Alma Mater Studiorum · Università di Bologna

Dipartimento di Informatica — Scienza e Ingegneria

Corso di Laurea Triennale in Informatica per il Management

Implementazione di un protocollo basato su Blind Signatures per il Mobile Crowdsensing con ricompensa variabile

Relatore:
Prof.
FEDERICO MONTORI

Presentata da:
ANDREA CROCI

Sessione III

Anno Accademico 2024/2025

Sommario

Il Mobile Crowdsensing (MCS) rappresenta un paradigma emergente per la raccolta distribuita di dati attraverso dispositivi mobili appartenenti agli utenti. Grazie alla diffusione capillare degli smartphone, dotati di numerosi sensori integrati, è possibile ottenere grandi quantità di informazioni utili per diverse applicazioni, come il monitoraggio ambientale, l'analisi del traffico o la raccolta di dati urbani.

Uno dei principali problemi dei sistemi di Mobile Crowdsensing riguarda la gestione degli incentivi e la tutela della privacy degli utenti. Da un lato è necessario incentivare la partecipazione degli utenti attraverso meccanismi di ricompensa, dall'altro è fondamentale garantire che le informazioni personali degli utenti non possano essere facilmente tracciate o correlate alle contribuzioni inviate al sistema.

In questo lavoro viene progettato e implementato un protocollo basato su *blind signatures* per consentire la distribuzione di ricompense in maniera anonima. Il sistema consente agli utenti di ottenere token firmati dal server senza rivelare il contenuto del messaggio firmato, garantendo così proprietà di anonimato e unlinkability.

Sono state implementate e confrontate diverse configurazioni del sistema, basate su differenti architetture e primitive crittografiche, tra cui RSA, GDH e GDH con aggregazione. I risultati sperimentali mostrano l'impatto delle scelte architetturali e crittografiche sulle prestazioni del sistema e permettono di valutare i compromessi tra efficienza e proprietà di sicurezza.

Introduzione

Negli ultimi anni la diffusione degli smartphone e dei dispositivi mobili ha reso possibile la raccolta distribuita di grandi quantità di dati attraverso il contributo diretto degli utenti. Questo paradigma, noto come *Mobile Crowdsensing* (MCS), consente di sfruttare i sensori presenti nei dispositivi mobili per raccogliere informazioni utili in diversi contesti applicativi, come il monitoraggio ambientale, l'analisi del traffico urbano o la raccolta di dati relativi alla qualità dell'aria.

Nonostante le numerose potenzialità offerte da questo approccio, i sistemi di Mobile Crowdsensing presentano alcune sfide importanti. In particolare, uno degli aspetti più critici riguarda il coinvolgimento degli utenti e la protezione della loro privacy. Per incentivare la partecipazione è spesso necessario prevedere meccanismi di ricompensa, ma tali sistemi devono essere progettati in modo da evitare che i contributi degli utenti possano essere collegati alla loro identità.

Le *blind signatures* rappresentano una soluzione efficace per affrontare questo problema. Questo meccanismo crittografico consente a un server di firmare un messaggio senza conoscere il contenuto del messaggio stesso, permettendo quindi di distribuire token o ricevute firmate senza compromettere l'anonimato degli utenti.

L'obiettivo di questa tesi è progettare e implementare un protocollo basato su blind signatures per la gestione di ricompense anonime in un sistema di Mobile Crowdsensing. Il lavoro analizza diverse possibili configurazioni architetturali e crittografiche, confrontandone le prestazioni attraverso una

serie di benchmark sperimentali.

In particolare, sono state implementate due principali architetture del sistema: una configurazione *single-key* e una configurazione *multi-key*. Ciascuna architettura è stata combinata con tre diverse primitive crittografiche: RSA, GDH e GDH con aggregazione.

I risultati ottenuti permettono di analizzare l'impatto delle diverse scelte progettuali sulle prestazioni del sistema e di valutare i compromessi tra efficienza computazionale e proprietà di sicurezza offerte dal protocollo.

La tesi è organizzata come segue.

Il Capitolo 1 presenta lo stato dell'arte relativo al Mobile Crowdsensing e introduce i principali concetti crittografici necessari alla comprensione del lavoro, con particolare attenzione alle blind signatures e ai sistemi di incentivazione privacy-preserving.

Il Capitolo 2 descrive l'architettura del sistema proposto, illustrando le diverse configurazioni considerate. In particolare vengono presentate le architetture single-key e multi-key e le diverse varianti crittografiche basate su RSA, GDH e GDH con aggregazione.

Il Capitolo 3 è dedicato ai dettagli implementativi del sistema, descrivendo l'architettura client–server sviluppata, le tecnologie utilizzate e il funzionamento dei principali componenti del protocollo.

Il Capitolo 4 presenta i risultati sperimentali ottenuti attraverso una serie di benchmark progettati per analizzare le prestazioni del sistema e l'overhead introdotto dall'utilizzo delle primitive crittografiche.

Infine, il Capitolo 5 descrive l'applicazione mobile sviluppata come dimostrazione del funzionamento del sistema, mentre l'ultima sezione riporta le conclusioni del lavoro e possibili sviluppi futuri.

Indice

Elenco delle figure

Capitolo 1

Stato dell'arte

1.1 Mobile Crowdsensing

Negli ultimi anni, il Mobile Crowdsensing (MCS) si è affermato come uno dei paradigmi più promettenti nell'ambito della raccolta e analisi dei dati. Esso sfrutta le capacità di rilevamento e comunicazione integrate nei dispositivi mobili di uso quotidiano, come smartphone, tablet e dispositivi indossabili, per acquisire informazioni provenienti da un'ampia base di utenti. Questa caratteristica rende il MCS un approccio innovativo e sostenibile per la costruzione di smart cities e per la gestione intelligente di fenomeni complessi che coinvolgono la società contemporanea [1].

L'idea alla base del Mobile Crowdsensing è semplice ma potente: trasformare i cittadini in sensori mobili in grado di raccogliere dati sul proprio ambiente circostante. Ogni dispositivo, infatti, dispone di una varietà di sensori, come accelerometri, giroscopi, GPS, microfoni e fotocamere, che possono essere impiegati per monitorare parametri ambientali, condizioni del traffico, qualità dell'aria, infrastrutture urbane o persino aspetti legati alla salute pubblica. I dati raccolti vengono poi aggregati e analizzati in piattaforme cloud, permettendo di estrarre informazioni utili e fornire servizi orientati alle persone.

Il termine Mobile Crowdsensing è stato introdotto nel 2011 da Ganti et

al. come estensione del precedente concetto di mobile phone sensing. Rispetto a quest'ultimo, il MCS si distingue per il coinvolgimento attivo e diffuso di un insieme di utenti, la cosiddetta "folla" (crowd), che contribuisce alla creazione collettiva di conoscenza. Come definito da Guo et al., il MCS rappresenta un nuovo paradigma di rilevamento che consente ai cittadini comuni di contribuire con dati generati dai propri dispositivi mobili, aggregandoli e fondendoli nel cloud per estrarre intelligenza collettiva e fornire servizi centrati sulle persone [1].

L'architettura di un sistema MCS [1] può essere descritta come una struttura multilivello composta da quattro componenti principali:

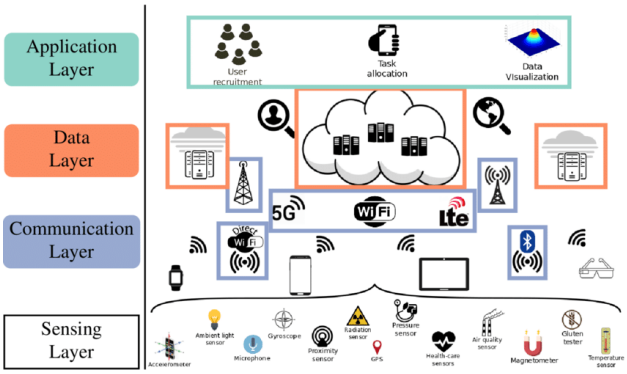

Figura 1.1: Architettura a livelli dei sistemi MCS

- **Livello Applicativo**: rappresenta il punto di interazione con l'utente finale. Comprende le funzionalità legate al reclutamento dei partecipanti, alla gestione delle campagne di raccolta dati e alla presentazione dei risultati ottenuti. Questo livello svolge un ruolo centrale nell'organizzazione e nel coordinamento delle attività di crowdsensing.

- **Livello Dati**: è deputato alla raccolta, memorizzazione, elaborazione e analisi delle informazioni acquisite dai dispositivi mobili. I dati provenienti dai partecipanti vengono aggregati e processati al fine di estrarre conoscenza utile. In tale contesto, aspetti quali integrità, affidabilità e accuratezza delle misurazioni rappresentano criticità ri-

levanti, soprattutto quando si utilizzano sensori eterogenei o a basso costo.

- **Livello di Trasmissione**: riguarda l'infrastruttura di comunicazione che consente l'invio dei dati dai dispositivi mobili verso i server centrali. Le tecnologie impiegate includono tipicamente reti cellulari, Wi-Fi e Bluetooth. La scelta del canale trasmissivo incide direttamente su parametri quali consumo energetico, latenza e affidabilità del sistema.

- **Livello di Rilevamento**: comprende l'insieme dei sensori fisici integrati nei dispositivi mobili, responsabili dell'acquisizione delle misurazioni. Tali sensori possono presentare limitazioni legate alla precisione, alla presenza di rumore nei segnali e all'accuratezza temporale delle rilevazioni.

Nonostante le potenzialità applicative del Mobile Crowdsensing, la letteratura evidenzia come la tutela della privacy rappresenti una delle principali criticità del paradigma. In particolare, nei sistemi di Opportunistic MCS, i partecipanti inviano automaticamente misurazioni geolocalizzate durante i propri spostamenti, spesso senza un'interazione esplicita con il sistema. Tale modalità operativa implica la trasmissione della posizione dell'utente insieme ai dati raccolti, rendendo possibile, attraverso tecniche di correlazione spazio-temporale o analisi inferenziale, la ricostruzione delle traiettorie individuali e dei pattern comportamentali.

La divulgazione della posizione precisa può generare gravi problemi di privacy, soprattutto in scenari in cui i partecipanti non sono consapevoli delle condizioni globali del sistema o della presenza di altri utenti [3].

In tale contesto emerge la necessità di meccanismi crittografici in grado di garantire anonimato e non tracciabilità, pur consentendo la verifica dei contributi e l'assegnazione corretta delle ricompense.

1.2 Blind Signature

Il concetto di *blind signature* è stato introdotto nel 1983 da David Chaum come strumento crittografico per la realizzazione di pagamenti elettronici non tracciabili [4]. Tale schema consente a un firmatario di apporre una firma digitale su un messaggio senza conoscerne il contenuto e senza poter successivamente associare la firma all'identità del richiedente.

L'idea alla base del meccanismo è il disaccoppiamento tra l'atto della firma e l'identità dell'utente: il firmatario produce una firma crittograficamente valida, pur non avendo accesso al messaggio originale né alla possibilità di collegare la firma finale all'interazione iniziale.

1.2.1 Principio di funzionamento

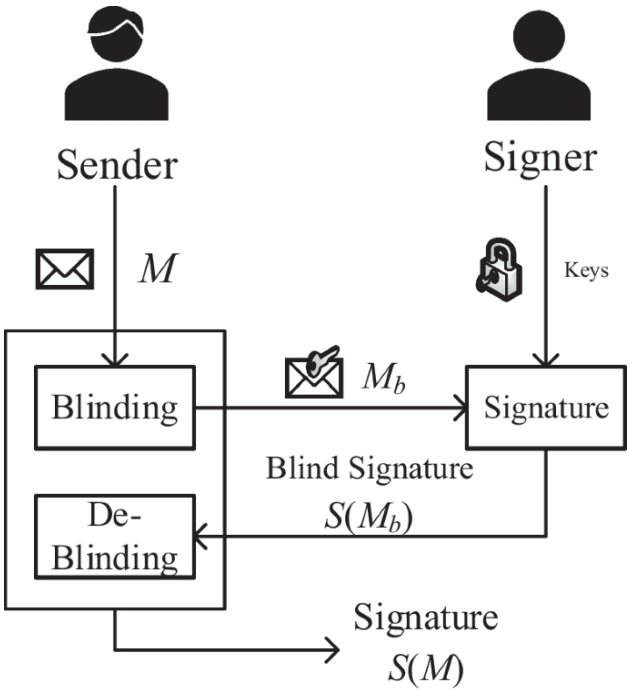

Figura 1.2: Processo di Blind Signature

Uno schema di blind signature coinvolge due entità:

- un **richiedente**, che desidera ottenere una firma su un messaggio m;
- un **firmatario**, che possiede una coppia di chiavi crittografiche e genera la firma.

Il protocollo si articola in tre fasi:

1. **Blinding**: il richiedente trasforma il messaggio m in una versione offuscata m' mediante un valore casuale segreto, detto *blinding factor*. In questo modo il contenuto effettivo del messaggio risulta nascosto.
2. **Firma**: il firmatario applica la propria chiave privata al messaggio offuscato m', producendo una firma su tale valore.
3. **Unblinding**: il richiedente rimuove il fattore di offuscamento dalla firma ricevuta, ottenendo una firma valida sul messaggio originale m.

La proprietà essenziale dello schema è che il firmatario non può stabilire alcun legame tra la firma finale e la richiesta di firma effettuata, garantendo così la *non tracciabilità* (*unlinkability*).

1.2.2 Blind Signature basata su RSA

Una realizzazione classica delle blind signatures può essere costruita a partire dallo schema di firma RSA, la cui sicurezza si basa sulla difficoltà computazionale del problema della fattorizzazione e sull'ipotesi RSA [5].

Nel sistema RSA, il firmatario genera una coppia di chiavi come segue: si scelgono due numeri primi grandi p e q, si calcola $n = pq$ e la funzione di Eulero $\varphi(n) = (p-1)(q-1)$. Si seleziona quindi un esponente pubblico e tale che $\gcd(e, \varphi(n)) = 1$ e si determina l'esponente privato d come inverso moltiplicativo di e modulo $\varphi(n)$, ovvero

$$ed \equiv 1 \pmod{\varphi(n)}.$$

La chiave pubblica è (n, e), mentre la chiave privata è d.

Nel caso di una firma RSA standard, data una rappresentazione numerica del messaggio $m \in \mathbb{Z}_n$, la firma è calcolata come:

$$s = m^d \mod n,$$

e la verifica consiste nel controllare che:

$$s^e \equiv m \pmod{n}.$$

Per ottenere una versione cieca della firma, il richiedente introduce un fattore casuale $r \in \mathbb{Z}_n^*$, scelto uniformemente e tale che $\gcd(r, n) = 1$. Il messaggio viene quindi offuscato mediante la trasformazione:

$$m' = m \cdot r^e \mod n.$$

Il firmatario applica la propria chiave privata al valore ricevuto:

$$s' = (m')^d \mod n.$$

Sfruttando le proprietà dell'aritmetica modulare e la relazione $ed \equiv 1 \pmod{\varphi(n)}$, si ottiene:

$$s' = (m \cdot r^e)^d \equiv m^d \cdot r^{ed} \equiv m^d \cdot r \pmod{n}.$$

Il richiedente può quindi rimuovere il fattore casuale calcolando:

$$s = s' \cdot r^{-1} \mod n,$$

ottenendo:

$$s \equiv m^d \pmod{n},$$

che corrisponde esattamente a una firma RSA valida sul messaggio originale m.

La correttezza del protocollo deriva direttamente dalla struttura algebrica di RSA, mentre la proprietà di cecità dipende dal fatto che il firmatario

osserva esclusivamente il valore m', il quale, per la casualità di r, risulta computazionalmente indistinguibile da un elemento casuale di $\mathbb{Z}_n^*$ sotto l'ipotesi RSA.

Di conseguenza, il firmatario non è in grado di ricostruire né il messaggio originale né di stabilire un collegamento tra la fase di firma e l'eventuale utilizzo successivo della firma stessa.

1.2.3 Rilevanza per il Mobile Crowdsensing

Sebbene introdotte nel contesto dei pagamenti elettronici, le blind signatures risultano particolarmente adatte anche ai sistemi di incentivazione.

Nel Mobile Crowdsensing è infatti necessario garantire che un partecipante possa ottenere una ricompensa per il contributo fornito, evitando al contempo che il server possa risalire alla sua identità o tracciare le sue interazioni nel tempo. Si configura quindi una tensione tra anonimato e verificabilità.

Le blind signatures permettono di emettere token o voucher crittograficamente validi e verificabili, senza esporre l'identità del richiedente né consentire la correlazione tra fase di emissione e fase di utilizzo. Per questo motivo rappresentano uno strumento centrale nella progettazione di sistemi MCS orientati alla tutela della privacy.

1.3 Lavori Correlati su MCS e Blind Signature

Numerosi lavori hanno proposto meccanismi di incentivazione privacy-preserving per i sistemi di Mobile Crowdsensing (MCS), adottando differenti tecniche crittografiche quali cifratura, coordinamento tra partecipanti, firme di gruppo, firme ad anello, valuta digitale e blind signature.

1.3.1 Approcci basati su cifratura

Alcuni lavori utilizzano principalmente tecniche di cifratura per proteggere la privacy dei partecipanti. Un esempio è rappresentato dal sistema di

incentivazione basato su aste, in cui i task di crowdsensing vengono pubblicati sotto forma di aste. I partecipanti inviano offerte e profili di sensing cifrati utilizzando la chiave pubblica dell'ente che gestisce l'asta. Sebbene le informazioni relative alle offerte e ai pagamenti siano cifrate, la piattaforma mantiene la conoscenza delle identità dei vincitori e dell'ammontare dei pagamenti, risultando potenzialmente vulnerabile ad attacchi di inferenza.

In modo analogo, Zhang et al. [8] propongono un sistema in cui i dati vengono inoltrati iterativamente attraverso partecipanti vicini prima di raggiungere il server centrale. Il meccanismo di selezione dei vincitori avviene in più round e include tecniche di mescolamento (shuffling) e utilizzo di indirizzi IP differenti per ridurre la linkabilità. Tuttavia, il server rimane responsabile della distribuzione delle ricompense e può sfruttare le informazioni sui pagamenti per collegare contributi e premi. Inoltre, il sistema risulta vulnerabile ad attacchi di collusione e non affronta esplicitamente la proprietà di non ripudio.

1.3.2 Approcci basati su valuta digitale e pseudonimi

Il sistema E-cent [12] introduce una valuta digitale per garantire l'anonimato dei partecipanti. Gli utenti operano utilizzando pseudonimi differenti all'interno di una mix-zone e possono cambiarli periodicamente. Le monete digitali vengono firmate dal server e marcate con numeri segreti, rendendo difficile collegare una specifica transazione al partecipante. Tuttavia, poiché il server gestisce il pagamento delle ricompense, è comunque possibile stabilire un collegamento tra contributo e premio. Inoltre, il livello di anonimato dipende dal numero di partecipanti presenti nella mix-zone, rendendo il sistema vulnerabile in caso di collusione.

1.3.3 Firme di gruppo e firme ad anello

Altri lavori adottano firme di gruppo per garantire anonimato e accountability. Gisdakis et al. [11] propongono un sistema in cui il partecipante

può ottenere ricompense utilizzando uno pseudonimo temporaneo rilasciato da una Pseudonym Certification Authority. I task vengono firmati mediante firma di gruppo, impedendo il collegamento tra identità e dati raccolti. Tuttavia, in presenza di collusione tra entità fidate, l'anonimato può essere compromesso.

Li et al. [9] propongono CreditCoin, un sistema basato su blockchain che utilizza firme ad anello per garantire anonimato nelle reti veicolari. Per firmare un messaggio è necessario il contributo di almeno k partecipanti, rendendo difficile identificare il firmatario effettivo. Nonostante ciò, il sistema richiede un'autorità fidata che mantenga la corrispondenza tra indirizzi e identità reali, introducendo un potenziale punto di vulnerabilità in caso di collusione.

1.3.4 Blind signature nei sistemi MCS

Diversi ricercatori hanno adottato le blind signature per garantire unlinkability, non falsificabilità e protezione contro attacchi di linking [9, 10].

Li e Cao [9, 10] propongono un sistema basato su crediti in cui vengono utilizzate firme parzialmente blind per prevenire attacchi di linking. I partecipanti registrano token associati alla propria identità reale, ma l'emissione e l'utilizzo dei crediti avvengono mediante meccanismi di accecamento. Sebbene il data collector non possa collegare direttamente report e ricevute, esso gestisce comunque gli account dei partecipanti, rendendo potenzialmente possibile associare ricompense a task specifici. Inoltre, il protocollo richiede numerose interazioni tra partecipante e data collector, aumentando l'overhead comunicativo.

Dimitriou [7] propone un meccanismo che combina token anonimi, zero-knowledge proof e firme parzialmente blind. Il partecipante utilizza un identificativo segreto e genera commitment per ottenere firme blind sui token. Le ricompense possono essere aggregate su un unico token aggiornabile. Il sistema garantisce che le ricompense non siano collegabili allo stesso partecipante e resiste a determinati attacchi di collusione. Tuttavia, l'adozione congiunta

di più primitive crittografiche rende il sistema complesso da implementare e non viene considerata la separazione tra ente che emette la ricompensa e ente che la riscatta.

1.3.5 PARS: Privacy-Aware Reward System

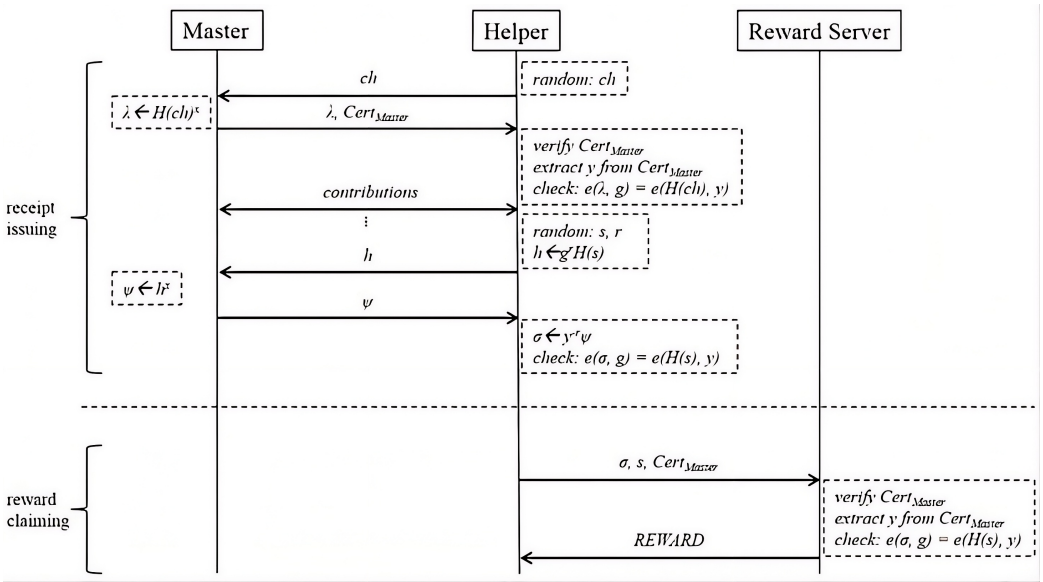

Figura 1.3: Schema: PARS

Un contributo particolarmente rilevante nell'ambito dei sistemi di incentivazione privacy-preserving per MCS è PARS (Privacy-Aware Reward System) [6]. PARS formalizza un modello di rewarding a tre entità composto da: (i) il *Master* ($\mathcal{M}$), che crea e diffonde il task e raccoglie i report; (ii) l'*Helper* ($\mathcal{H}$), che esegue il task e restituisce il contributo; (iii) il *Reward Service* ($\mathcal{RS}$), che verifica il contributo ed eroga la ricompensa. È inoltre prevista una *Certificate Authority* (CA), responsabile della certificazione delle chiavi pubbliche dei Master.

Il flusso operativo segue una struttura ben definita: il Master genera un task e lo invia all'Helper; l'Helper esegue il task e restituisce il risultato; a

seguito della ricezione del contributo, il Master rilascia una ricevuta crittografica che attesta l'avvenuta partecipazione; infine, l'Helper presenta tale ricevuta al $\mathcal{RS}$ per ottenere la ricompensa.

Lo schema di ricompensa viene formalizzato tramite quattro algoritmi fondamentali: Kg (generazione delle chiavi), Helper, Master e Vrfy. L'algoritmo $\mathsf{Kg}(1^k)$ genera una coppia di chiavi (sk, vk), rispettivamente chiave privata per l'emissione delle ricevute e chiave pubblica per la verifica. L'emissione della ricevuta avviene attraverso un protocollo interattivo tra Master e Helper, al termine del quale l'Helper ottiene una ricevuta σ valida ma non collegabile direttamente al contributo specifico. La verifica è eseguita dal $\mathcal{RS}$ mediante l'algoritmo deterministico $\mathsf{Vrfy}(vk, s, \sigma, L)$, che controlla sia l'autenticità crittografica sia l'assenza di double-spending.

Dal punto di vista crittografico, PARS si basa su uno schema di *blind signature* costruito su gruppi Gap Diffie–Hellman (GDH), in cui il problema Computational Diffie–Hellman (CDH) è difficile mentre il Decisional Diffie–Hellman (DDH) è efficiente da verificare. L'utilizzo di una mappa bilineare consente inoltre l'aggregazione di più ricevute in una singola ricevuta di lunghezza costante, migliorando l'efficienza del sistema.

Il meccanismo di blind signature garantisce che il Master firmi un valore blindato ($h = g^r \cdot H(s)$) senza conoscere il numero seriale effettivo s scelto dall'Helper. Dopo la fase di "unblinding", l'Helper ottiene una ricevuta σ verificabile pubblicamente tramite la relazione bilineare:

$$e(\sigma, g) = e(H(s), y),$$

senza che sia possibile collegare la ricevuta al contributo originario o all'identità dell'Helper.

PARS definisce inoltre un chiaro modello di minaccia che include: (i) tentativi del Reward Service di inferire informazioni aggiuntive sui contributi (privacy leakage), (ii) possibilità di collusione tra Master e $\mathcal{RS}$, (iii) tentativi dell'Helper di ottenere ricompense indebite o multiple. Le proprietà di sicurezza perseguite includono proof-to-contribution unlinkability,

contribution-to-contribution unlinkability, non ripudio e non falsificabilità delle ricevute.

Rispetto ad altri approcci basati su firme di gruppo o firme ad anello, PARS fornisce un formalismo crittografico rigoroso fondato su assunzioni standard (chosen-target CDH) e un modello esplicito di sicurezza. Tuttavia, il sistema introduce una struttura relativamente complessa e presuppone una forte fiducia nel corretto funzionamento del Master e della CA. Inoltre, la separazione tra ente emittente della ricevuta e ente erogatore della ricompensa, pur modellata logicamente, non sempre riflette scenari applicativi in cui tali entità risultano pienamente indipendenti.

Un ulteriore aspetto rilevante di PARS è il supporto all'aggregazione delle ricevute. Grazie alla struttura dello schema di firma basato su gruppi GDH con mappa bilineare, più ricevute valide, ottenute anche da Master distinti, possono essere aggregate in un'unica ricevuta di lunghezza costante.

In pratica, l'Helper può presentare al $\mathcal{RS}$ un insieme di ricevute aggregate invece di inviarle singolarmente. Il Reward Service è quindi in grado di effettuare una sola operazione di verifica crittografica per validare l'intero insieme delle contribuzioni.

Questo meccanismo consente di ridurre il numero di verifiche bilineari necessarie e di velocizzare significativamente il processo di ottenimento della ricompensa, migliorando l'efficienza complessiva del sistema, specialmente in scenari in cui l'Helper accumula più contributi prima di richiedere il rewarding.

1.4 Limiti degli approcci esistenti e motivazione del lavoro

Dall'analisi dei lavori presenti in letteratura emerge come le blind signatures rappresentino uno strumento efficace per realizzare sistemi di incentivazione privacy-preserving nel contesto del Mobile Crowdsensing. In particolare, il sistema PARS introduce un modello formale di rewarding basato su ricevute

crittografiche ottenute tramite blind signature su gruppi Gap Diffie–Hellman (GDH), garantendo proprietà di unlinkability tra contributo e ricompensa.

Tuttavia, PARS si concentra principalmente sulla definizione teorica del protocollo e sull'analisi delle proprietà di sicurezza, adottando una specifica primitiva crittografica basata su gruppi bilineari. In questo contesto non viene esplorato il confronto con altre possibili soluzioni crittografiche che potrebbero essere utilizzate per realizzare meccanismi analoghi.

Inoltre, il protocollo considerato prevede l'emissione di una ricevuta per ogni interazione tra Helper e Master. In uno scenario implementativo reale, può risultare utile consentire al partecipante di ottenere la firma di più token all'interno della stessa interazione, permettendo anche di supportare meccanismi di ricompensa più flessibili, in cui il server può decidere di firmare un numero variabile di token in base alle caratteristiche del contributo fornito.

Alla luce di queste osservazioni, il presente lavoro si propone di progettare e implementare un prototipo di sistema di incentivazione privacy-preserving per Mobile Crowdsensing che consenta di analizzare sperimentalmente l'impatto di diverse scelte crittografiche e architetturali.

In particolare, vengono implementate tre varianti principali basate su blind signatures:

- una soluzione basata su RSA;
- una soluzione basata su GDH;
- una soluzione basata su GDH con aggregazione delle receipt.

L'obiettivo è confrontare tali soluzioni dal punto di vista delle prestazioni computazionali, valutando il costo delle operazioni crittografiche coinvolte e osservando le differenze tra approcci basati su firme RSA e su schemi pairing-based.

Inoltre, l'implementazione sviluppata consente di richiedere la firma di più token all'interno della stessa interazione tra client e server, permettendo di studiare l'impatto di tale scelta sull'efficienza del sistema.

Capitolo 2

Architettura

2.1 Introduzione

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.

L'idea di fondo è quella di impedire che il server possa collegare direttamente un determinato datapoint fornito dall'utente al successivo utilizzo del token di rewarding. Per raggiungere questo obiettivo, il contributo viene certificato mediante un meccanismo crittografico che consente al server di attestare la validità della partecipazione senza conoscere il contenuto finale del token che sarà poi speso dall'utente.

L'intero studio è stato sviluppato considerando sei sistemi distinti, ottenuti combinando due diverse architetture, *single-key* e *multi-key*, con tre differenti varianti crittografiche:

- single-key RSA;
- single-key GDH;
- single-key GDH con aggregazione;
- multi-key RSA;

- multi-key GDH;
- multi-key GDH con aggregazione.

Tali sistemi condividono lo stesso obiettivo generale, ma differiscono per il numero di chiavi utilizzate dal server, per il meccanismo con cui viene emesso il token e, nel caso delle varianti con aggregazione, per la modalità con cui avviene la fase finale di verifica della ricompensa.

2.2 Architettura single-key

Nell'architettura single-key il server utilizza una sola chiave crittografica per l'emissione dei token o delle receipt associate ai contributi forniti dagli utenti.

Il principio rimane quello già discusso nel capitolo precedente: il partecipante esegue un task di Mobile Crowdsensing, invia il relativo contributo al server e ottiene in cambio una certificazione crittografica blindata. In un secondo momento, tale certificazione può essere utilizzata per riscattare la ricompensa, senza che il server sia in grado di collegare in modo diretto la fase di emissione del token alla fase di spesa dello stesso.

Dal punto di vista applicativo, questa architettura è stata pensata come una piattaforma in cui l'utente può selezionare una campagna di raccolta dati e fornire un contributo sfruttando i sensori del proprio dispositivo. La reward dipende da parametri specifici della submission. Ad esempio, il livello di precisione del GPS utilizzato durante la raccolta può influenzare il valore del premio, poiché una misura più precisa può risultare maggiormente utile per il sistema. In maniera analoga, anche il tipo di task selezionato inciderà sulla ricompensa finale.

Pur variando la primitiva crittografica adottata, le tre soluzioni single-key mantengono la stessa struttura generale: il client prepara un valore blindato, il server lo certifica applicando la propria chiave segreta e il client, una volta rimosso il blindaggio, ottiene un token spendibile in una fase successiva.

2.2.1 Flusso di interazione client-server

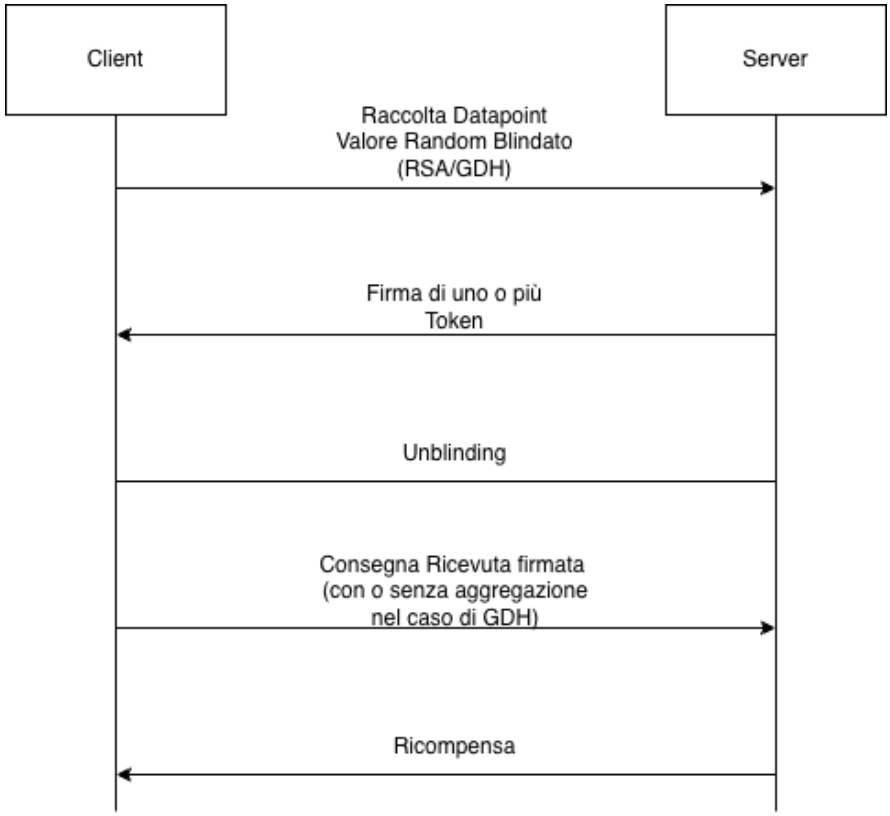

Figura 2.1: Flusso del protocollo nell'architettura single-key

Il flusso di interazione tra client e server nel modello single-key può essere riassunto nei seguenti passaggi:

1. il client richiede al server la lista delle campagne attive;
2. l'utente seleziona un task;
3. il client raccoglie i dati richiesti dal task, comprensivi di informazioni GPS ed eventualmente di contenuti multimediali;
4. il client invia al server il contributo raccolto;
5. il client genera uno o più token blindati e ne richiede la firma;

6. il server certifica i token secondo le regole previste dal protocollo;
7. il client rimuove il blindaggio dai token ricevuti;
8. in una fase successiva, i token validi vengono presentati al server per ottenere la reward.

Questa struttura rimane sostanzialmente invariata nelle tre varianti considerate. Ciò che cambia è il modo in cui vengono realizzate le operazioni di blindaggio, emissione, unblinding e verifica.

2.2.2 Single-key RSA

La prima variante considerata è il sistema *single-key RSA*. In questo caso il meccanismo di emissione del token si basa su blind signatures RSA.

Il client genera uno o più messaggi casuali, li offusca mediante la chiave pubblica del server e invia i valori blindati insieme al contributo raccolto. Il server, dopo aver ricevuto la submission, firma un sottoinsieme dei token blindati utilizzando la propria chiave privata. Il client può quindi rimuovere il blindaggio e ottenere token validi, spendibili in un secondo momento.

Inoltre, come verrà discusso nel capitolo 5, per tale architettura è stata sviluppata anche un'applicazione mobile dimostrativa, con lo scopo di osservare il comportamento del sistema in uno scenario d'uso più realistico e vicino a una possibile applicazione concreta.

2.2.3 Single-key GDH

La seconda variante è il sistema *single-key GDH*. In questo caso l'architettura di base rimane invariata, ma la blind signature RSA viene sostituita da un meccanismo pairing-based ispirato allo schema PARS descritto nel Capitolo 1.

Il token non è più rappresentato da una firma RSA su un messaggio blindato, bensì da una receipt ottenuta tramite operazioni in gruppi bilineari. Anche in questa configurazione il client prepara il materiale crittografico

blindato, il server applica la propria chiave segreta al valore ricevuto e il client rimuove successivamente il blindaggio, ottenendo una receipt valida.

La verifica del token avviene successivamente mediante una procedura pairing-based. Dal punto di vista logico, il comportamento del sistema non cambia: il server certifica l'avvenuto contributo senza conoscere il token finale che verrà poi utilizzato nella fase di rewarding.

L'interesse principale di questa variante risiede nella possibilità di confrontare due famiglie crittografiche differenti, RSA e GDH, mantenendo però invariata la struttura complessiva del sistema. In questo modo è possibile osservare con maggiore chiarezza l'impatto della primitiva crittografica sulle prestazioni.

2.2.4 Single-key GDH con aggregazione

La terza variante dell'architettura single-key è il sistema *single-key GDH con aggregazione*. Tale soluzione estende il modello precedente introducendo un meccanismo di aggregazione delle receipt, anch'esso ispirato a PARS.

Nelle varianti precedenti, ogni token o receipt viene verificato singolarmente al momento della riscossione della ricompensa. In questo caso, invece, le receipt ottenute dal client possono essere accumulate durante l'esecuzione di un blocco e successivamente aggregate in un unico oggetto crittografico.

Nel contesto di questo lavoro, l'aggregazione viene eseguita al termine di ciascun blocco di benchmark. Il client raccoglie quindi tutte le receipt valide prodotte nel blocco corrente, le combina localmente e invia al server una sola richiesta di verifica aggregata. Il server verifica l'autenticità dell'insieme e controlla che i seriali coinvolti non siano già stati spesi.

Questa soluzione è particolarmente interessante perché consente di studiare il vantaggio dell'aggregazione nella fase di verifica, riducendo il numero di richieste inviate al server ed il costo complessivo della fase di riscossione.

2.3 Architettura multi-key

Nel modello *multi-key* il server utilizza più coppie di chiavi crittografiche, invece di una sola. L'obiettivo è associare il contributo dell'utente a diverse classi di ricompensa, mantenendo separati il momento della raccolta del dato e quello della riscossione del premio.

La reward viene determinata tramite una *policy* applicativa basata su due fattori:

- la tipologia di dato raccolto;
- la quantità di dati dello stesso tipo già presenti nella medesima *tile*.

Per distinguere i contributi più rari da quelli già abbondanti, è stata introdotta una soglia pari a 50 contributi. Combinando il criterio quantitativo con quello qualitativo si ottengono sei classi di reward: `low_many`, `medium_many`, `high_many`, `low_few`, `medium_few` e `high_few`.

Dal punto di vista architetturale, il client genera un unico messaggio casuale e ne costruisce una versione blindata per ciascuna chiave pubblica disponibile sul server. Tutti i token blindati vengono inviati insieme al datapoint. Il server analizza il contributo, determina la classe di reward corrispondente e, tramite una mappatura deterministica, seleziona la chiave da utilizzare per la firma o per l'emissione della receipt.

L'elemento centrale del modello *multi-key* è quindi che il server non certifica tutti i token ricevuti, ma solo quello associato alla chiave selezionata dalla policy. Il client, conoscendo il fattore di blindaggio relativo a ciascuna chiave, può rimuovere correttamente l'offuscamento solo dal token indicato dal server e ottenere così il gettone valido per la successiva fase di verifica e riscossione della ricompensa.

2.3.1 Flusso di interazione client-server

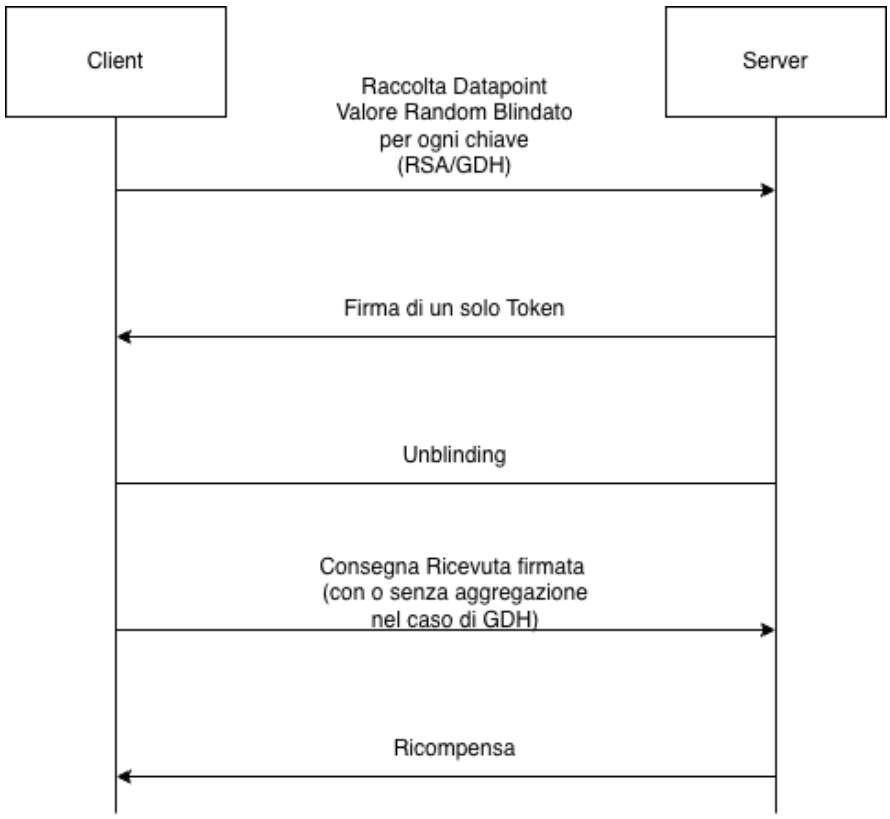

Figura 2.2: Flusso del protocollo nell'architettura multi-key

1. il client raccoglie il datapoint sensoriale, insieme alle informazioni contestuali rilevanti, come tipo di dato, tile e timestamp;
2. il client genera un messaggio casuale che costituirà la base del token;
3. il client crea una versione blindata dello stesso messaggio per ciascuna delle chiavi pubbliche disponibili;
4. il client invia al server il datapoint e l'insieme dei token blindati;
5. il server analizza il contributo e determina la classe di reward tramite la propria policy;

6. il server seleziona la chiave corrispondente e certifica soltanto il token blindato associato a quella chiave;

7. il server restituisce al client il token firmato, insieme all'identificativo della chiave utilizzata;

8. il client rimuove il blindaggio corretto usando il fattore casuale associato alla chiave selezionata;

9. in una fase successiva, il token viene presentato al server per la verifica e per l'ottenimento della ricompensa.

2.3.2 Varianti crittografiche dell'architettura multi-key

Le varianti crittografiche del modello multi-key seguono lo stesso principio descritto per l'architettura single-key. La differenza principale è che il client genera una versione blindata del messaggio per ciascuna chiave pubblica disponibile sul server, mentre quest'ultimo seleziona la chiave da utilizzare in base alla classe di reward determinata dalla policy.

Nel sistema *multi-key RSA* la certificazione del token avviene tramite blind signatures RSA. Nel sistema *multi-key GDH* la stessa architettura viene mantenuta, ma la firma RSA è sostituita da una receipt pairing-based ispirata a PARS.

Infine, nel sistema *multi-key GDH con aggregazione*, oltre alla struttura multi-key viene introdotto un meccanismo di aggregazione delle receipt, che consente di verificare con un'unica operazione più token ottenuti durante un blocco di esecuzione. In questo caso la reward finale corrisponde alla somma delle ricompense associate ai contributi inclusi nell'aggregato.

Capitolo 3

Implementazione

In questo capitolo viene presentata l'implementazione della soluzione proposta, disponibile al seguente link: `https://github.com/andreacrox/Tesi_Croci`

L'intero progetto è stato sviluppato in linguaggio Python secondo un'architettura client-server.

Il sistema è composto da due elementi principali:

- un **Server Flask**, responsabile della gestione dei datapoint, della firma o emissione delle receipt e della verifica finale dei token;
- un **Client**, che simula il comportamento dell'utente, raccogliendo il dato, generando i token blindati e interagendo con il server per ottenere la ricompensa.

Per la memorizzazione dei dati è stato utilizzato un database non relazionale **MongoDB**, impiegato per salvare i datapoint ricevuti e i token già utilizzati, così da prevenire fenomeni di double spending.

Come descritto nel capitolo precedente, sono stati implementati sei sistemi differenti, ottenuti combinando le architetture *single-key* e *multi-key* con tre varianti crittografiche: RSA, GDH e GDH con aggregazione. Dal punto di vista implementativo, tali sistemi condividono gran parte della struttu-

ra applicativa; le differenze principali riguardano il meccanismo di firma e verifica dei token.

3.1 Implementazione dei sistemi basati su RSA

Il sistema *single-key RSA* costituisce la base dell'intero lavoro e rappresenta la variante più semplice dal punto di vista architetturale e implementativo. In questo modello il server genera una sola coppia di chiavi RSA e la utilizza per firmare i token blindati ricevuti dal client.

Nel server, la generazione o il caricamento della chiave RSA persistente avviene come segue:

```
def load_or_create_rsa(path: str) -> RSA.RsaKey:
    if os.path.exists(path):
        with open(path, "rb") as f:
            return RSA.import_key(f.read())
    key = RSA.generate(2048)
    with open(path, "wb") as f:
        f.write(key.export_key("PEM"))
    return key
```

La firma di un token blindato viene poi effettuata con una semplice operazione di elevamento a potenza modulare:

```
def sign_blind(blinded: int) -> int:
    return pow(blinded, priv.d, priv.n)
```

Il client, una volta ottenuta la chiave pubblica dal server, genera un messaggio casuale e lo offusca tramite il meccanismo delle blind signatures. La funzione di blindaggio è la seguente:

```
def blind(message: int, pub: dict):
    N = int(pub["n"])
```

```
    E = int(pub["e"])

    while True:
        r = random.randint(2, N - 2)
        if gcd(r, N) == 1:
            break

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

Dopo aver ricevuto dal server il token firmato, il client rimuove il blindaggio utilizzando il valore casuale r scelto in precedenza:

```
def unblind(signature: int, r: int, N: int):
    return (signature * inverse(r, N)) % N
```

La verifica del token da parte del server avviene ancora tramite RSA, controllando che il messaggio ottenuto dalla firma coincida con il messaggio originario:

```
def verify_sig(sig: int, msg: int) -> bool:
    return (msg % pub.n) == pow(sig, pub.e, pub.n)
```

Se il token risulta valido e non è già stato utilizzato, il server assegna la ricompensa all'utente e memorizza l'identificativo del token nel database, così da impedirne il riutilizzo.

Questo sistema è stato utilizzato anche come base per lo sviluppo dell'applicazione mobile, che verrà descritta nel Capitolo 5. Per tale motivo esso rappresenta il riferimento principale sia dal punto di vista architetturale sia dal punto di vista implementativo.

Il sistema multi-key RSA, invece, riutilizza esattamente le stesse operazioni crittografiche descritte in questa sezione. La differenza principale risiede nella gestione di più coppie di chiavi lato server.

In questo caso il client genera una versione blindata del messaggio per ciascuna chiave pubblica disponibile, mentre il server seleziona la chiave da utilizzare in base alla policy di rewarding e firma esclusivamente il token corrispondente.

3.2 Implementazione dei sistemi basati su GDH

Le varianti basate su GDH mantengono la stessa logica generale del sistema RSA, ma sostituiscono la firma cieca con un meccanismo pairing-based ispirato a PARS. In questo caso non si lavora più con interi modulo N, ma con punti di curva ellittica sulla curva BLS12-381, utilizzando la libreria `py_ecc`.

Un primo elemento caratteristico dell'implementazione è la conversione del seriale in un punto del gruppo tramite una funzione di hash-to-curve:

```
def hash_to_point(serial_str: str):
    return hash_to_G2(serial_str.encode("utf-8"), DST, hashlib.sha256)
```

Nel sistema GDH, il blindaggio non consiste più in una moltiplicazione modulare, ma nella combinazione tra il punto hashato e una componente casuale:

```
def blind(serial_str: str):
    r = random_scalar()
    hs = hash_to_point(serial_str)
    blinded = add(hs, multiply(G2, r))
    return blinded, r
```

Il server applica quindi la propria chiave segreta al punto blindato ricevuto:

```
def blind_sign(h_point):
    return multiply(h_point, x)
```

Il client rimuove successivamente il blindaggio ottenendo la receipt finale:

```
def unblind(psi, r):
    return add(psi, neg(multiply(_Y2, r)))
```

La verifica della receipt avviene tramite pairing, secondo una relazione del tipo:

$$e(\sigma, g) = e(H(s), y)$$

che nel codice è implementata come:

```
def verify_receipt(sig_point, serial_str: str) -> bool:
    hs = hash_to_point(serial_str)
    return pairing(sig_point, G1) == pairing(hs, Y1)
```

Dal punto di vista software, il comportamento del sistema resta quindi analogo a quello RSA: il client genera un token blindato, il server lo certifica e il client lo spende in un secondo momento. Ciò che cambia è la primitiva crittografica sottostante e, di conseguenza, il costo computazionale delle operazioni coinvolte.

3.3 Implementazione dell'aggregazione

Nei sistemi GDH con aggregazione, la struttura generale rimane invariata fino alla fase di ottenimento delle receipt. La differenza principale riguarda la verifica finale: invece di verificare ogni receipt singolarmente, il client accumula le receipt ottenute durante un blocco di esecuzione e le aggrega in un unico oggetto crittografico.

L'aggregazione viene implementata come somma dei punti di curva corrispondenti alle singole receipt:

```
def aggregate_points(points):
    agg = None
    for p in points:
        agg = p if agg is None else add(agg, p)
    return agg
```

Nel caso *single-key*, tutte le receipt del blocco sono associate alla stessa chiave pubblica, per cui il client aggrega localmente le firme e invia al server una sola richiesta di verifica. Il server controlla l'intero aggregato confrontando il pairing della receipt aggregata con il prodotto dei pairing dei singoli messaggi:

```
def verify_aggregate(sig_agg, serials) -> bool:
    lhs = pairing(sig_agg, G1)

    rhs = None
    for s in serials:
        term = pairing(hash_to_point(str(s)), Y1)
        rhs = term if rhs is None else rhs * term

    return lhs == rhs
```

Nel caso *multi-key*, la logica è analoga, ma la verifica deve tenere conto del fatto che le diverse receipt possono essere state emesse con chiavi differenti. Per questo motivo, insieme alla receipt aggregata, il client trasmette al server anche i seriali e gli identificativi delle chiavi utilizzate.

L'introduzione dell'aggregazione non modifica quindi la fase di emissione dei token, ma interviene esclusivamente nella fase finale di verifica e riscossione della reward. Questo consente di ridurre il numero di controlli necessari lato server e di osservare sperimentalmente l'impatto di tale tecnica sulle prestazioni complessive del sistema.

3.4 Comunicazione client-server

La comunicazione tra client e server avviene tramite richieste HTTP in formato JSON. In tutte le varianti implementate, il client invia al server un datapoint e l'insieme dei token blindati da certificare. Un esempio semplificato di richiesta è il seguente:

```
requestBody = {
    "datapoint": {
        "geolocation": "0.0 , 0.0",
        "datatype": "Light Sensor",
        "value": "0.0",
        "timestamp": "01/01/1990",
        "tile": {"x": "0", "y": "0", "z": "0"}
    },
    "tokens": tokens
}
```

Nel caso dei sistemi baseline, il client invia invece direttamente il datapoint e un identificativo, ottenendo in risposta il valore della reward senza ricorrere al meccanismo di blind signature o di receipt blindata.

Gli endpoint implementati sul server consentono di separare chiaramente:

- la fase di ottenimento delle chiavi pubbliche;
- la fase di emissione della firma o della receipt;
- la fase di verifica finale e di assegnazione della ricompensa;
- la fase di benchmark crypto-only, utilizzata per isolare il solo costo delle primitive crittografiche.

Capitolo 4

Risultati ottenuti

In questo capitolo vengono presentati i risultati sperimentali ottenuti dall'implementazione del protocollo proposto. L'obiettivo dei test è valutare le prestazioni del sistema e analizzare l'overhead introdotto dall'utilizzo delle blind signatures rispetto ad una comunicazione baseline priva di meccanismi crittografici.

Inoltre, i benchmark sono stati progettati per confrontare le diverse configurazioni architetturali e crittografiche implementate nel sistema.

L'analisi dei risultati permette quindi di valutare il comportamento del protocollo al variare della primitiva crittografica utilizzata, della struttura architetturale del sistema e del numero massimo di token o chiavi gestite.

4.1 Setup sperimentale

Gli esperimenti sono stati eseguiti su un computer portatile **MacBook Pro Retina 13-inch (2019)** dotato di processore **Intel Core i5 dual-core a 1.6 GHz**, **8 GB di memoria RAM** e sistema operativo **macOS Sonoma 14.6.1**.

Tutti i test sono stati eseguiti localmente utilizzando l'interfaccia di loopback `127.0.0.1`. Questa configurazione permette di eliminare l'influenza

della latenza di rete e di misurare esclusivamente il costo computazionale delle operazioni crittografiche e delle operazioni applicative del protocollo.

Per analizzare il comportamento del sistema al variare del carico di lavoro sono stati eseguiti benchmark con diversi numeri di iterazioni del protocollo:

- 50 iterazioni
- 100 iterazioni
- 200 iterazioni
- 400 iterazioni

Ogni benchmark è stato ripetuto **20 volte**, permettendo di calcolare valori medi e ridurre l'impatto della variabilità delle misurazioni.

Durante i test sono state misurate due metriche principali:

- **tot_time_token**: tempo necessario per ottenere il token firmato dal server;
- **tot_time_reward**: tempo necessario per verificare il token e ottenere la ricompensa.

I benchmark sono stati eseguiti confrontando due modalità operative:

- **Baseline**, in cui il datapoint viene inviato al server senza utilizzare meccanismi crittografici;
- **Blinded**, in cui il client utilizza blind signatures per ottenere un token firmato dal server senza rivelare il contenuto del messaggio.

Questo confronto consente di quantificare l'overhead introdotto dalla crittografia.

4.2 Confronto tra comunicazione baseline e blind

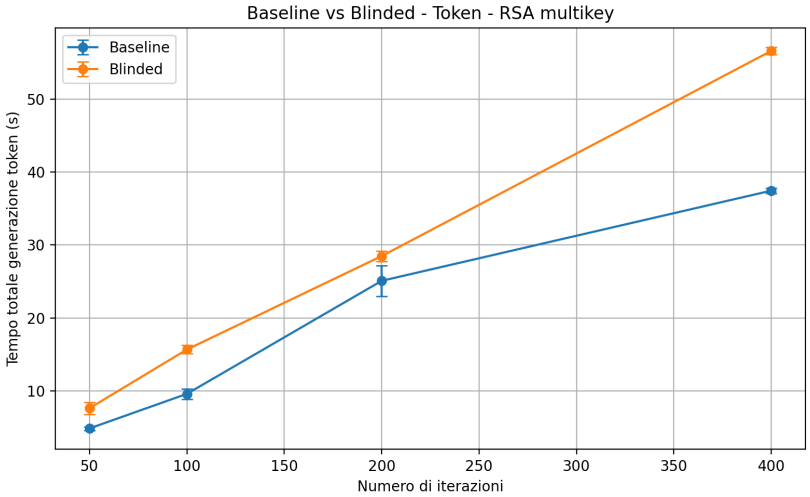

Figura 4.1: Confronto tra comunicazione baseline e blindata nella fase di generazione del token per il sistema multi-key basato su RSA

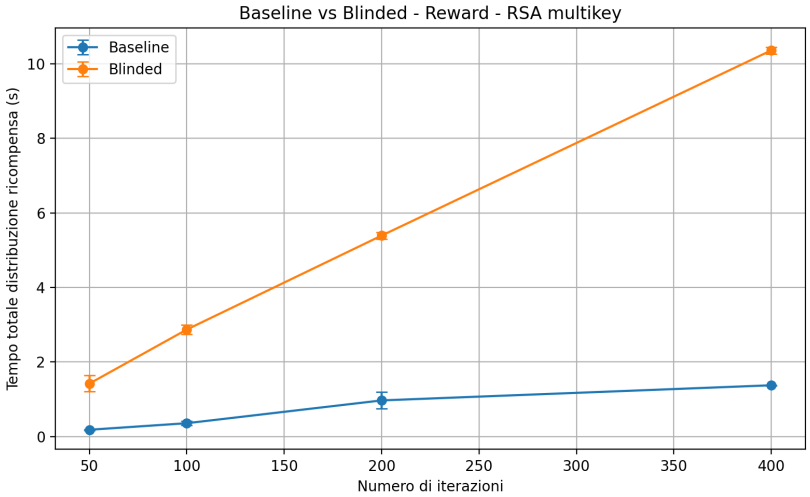

Figura 4.2: Confronto tra comunicazione baseline e comunicazione blindata nella fase di distribuzione della ricompensa per il sistema multi-key basato su RSA

Il primo obiettivo dei benchmark è stato quello di valutare l'impatto delle blind signatures sulle prestazioni complessive del sistema. I risultati mostrano chiaramente che l'introduzione della crittografia comporta un aumento del tempo di esecuzione rispetto alla comunicazione baseline. Questo incremento è dovuto principalmente alle operazioni di blindatura del messaggio, firma da parte del server e successiva sblindatura del token da parte del client. Tuttavia, l'entità dell'overhead varia significativamente in funzione dello schema crittografico utilizzato.

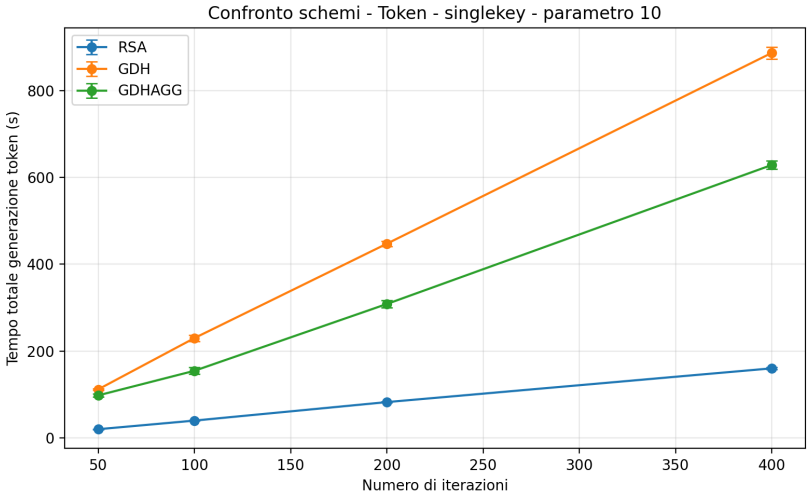

Figura 4.3: Confronto tra gli schemi crittografici RSA, GDH e GDH con aggregazione nell'architettura single-key. Il grafico mostra il tempo necessario per ottenere il token blindato al variare del numero di iterazioni.

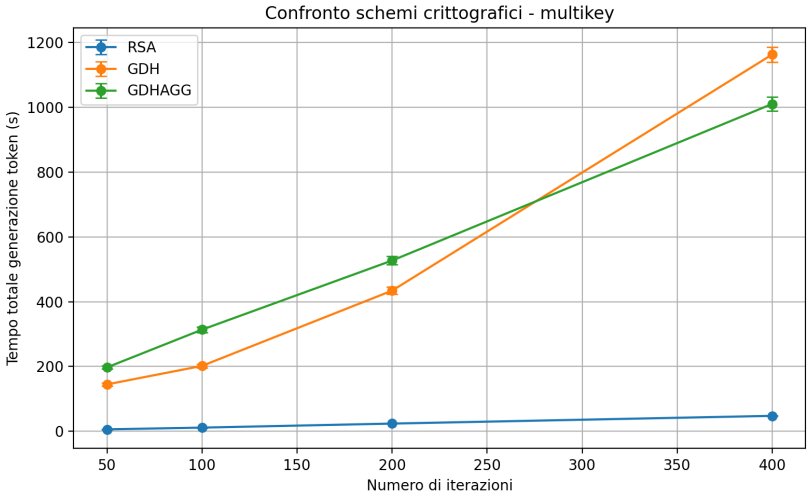

Figura 4.4: Confronto tra gli schemi crittografici RSA, GDH e GDH con aggregazione nell'architettura multi-key. Il grafico rappresenta il tempo necessario per ottenere il token blindato al variare del numero di iterazioni.

Nel caso degli schemi basati su **RSA**, l'overhead risulta relativamente contenuto. Nei sistemi multi-key basati su RSA, l'overhead medio per la generazione dei token risulta essere circa **1.4 volte** rispetto alla baseline, mentre nella fase di distribuzione delle ricompense l'overhead è di circa **7 volte**. Questo indica che l'utilizzo di blind signatures RSA introduce un costo computazionale moderato.

Al contrario, gli schemi basati su **GDH** presentano un costo computazionale significativamente maggiore. Nei benchmark eseguiti l'overhead medio per la generazione dei token supera **260 volte** rispetto alla baseline, mentre nella fase di ricompensa può superare **1200 volte**. Questo comportamento è dovuto principalmente al costo computazionale delle operazioni di pairing su curve ellittiche utilizzate negli schemi GDH.

Lo schema **GDH con aggregazione** mostra prestazioni intermedie. L'aggregazione delle ricevute consente infatti di ridurre il numero di operazioni crittografiche necessarie, diminuendo il costo complessivo rispetto allo schema GDH puro.

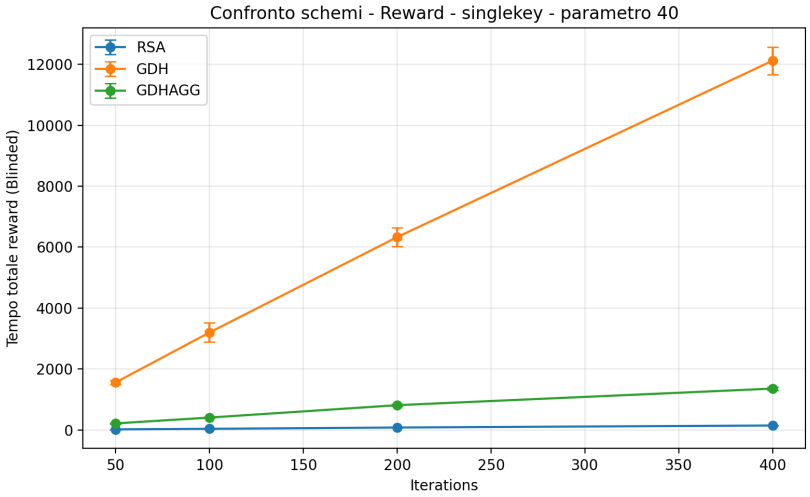

Figura 4.5: Confronto tra gli schemi crittografici RSA, GDH e GDH con aggregazione nell'architettura single-key (parametro 40) per la fase di distribuzione delle ricompense.

Dall'analisi del grafico relativo alla fase di distribuzione delle ricompense nell'architettura single-key con parametro pari a 40 emerge una differenza significativa tra i tre schemi crittografici considerati.

Lo schema basato su RSA risulta il più efficiente, mantenendo tempi di esecuzione sensibilmente inferiori rispetto alle soluzioni basate su GDH. Questo risultato è coerente con il diverso costo computazionale delle operazioni coinvolte: mentre RSA si basa principalmente su operazioni di esponenziazione modulare, gli schemi GDH richiedono operazioni di pairing su curve ellittiche, che risultano più onerose.

Un risultato interessante riguarda la variante GDH con aggregazione. Come evidenziato dal grafico, l'introduzione dell'aggregazione consente di ridurre significativamente il tempo necessario per la fase di reward rispetto allo schema GDH puro. In particolare, per valori elevati del numero di iterazioni, il divario tra GDH e GDH con aggregazione si riduce in maniera sostanziale.

Sebbene le soluzioni basate su RSA risultino comunque le più efficienti nel prototipo implementato, questi risultati mostrano come l'aggregazione

possa contribuire a mitigare il costo computazionale degli schemi pairing-based, rendendoli sensibilmente più performanti rispetto alla versione GDH non aggregata.

4.3 Confronto tra architetture single-key e multi-key

Un ulteriore aspetto analizzato riguarda il confronto tra le due architetture proposte: **single-key** e **multi-key**.

I risultati mostrano che le due architetture presentano caratteristiche differenti in termini di prestazioni, dovute principalmente alla diversa struttura del protocollo e al numero di operazioni crittografiche richieste durante l'esecuzione.

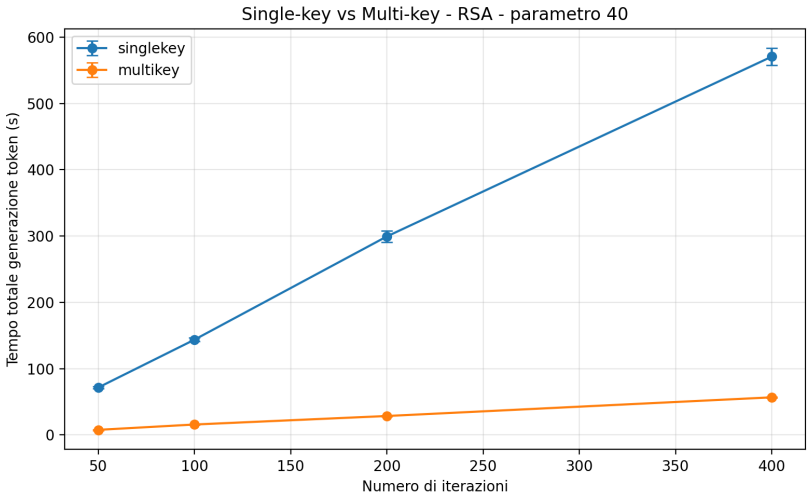

Figura 4.6: Confronto tra architettura single-key e multi-key utilizzando lo schema RSA con parametro pari a 40 in caso di single-key. Il grafico mostra il tempo necessario per ottenere il token blindato al variare del numero di iterazioni.

Dal grafico si osserva che l'architettura **single-key** richiede generalmente un tempo maggiore rispetto all'architettura **multi-key**. Questo comporta-

mento è legato principalmente al numero di operazioni crittografiche eseguite durante il protocollo.

Nel modello single-key il client genera più token blindati che vengono inviati al server per essere firmati. Di conseguenza il server deve eseguire più operazioni di firma RSA con la chiave privata, che rappresentano la componente computazionalmente più costosa dell'intero protocollo. Il tempo di esecuzione dipende inoltre dal numero di token generati dal client, che è regolato dal parametro del sistema: al crescere di questo parametro aumenta il numero di operazioni di firma necessarie e quindi il tempo complessivo richiesto.

Nel modello multi-key, invece, ogni richiesta coinvolge la firma di un singolo token. Il server seleziona una chiave tra quelle disponibili e utilizza tale chiave per firmare il token corrispondente, riducendo il numero complessivo di operazioni crittografiche eseguite in ogni interazione. Questo spiega perché, nel grafico, l'architettura multi-key risulti più efficiente nella fase di generazione dei token.

Le operazioni che incidono maggiormente sul tempo di esecuzione sono quelle di firma RSA con chiave privata, mentre le operazioni di blinding, unblinding e verifica della firma presentano un costo computazionale inferiore.

Un ulteriore aspetto riguarda il meccanismo di distribuzione delle ricompense. Nell'architettura single-key il client può ottenere la firma di più token all'interno della stessa interazione, che possono essere successivamente utilizzati per riscattare più ricompense. Questo consente di implementare meccanismi di reward variabile basati sul numero di token firmati.

Nel modello multi-key, invece, la ricompensa è determinata dalla chiave utilizzata per firmare il token. Ogni chiave è associata a una specifica classe di ricompensa e ogni richiesta coinvolge la firma di un singolo token. Questo approccio consente di controllare il valore della ricompensa attraverso la selezione della chiave appropriata.

Dal punto di vista architetturale emerge quindi un compromesso tra le

due soluzioni. L'architettura single-key risulta più semplice da gestire poiché richiede l'utilizzo di una sola chiave crittografica e permette una maggiore flessibilità nel meccanismo di ricompensa, ma risulta meno efficiente quando il numero di token generati aumenta. L'architettura multi-key introduce una maggiore complessità nella gestione delle chiavi, ma consente di ridurre il numero di operazioni crittografiche necessarie per ogni richiesta e di migliorare le prestazioni complessive del sistema.

In termini applicativi, la scelta tra le due architetture dipende quindi dalle caratteristiche del sistema di crowdsensing considerato. L'approccio single-key può risultare preferibile in scenari in cui è necessario mantenere una maggiore flessibilità nel meccanismo di incentivazione, poiché consente al client di ottenere la firma di più token e quindi di riscattare ricompense differenti in base ai contributi forniti.

Al contrario, l'architettura multi-key può risultare più adatta in contesti in cui l'obiettivo principale è l'efficienza del sistema e la riduzione del costo computazionale delle operazioni crittografiche. In questo modello, infatti, ogni richiesta comporta generalmente la firma di un singolo token, permettendo di limitare il numero di operazioni crittografiche e migliorare le prestazioni complessive.

4.4 Scalabilità del sistema

I benchmark permettono inoltre di analizzare la scalabilità del sistema al variare del parametro che controlla il numero massimo di token (nell'architettura single-key) o il numero massimo di chiavi (nell'architettura multi-key).

Nel caso dell'architettura single-key si osserva una crescita quasi lineare del tempo di esecuzione all'aumentare del numero massimo di token. Ad esempio, nel sistema RSA single-key il tempo medio per ottenere un token passa da circa **7.9s** con parametro 1 a oltre **270s** con parametro 40.

Un comportamento simile si osserva anche negli schemi basati su GDH e GDH con aggregazione, dove l'aumento del numero di token comporta un aumento significativo del tempo di esecuzione.

Nel caso dell'architettura multi-key l'impatto del parametro sulle prestazioni risulta più limitato, soprattutto nel caso di RSA. Questo suggerisce che l'architettura multi-key risulta più scalabile quando il numero di chiavi disponibili aumenta.

4.5 Discussione dei risultati

Nel complesso, i risultati sperimentali mostrano che il protocollo proposto è in grado di garantire proprietà di privacy avanzate introducendo un overhead computazionale variabile a seconda della configurazione scelta.

Gli schemi basati su **RSA** risultano i più efficienti dal punto di vista computazionale e presentano un overhead relativamente contenuto, rendendoli particolarmente adatti per scenari reali di Mobile Crowdsensing.

Gli schemi basati su **GDH** offrono proprietà crittografiche più avanzate ma presentano costi computazionali significativamente più elevati a causa dell'utilizzo di operazioni di pairing.

L'introduzione dell'**aggregazione** permette di ridurre parzialmente questo costo, rendendo lo schema GDH con aggregazione più efficiente rispetto alla versione base.

Infine, il confronto tra architettura single-key e multi-key evidenzia come le due soluzioni presentino diversi compromessi tra prestazioni e scalabilità, offrendo diverse possibilità di configurazione a seconda delle esigenze del sistema.

Capitolo 5

Applicazione Mobile per Android

Al fine di dimostrare l'applicabilità e l'usabilità della soluzione proposta nei capitoli precedenti, è stata progettata e sviluppata un'applicazione mobile per dispositivi Android che implementa il protocollo di rewarding basato su blind signatures nel contesto di un sistema di Mobile Crowdsensing. L'applicazione consente agli utenti di partecipare a campagne di Mobile Crowdsensing utilizzando il proprio smartphone, raccogliendo dati sensoristici o contenuti multimediali e inviandoli a un backend remoto.

L'app rappresenta un'implementazione pratica del sistema descritto nei capitoli precedenti, permettendo di osservare il funzionamento dell'architettura proposta in uno scenario realistico. In particolare, l'applicazione permette agli utenti di visualizzare le campagne di crowdsensing attive, selezionare un task e completarlo tramite l'acquisizione di dati dal dispositivo.

Dal punto di vista architetturale, l'applicazione segue un modello *client–server*. Il client Android si occupa dell'interazione con l'utente, della raccolta dei dati e della gestione del flusso di partecipazione alle campagne, mentre il backend gestisce la logica applicativa, la memorizzazione delle submission e il sistema di ricompensa.

Un aspetto centrale dell'applicazione è la tutela della privacy dell'utente. Il sistema non richiede alcuna registrazione né associa le submission a un'identità persistente. Inoltre, l'utente può controllare il livello di preci-

sione della posizione GPS condivisa con il backend, scegliendo il numero di decimali delle coordinate da inviare.

Il sistema integra inoltre un meccanismo di ricompensa anonimo basato su *blind signature*. Dopo aver completato un task, l'utente può ottenere token digitali firmati dal server, che possono essere successivamente utilizzati per ottenere una ricompensa senza che il backend possa collegare i token alla submission originale.

5.1 Requisiti funzionali

L'applicazione Android sviluppata consente agli utenti di partecipare a campagne di Mobile Crowdsensing eseguendo task che richiedono la raccolta di dati dal dispositivo mobile.

In particolare, l'app supporta diverse tipologie di task, tra cui:

- acquisizione di fotografie tramite la fotocamera del dispositivo;
- registrazione di video;
- registrazione di audio;
- raccolta di dati dall'accelerometro;
- raccolta di dati dal giroscopio;
- raccolta di dati dal sensore di luminosità;
- stima del livello sonoro tramite registrazione audio.

L'utente può visualizzare una lista delle campagne di crowdsensing attive direttamente dalla schermata principale dell'applicazione. Ogni campagna viene mostrata sotto forma di una card contenente una breve descrizione del task richiesto, il tipo di dato da raccogliere e il numero massimo di token che possono essere ottenuti completando l'attività.

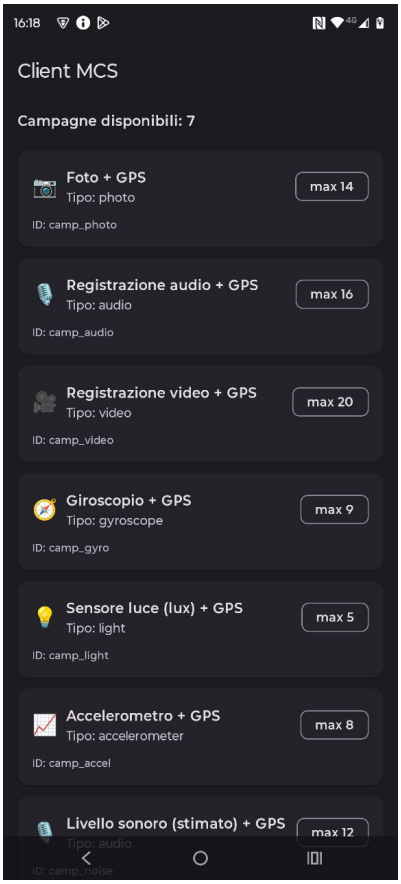

Figura 5.1: Schermata principale dell'applicazione con la lista delle campagne di crowdsensing disponibili.

Quando l'utente seleziona una campagna, l'applicazione apre una schermata dedicata all'esecuzione del task. In questa fase l'utente viene guidato nell'acquisizione dei dati richiesti, che possono consistere in letture dei sensori oppure in contenuti multimediali.

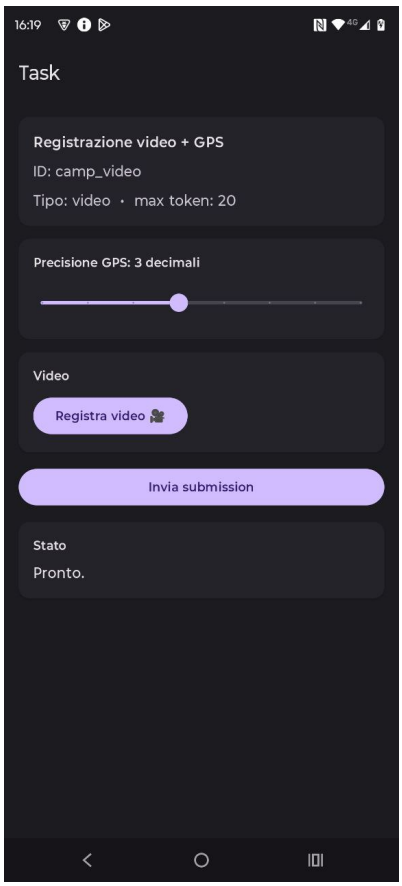

Figura 5.2: Schermata di esecuzione del task aperta dopo la selezione di una campagna.

Prima dell'esecuzione del task, l'utente può configurare il livello di precisione della posizione GPS tramite uno *slider* presente nell'interfaccia dell'applicazione. Questo meccanismo consente di scegliere consapevolmente il compromesso tra precisione della posizione e tutela della privacy. Riducendo il numero di decimali delle coordinate GPS, l'utente può condividere una posizione approssimata anziché quella esatta.

Il livello di precisione selezionato ha inoltre un impatto sul sistema di ricompensa: una maggiore precisione della posizione può consentire al backend di firmare un numero maggiore di token.

Una volta completato il task, l'utente può inviare la submission al bac-

kend. L'applicazione raccoglie quindi i dati acquisiti insieme alla posizione GPS mascherata e li invia al server tramite una richiesta REST. Nel caso di task multimediali, i file vengono caricati separatamente tramite richieste *multipart*.

Dopo l'invio della submission, l'applicazione avvia il meccanismo di ricompensa basato su *blind signature*. Il client genera localmente un insieme di token casuali, applica il processo di *blinding* e invia i token blindati al server per la firma. Il backend firma un sottoinsieme dei token in base alle regole definite e restituisce le firme al client, che provvede all'*unblinding* e alla successiva spesa dei token.

Al termine del processo, l'utente riceve un feedback che riassume il risultato dell'operazione, indicando il numero di token richiesti, il numero di token firmati e la ricompensa totale ottenuta.

Per migliorare l'esperienza d'uso, l'applicazione integra inoltre un sistema di notifiche che informa l'utente della disponibilità di nuove campagne di crowdsensing. Questo meccanismo è implementato tramite `WorkManager`, che permette di eseguire controlli periodici in background interrogando il backend.

5.2 Dettagli implementativi

L'applicazione è stata sviluppata in **Kotlin**, linguaggio moderno e completamente interoperabile con Java, largamente utilizzato nello sviluppo di applicazioni Android.

Per la realizzazione dell'interfaccia grafica è stato utilizzato **Jetpack Compose**, che consente di descrivere l'interfaccia utente come funzione dello stato dell'applicazione e facilita la gestione degli aggiornamenti dinamici dell'interfaccia.

La schermata principale dell'applicazione mostra l'elenco delle campagne disponibili tramite una lista scrollabile realizzata con il componente

`LazyColumn`. Ogni campagna viene rappresentata tramite una *card* contenente le informazioni principali relative al task.

La comunicazione tra client e server avviene tramite API REST basate su HTTP. Le richieste di rete sono gestite tramite la libreria **Retrofit**, che semplifica l'esecuzione di chiamate HTTP asincrone. La libreria **OkHttp** viene utilizzata come client HTTP sottostante per la gestione delle connessioni di rete.

Le operazioni di rete vengono eseguite in modo asincrono tramite **Coroutines**, garantendo la reattività dell'interfaccia utente anche durante operazioni potenzialmente lunghe, come il caricamento di file multimediali.

Per la gestione delle operazioni periodiche in background, come il controllo della presenza di nuove campagne, viene utilizzato **WorkManager**. Questo componente garantisce l'esecuzione affidabile dei task anche nel caso in cui l'applicazione venga chiusa o il dispositivo venga riavviato.

Il backend con cui l'applicazione interagisce è stato sviluppato in **Python** utilizzando il framework **Flask**, come descritto nel Capitolo 3. Il server espone una serie di API REST che permettono al client di recuperare le campagne attive, inviare submission e gestire il sistema di token.

Per la persistenza dei dati lato server viene utilizzato **MongoDB**, che consente di memorizzare in modo flessibile le informazioni relative alle campagne, alle submission degli utenti, ai file multimediali e ai token utilizzati.

Il sistema implementa inoltre il meccanismo di **blind signature RSA** descritto nei capitoli precedenti. In questo modo il backend può firmare i token utilizzati come ricompensa senza conoscere il loro valore originale, garantendo l'anonimato dell'utente e impedendo il collegamento tra la submission e la successiva fase di spesa dei token.

Conclusioni

In questa tesi è stato progettato e implementato un sistema di incentivazione privacy-preserving per il Mobile Crowdsensing basato sull'utilizzo delle *blind signatures*. L'obiettivo principale del lavoro è stato quello di consentire agli utenti di ottenere ricompense per i propri contributi senza permettere al server di collegare la fase di emissione del token con quella di utilizzo della ricompensa, garantendo così proprietà di anonimato e non tracciabilità.

Il sistema è stato sviluppato secondo un'architettura client–server ed è stato implementato utilizzando differenti configurazioni ottenute combinando due modelli architetturali, *single-key* e *multi-key*, con diverse primitive crittografiche, tra cui RSA e schemi basati su Gap Diffie–Hellman (GDH), con e senza aggregazione delle ricevute. Questa struttura ha permesso di analizzare sperimentalmente l'impatto delle diverse scelte progettuali sulle prestazioni complessive del sistema.

I risultati dei benchmark mostrano che gli schemi basati su RSA introducono un overhead computazionale relativamente contenuto rispetto alla comunicazione baseline, rendendoli particolarmente adatti a scenari reali di Mobile Crowdsensing. Gli schemi basati su GDH, pur offrendo funzionalità più avanzate come l'aggregazione delle receipt, presentano invece un costo computazionale significativamente maggiore, principalmente dovuto alle operazioni di pairing su curve ellittiche.

Infine, per dimostrare l'applicabilità pratica della soluzione proposta, è stata sviluppata un'applicazione mobile per dispositivi Android che consente agli utenti di partecipare a campagne di Mobile Crowdsensing e ottenere

ricompense tramite token anonimi basati su blind signatures.

Nel complesso, il lavoro mostra come l'utilizzo di tecniche crittografiche basate su blind signatures possa rappresentare una soluzione efficace per conciliare meccanismi di incentivazione e tutela della privacy nei sistemi di Mobile Crowdsensing.

Come possibile sviluppo futuro, sarebbe interessante valutare il comportamento del sistema in scenari distribuiti reali, in cui client e server siano connessi attraverso reti mobili e infrastrutture cloud, così da analizzare l'impatto della latenza di rete e della variabilità delle condizioni di comunicazione sulle prestazioni del protocollo.

Un ulteriore aspetto rilevante riguarda l'analisi della scalabilità del sistema in presenza di un numero elevato di partecipanti e di campagne di crowdsensing simultanee, al fine di valutare il comportamento del protocollo in contesti applicativi più realistici.

Dal punto di vista dei meccanismi di incentivazione, potrebbe inoltre essere interessante integrare il sistema con modelli di reward più avanzati, capaci di tenere conto non solo della quantità dei contributi ma anche della qualità e dell'affidabilità dei dati raccolti. In questo contesto potrebbero essere esplorate tecniche di validazione dei contributi basate su reputazione degli utenti o su meccanismi di consenso tra partecipanti.

Infine, un ulteriore sviluppo potrebbe riguardare l'estensione dell'applicazione mobile e l'integrazione del protocollo con piattaforme di crowdsensing su larga scala, al fine di valutare l'effettiva applicabilità della soluzione proposta in scenari operativi reali.

Bibliografia

[1] Capponi, Andrea, Claudio Fiandrino, Burak Kantarci, Luca Foschini, Dzmitry Kliazovich, and Pascal Bouvry. "A survey on mobile crowdsensing systems: Challenges, solutions, and opportunities." IEEE communications surveys & tutorials 21, no. 3 (2019): 2419-2465.

[2] 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

[3] Bedogni, Luca, and Federico Montori. "Joint privacy and data quality aware reward in opportunistic Mobile Crowdsensing systems." Journal of Network and Computer Applications 215 (2023): 103634.

[4] Chaum, David. "Blind signatures for untraceable payments." In Advances in Cryptology: Proceedings of Crypto 82, pp. 199-203. Springer, 1983.

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

[6] Zhang, Zhong, Dae Hyun Yum, and Minho Shin. "PARS: Privacy-aware reward system for mobile crowdsensing systems." Sensors 21, no. 21 (2021): 7045.

[7] Dimitriou T. Privacy-respecting rewards for participatory sensing applications; Proceedings of the 2018 IEEE Wireless Communications and

Networking Conference (WCNC); Barcelona, Spain. 15–18 April 2018; pp. 1–6.

[8] Zhang B., Liu C.H., Lu J., Song Z., Ren Z., Ma J., Wang W. Privacy-preserving QoI-aware participant coordination for mobile crowdsourcing. Comput. Netw. 2016;101:29–41. doi: 10.1016/j.comnet.2015.12.022

[9] Li Q., Cao G. Providing privacy-aware incentives for mobile sensing; Proceedings of the 2013 IEEE International Conference on Pervasive Computing and Communications (PerCom); San Diego, CA, USA. 18–23 March 2013; pp. 76–84.

[10] Li Q., Cao G. Providing efficient privacy-aware incentives for mobile sensing; Proceedings of the 2014 IEEE 34th International Conference on Distributed Computing Systems; Madrid, Spain. 30 June–3 July 2014; pp. 208–217.

[11] Gisdakis S., Giannetsos T., Papadimitratos P. Security, privacy, and incentive provision for mobile crowd sensing systems. IEEE Internet Things J. 2016;3:839–853. doi: 10.1109/JIOT.2016.2560768.

[12] Niu X., Li M., Chen Q., Cao Q., Wang H. EPPI: An E-cent-based privacy-preserving incentive mechanism for participatory sensing systems; Proceedings of the 2014 IEEE 33rd International Performance Computing and Communications Conference (IPCCC); Austin, TX, USA. 5–7 December 2014; pp. 1–8.

Ringraziamenti

Desidero innanzitutto ringraziare i miei genitori, Cersino e Tiziana, che mi hanno sempre sostenuto (e soprattutto mantenuto) con affetto, pazienza e fiducia durante tutto il mio percorso di studi.

Grazie di cuore.

Un ringraziamento speciale va anche alle mie sorelle, Giulia e Federica e a tutti gli amici che mi sono stati accanto in questi anni, per il sostegno, l'incoraggiamento e per aver reso questo viaggio più leggero e pieno di momenti da ricordare.

Infine, il ringraziamento più importante va a Cirillo.

Manchi tantissimo.