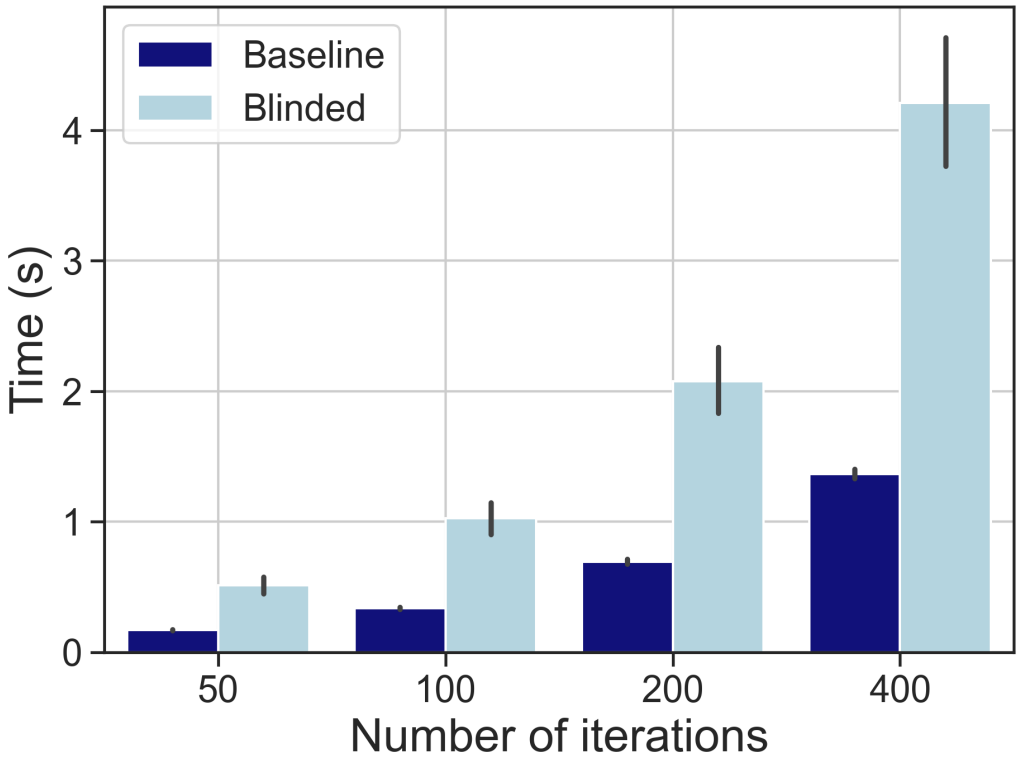

Figura 5.2: Tempo totale impiegato da ogni set di dati per ottenere le ricompense

Le barre di errore evidenziano una certa variabilità nei tempi di esecuzione, specialmente per le comunicazioni blindate, soprattutto nelle iterazioni più elevate (200 e 400). Questa variabilità è comunque limitata e rientra in un range accettabile, senza indicare un comportamento anomalo o inefficiente.

Come nel grafico precedente, possiamo osservare che l'overhead crittografico, rappresentato dall'incremento di tempo per le comunicazioni blindate, rimane contenuto anche per numeri elevati di iterazioni. Anche con 400 iterazioni, il tempo totale per la comunicazione blindata non supera i 4 secondi, un risultato che dimostra l'efficienza complessiva del sistema.

L'andamento relativamente lineare del tempo di esecuzione per entrambe le tipologie di comunicazione (baseline e blindata) indica che il sistema è in grado di gestire il carico aggiuntivo introdotto dalla crittografia senza un impatto significativo sulle prestazioni generali.

I risultati indicano chiaramente che l'implementazione delle blind signatures aggiunge un overhead contenuto e accettabile rispetto alla comunicazione baseline. Anche nelle condizioni più estreme (400 iterazioni), il tempo di esecuzione rimane entro limiti ragionevoli. Questo dimostra che è possibile integrare meccanismi di crittografia avanzati per garantire la privacy degli utenti, senza compromettere in modo significativo le prestazioni complessive del sistema.

Inoltre, per scenari di utilizzo più comuni, con meno di 200 iterazioni, l'overhead è quasi trascurabile, rendendo il sistema altamente scalabile e pronto per essere utilizzato in contesti reali, dove la sicurezza è una priorità ma le prestazioni devono rimanere elevate.

Capitolo 6

Applicazione Mobile per Android

Al fine di dimostrare l'applicabilità e l'usabilità della soluzione proposta, è stata progettata e sviluppata un'applicazione mobile. Questa applicazione permette di raccogliere i dati sensoristici dai dispositivi mobili degli utenti, gestire l'oscuramento e la firma dei token tramite firme cieche, e interagire con il server per la verifica e la ricezione delle ricompense. L'applicazione rappresenta un'implementazione pratica del sistema proposto, evidenziando la fattibilità della soluzione in un contesto reale.

6.1 Requisiti funzionali

L'applicazione SenseCollect è un tool funzionale che consente la raccolta di dati dai sensori dei dispositivi, tra cui il sensore di luce, accelerometro, giroscopio, la fotocamera e il microfono, utilizzato sia per registrare audio che per misurare il rumore ambientale.

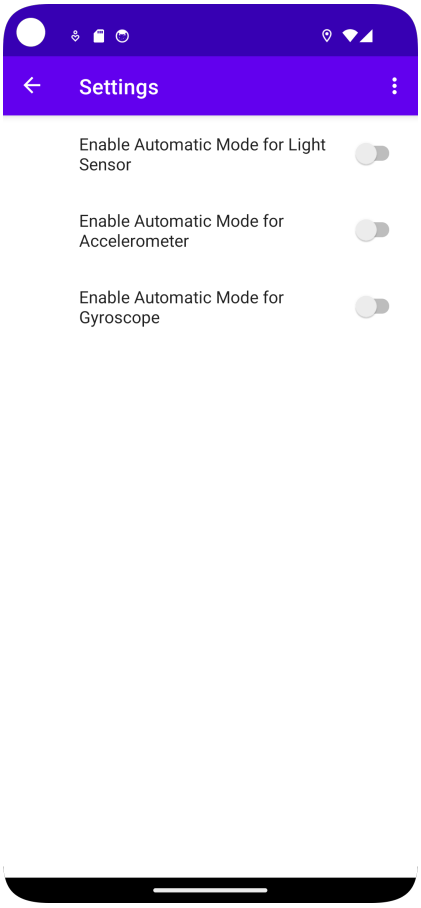

Figura 6.1: Impostazioni interne all'applicazione

I dati sensoristici possono essere raccolti manualmente, tramite l'attivazione dei sensori con un pulsante, oppure in modalità automatica. In questo caso la raccolta avviene anche quando l'applicazione è chiusa, attivando i sensori in background (solo dalle 8:00 alle 22:00, una volta all'ora).

La modalità con cui i sensori vengono attivati è decisa unicamente dall'utente, modificando le preferenze nelle impostazioni dell'applicazione.

Fotocamera e microfono sono invece utilizzabili solo in modalità manuale.

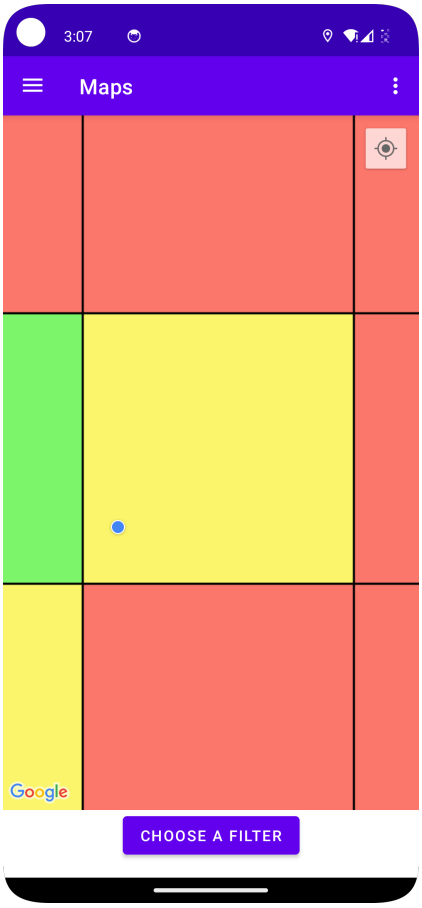

Figura 6.2: Map Tiles relative alla propria zona di rilevazione e a un sensore specifico

L'utente ha la possibilità di visualizzare, su una mappa, la quantità di dati raccolti per ciascun sensore nella propria zona di rilevazione e nelle zone adiacenti. Ogni zona di rilevazione corrisponde a una tile della mappa gestita tramite il Maps SDK per Android. Una tile è un riquadro della mappa che, a un determinato livello di zoom, rappresenta una porzione geografica specifica.

Le tile permettono di suddividere la mappa in griglie, e ogni tile può essere identificata in modo univoco in base alla sua posizione e livello di zoom. Quando l'utente seleziona un sensore (tramite una finestra di dialogo con scelta singola), la mappa mostra 9 riquadri colorati che rappresentano la propria zona di rilevazione e le 8 zone circostanti. Ogni riquadro viene

colorato in base alla quantità di dati raccolti per il sensore selezionato in quella zona.

Se colorato di verde indica che nella zona sono stati raccolti più di 100 dati per quel sensore, il giallo indica che la zona ha raccolto tra 50 e 100 dati, mentre il rosso indica che nella zona sono stati raccolti meno di 50 dati.

Questo sistema permette all'utente di avere una visione immediata della densità dei dati raccolti per ciascun sensore nella propria zona e nelle zone adiacenti, facilitando la comprensione delle aree dove sono già stati raccolti molti dati e quelle che necessitano di ulteriori rilevazioni.

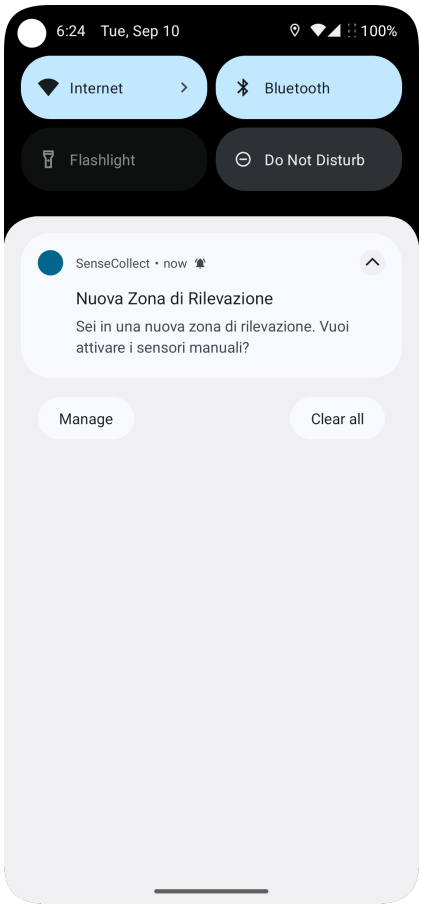

Figura 6.3: Notifica inviata all'uscita dal geofence

L'app fornisce un servizio di notifica che avvisa l'utente ogni volta che

esce dalla propria zona di rilevazione e al click su tale notifica viene aperta l'applicazione sulla schermata Home, quella coi bottoni utili all'attivazione dei sensori.

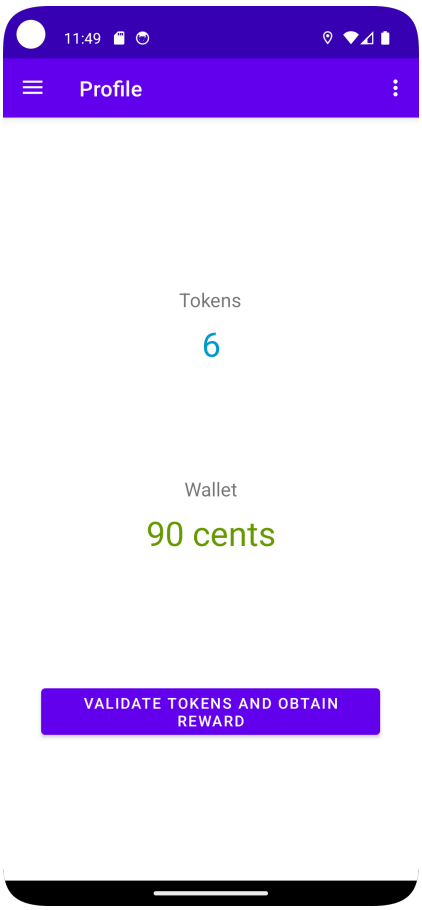

Figura 6.4: Profilo visualizzabile dall'utente

Inoltre, l'utente può accedere al proprio profilo, dove viene visualizzato il numero di gettoni accumulati in base ai dati raccolti e il proprio portafoglio. L'utente può quindi decidere quando validare i propri gettoni e ottenere la rispettiva ricompensa.

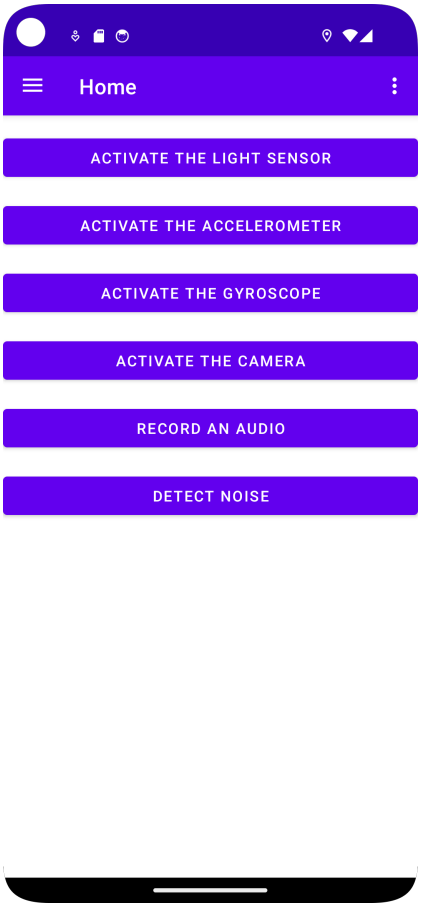

Figura 6.5: Schermata Home dell'applicazione

L'applicazione si interfaccia con una REST API Python ospitata su un server Flask, lo stesso presentato nel Capitolo 3. I dati sensoristici raccolti e i dati ottenuti grazie alle chiamate HTTP vengono salvati in un database locale.

L'app utilizza strumenti moderni per offrire un'esperienza fluida e ottimizzata.

6.2 Dettagli implementativi

Il linguaggio utilizzato per lo sviluppo dell'applicazione è Kotlin: moderno, sicuro e completamente interoperabile con Java, è la scelta ideale per le

applicazioni Android.
L'architettura utilizzata per separare logica di business e dati dalla logica di presentazione dell'interfaccia utente è l'architettura MVVM (Model-View-ViewModel). Le ViewModel gestiscono i dati e la logica aziendale, utilizzando LiveData per aggiornare dinamicamente l'interfaccia utente quando i dati cambiano.
L'istanza singleton del WorkManager è stata utilizzata per gestire task in background come la raccolta e l'upload dei dati dai sensori in modo automatico. WorkManager garantisce l'esecuzione affidabile dei task, anche se l'app viene chiusa o il dispositivo viene riavviato. Ogni sensore è associato ad un work unico e periodico, in modo da poter essere interrotto o messo in coda in base alle preferenze espresse dall'utente nelle impostazioni dell'applicazione.
Per visualizzare le mappe e i dati relativi alla posizione dell'utente e alle zone di rilevazione definite, sono state utilizzate le Google Maps API. L'applicazione rileva quando l'utente esce da una specifica area geografica grazie all'utilizzo delle Geofencing API, e invia notifiche in base a tale transizione.
Le chiamate API per inviare e recuperare dati in modo asincrono dalla REST API Python ospitata sul server Flask sono gestite da Retrofit, una libreria per effettuare chiamate HTTP.
La memorizzazione dei dati raccolti dai sensori e dei valori ottenuti dall'interazione con il server è gestito da Room, che permette di interagire con SQLite in maniera semplice e robusta.

Conclusioni e sviluppi futuri

In questa tesi, è stato proposto e sviluppato un nuovo sistema di Mobile Crowdsensing (MCS) basato sull'uso di meccanismi crittografici avanzati, con l'obiettivo di garantire la privacy degli utenti senza compromettere le prestazioni del sistema. Il cuore del sistema proposto risiede nell'applicazione delle blind signatures (firme cieche), un metodo crittografico che garantisce l'anonimato dei partecipanti durante il processo di raccolta e verifica dei dati.

I risultati ottenuti dai test condotti attraverso l'interfaccia di loopback hanno dimostrato che il sistema è in grado di bilanciare in modo efficace sicurezza e prestazioni. L'analisi dei benchmark ha mostrato che, sebbene l'uso delle blind signatures introduca un overhead computazionale, questo rimane contenuto e non compromette in modo significativo i tempi di risposta del sistema. Anche con un numero elevato di iterazioni, l'overhead rimane gestibile, dimostrando che il sistema proposto è in grado di scalare senza problemi per un'ampia gamma di applicazioni di crowdsensing. Anche in presenza di crittografia, i tempi di risposta restano contenuti e all'interno di limiti accettabili per un'applicazione di Mobile Crowdsensing che deve gestire grandi volumi di dati in tempo reale. Il sistema ha dimostrato di essere efficace nel preservare l'anonimato degli utenti, garantendo che i dati raccolti non possano essere ricondotti agli individui che li hanno forniti.

L'uso di meccanismi come le blind signatures si rivela quindi una soluzione valida per rispondere alle sfide sempre più pressanti legate alla sicurezza e alla protezione dei dati personali, specialmente in contesti dove la partecipazione massiva degli utenti è cruciale per la raccolta di informazioni. In aggiunta,

la possibilità di integrare questi meccanismi senza impatti significativi sulle prestazioni rende il sistema particolarmente adatto a essere utilizzato in ambienti reali, dove la scalabilità e l'efficienza sono fattori determinanti.

Sebbene il sistema proposto abbia dimostrato la sua validità, ci sono margini per ulteriori miglioramenti. Tra le possibili direzioni future c'è l'ulteriore ottimizzazione degli algoritmi crittografici, per ridurre ulteriormente l'overhead computazionale, l'integrazione con soluzioni multi-device e multi-server per rendere il sistema ancora più robusto e distribuito, la sperimentazione del sistema in scenari real-world, per valutarne le prestazioni in ambienti con condizioni di rete variabili e partecipanti eterogenei, e ancora l'integrazione di sistemi di reputazione, che potrebbe incentivare una partecipazione di qualità, premiando chi contribuisce in modo affidabile e costante, il supporto per nuove tipologie di sensori e contributi, aumentando così la flessibilità e l'applicabilità del protocollo in diversi ambiti, il design di un sistema per il recruiting esplicito degli utenti, ovvero un processo formale di invito e registrazione dei partecipanti, potrebbe essere utile in scenari dove è necessario assicurarsi una partecipazione attiva e consapevole.

In conclusione, il sistema di Mobile Crowdsensing proposto rappresenta un passo avanti nella direzione di una raccolta dati sicura e privacy-preserving, ponendo le basi per l'adozione di meccanismi crittografici avanzati in applicazioni di larga scala, dove la fiducia degli utenti è un elemento chiave per il successo.

Bibliografia

[1] A. Capponi, C. Fiandrino, B. Kantarci, L. Foschini, D. Kliazovich and P. Bouvry, "A Survey on Mobile Crowdsensing Systems: Challenges, Solutions, and Opportunities", in IEEE Communications Surveys & Tutorials, vol. 21, no. 3, pp. 2419-2465, thirdquarter 2019.

[2] R. K. Ganti, F. Ye and H. Lei, "Mobile crowdsensing: current state and future challenges", in IEEE Communications Magazine, vol. 49, no. 11, pp. 32-39, November 2011.

[3] Federico Montori, Prem Prakash Jayaraman, Ali Yavari, Alireza Hassani, Dimitrios Georgakopoulos, "The Curse of Sensing: Survey of techniques and challenges to cope with sparse and dense data in mobile crowd sensing for Internet of Things", Pervasive and Mobile Computing, Volume 49, 2018, Pages 111-125.

[4] L. Wang, D. Zhang, Y. Wang, C. Chen, X. Han and A. M'hamed, "Sparse mobile crowdsensing: challenges and opportunities," in IEEE Communications Magazine, vol. 54, no. 7, pp. 161-167, July 2016.

[5] Zhang, Zhong, Dae Hyun Yum, and Minho Shin. 2021. "PARS: Privacy-Aware Reward System for Mobile Crowdsensing Systems" Sensors 21, no. 21: 7045.

[6] Chaum, D. (1983). "Blind Signatures for Untraceable Payments." In: Chaum, D., Rivest, R.L., Sherman, A.T. (eds) Advances in Cryptology. Springer, Boston, MA.

[7] Pointcheval, D., Stern, J. "Security Arguments for Digital Signatures and Blind Signatures." J. Cryptology 13, 361-396 (2000).

[8] Goldwasser, S. and Bellare, M. "Lecture Notes on Cryptography" Archived 2012-04-21 at the Wayback Machine. Summer course on cryptography, MIT, 1996-2001.

[9] Bellare, M., Namprempre, C., Pointcheval, D. et al. "The One-More-RSA-Inversion Problems and the Security of Chaum's Blind Signature Scheme ". J. Cryptology 16, 185-215 (2003).

[10] S. Liu, Z. Wan, Y. Yuan, Q. Dong, B. Yang and F. Hao, "Efficient Certificateless Blind Signature Scheme With Conditional Revocation for Mobile Crowdsensing Within Smart City," in IEEE Internet of Things Journal, vol. 11, no. 9, pp. 15985-15997, 1 May, 2024..

[11] Yi, X., Lam, KY., Bertino, E., Rao, FY. (2019). "Location Privacy-Preserving Mobile Crowd Sensing with Anonymous Reputation". In: Sako, K., Schneider, S., Ryan, P. (eds) Computer Security – ESORICS 2019. ESORICS 2019. Lecture Notes in Computer Science(), vol 11736.