Security and Privacy for Mobile Crowdsensing: Improving User Relevance and Privacy

Cihan Eryonucu and Panos Papadimitratos

Networked Systems Security Group
KTH Royal Institute of Technology, Stockholm, Sweden
`https://www.eecs.kth.se/nss`
{`eryonucu,papadim`}`@kth.se`

Abstract. Mobile crowdsensing (MCS) leverages smart devices for diverse data collection tasks, ranging from noise measurements to traffic congestion levels. However, with security and privacy a prerequisite for deployment, creating a diverse ecosystem, considering user specifics, providing adequate privacy to task initiators, and enhancing user control are key factors for MCS systems to achieve their full potential. We introduce our secure and privacy-preserving architecture for MCS, designed to address these challenges, improving user control, relevance, and privacy. Our work utilizes a variant of identity-based encryption to capture user characteristics and attributes, enabling secure task enrollment and eligibility enforcement while reinforcing task initiator privacy. This study emphasizes modularity as a design goal, enabling system entities to function without relying upon others while supporting all security and privacy requirements of MCS stakeholders. We finally evaluate feasibility and efficiency to show that the proposed system is practical.

Keywords: Mobile Crowdsensing · Security · Privacy.

1 Introduction

Mobile crowdsensing (MCS) is revolutionizing the way data is gathered, by harnessing the ubiquity and connectivity of smart devices equipped with user-friendly interfaces and diverse sensing capabilities. MCS applications range from noise measurement [12] and environmental radiation monitoring [3] to traffic congestion levels [14] and popular times of businesses and places [11].

While the use of MCS offers a multitude of benefits, it also raises concerns. Privacy, as user-contributed data often expose sensitive information, e.g., frequently visited places [20], and security, as collected data can be manipulated, e.g., through Sybil-based attacks, leading to false information provided to users of popular apps [6]. Data verification is crucial to sift maliciously or erroneously submitted data [10,21], complemented by accountability mechanisms for offending users while protecting users' privacy, is another issue. Incentives should encourage user participation without connecting them to the submitted data. The primary challenge is crafting a comprehensive solution for MCS that coordinates the aforementioned concerns while maximizing the advantages of MCS. To this end, secure and privacy-preserving (S&P) MCS architectures offer decentralized solutions with system entities separated by operational roles. Despite the efforts of the research community [23,5,16,18], several challenges remain for S&P MCS architectures to address.

In spite of a good understanding of requirements [7] and significant efforts in the literature, the complexity of addressing all requirements simultaneously is challenging. Some [29,4,27,28] consider specific requirements in isolation without considering their integration with the rest, others [8,25,18] striving for comprehensive solutions, but there are still important aspects, like TI privacy, that necessitate further exploration.

Typically, the MCS service providers are regarded as single, unified entities [5,18] or a firmly interconnected network with multiple entities [8,25] with various collaborating entities responsible for specific services e.g., credentials, remunerations, etc. While these entities depend on each other for proper functionality, there is no mechanism to utilize them individually, leading to poor usability of the MCS system. Furthermore, users have weak influence over the MCS policies, specifically regarding selecting the security, privacy, or utility policies best suited to them, which in turn hinders user relevance, trust, and participation in the system.

To tackle the identified challenges while retaining the advantages of the S&P MCS architectures, we propose a fresh design look and leverage attribute-based cryptography (ABC). We capture the properties of the participants in a verifiable manner, enabling secure task release, hiding task information from incompetent users, and task eligibility enforcement, barring incapable users from tasks to which they cannot reliably contribute. We also highlight the importance of modularity as a design goal, an aspect that received little attention from the literature. Modularity entails allowing a multiplicity of actors to instantiate different parts of the architecture without compromising the system's integrity or functionality. Our contributions are summarized as follows:

- We improve the user relevance in MCS by incorporating user characteristics in a privacy-preserving and verifiable way into the broad system design. By harnessing ABC, we imprint user properties (computational power, sensory capabilities, and reputation) into the credentials to enable task eligibility enforcement and ensure the quality of contributions.
- We enhance task initiators' privacy by designing a secure task release mechanism to make the tasks accessible and visible *only* to users with adequate capabilities.
- Lastly, we evaluate our system's feasibility and efficiency, future-proofing the system's practicality and compatibility with extensions.

2 System and Adversary Model

System Model: We start by introducing general MCS system entities (actors).

- **Task Initiators (TIs)** are organizations, such as government agencies, non-profit organizations, private companies, and academic institutions. TIs launch their campaigns, seeking to collect data from contributors to form knowledge about phenomena and activities. Tasks specify various parameters for data collectors, such as the required sensors, data format, the area(s) of interest, task duration, remuneration budget, and a minimum number of users. Task areas can be defined using geographical coordinates or regions, i.e., historical quarters, municipalities, cities, etc.
- **Contributors**, participants, or users are individuals with mobile sensing devices providing the raw measurements needed to gain insights regarding tasks. Each contributor is unique, with characteristics based on location, demographics, motivation, expertise on the task, and the mobile computing and sensing platform (e.g., smartphone).
- **Security and Privacy Infrastructure** provides necessary technical platforms, software, and other resources to support user registration, task enrollment, data collection, and remuneration management. The infrastructure is the facilitator between TIs and contributors. Third-party stakeholders may participate in the infrastructure as identity providers (IdPs) and certificate authorities (CAs), providing authentication to their users, government agencies regulating sensitive data collection tasks, or data aggregators collecting and processing the collected data. We dissect and define the S&P infrastructure entities' roles in Sec. 5.

Adversary Model: We consider external and internal adversaries aiming to abuse the MCS system. We assume MCS infrastructure entities are *honest-but-curious*, i.e., they follow the defined protocol behavior, but they are curious to learn information about clients; such as sensory capabilities, device/network-specific information, remuneration details, and task enrollment history. Further, they actively try to link users syntactically by observing changes in credentials in use and semantically by inspecting the information, like their whereabouts over time. Malicious infrastructure entities collude to de-anonymize participants. Sec. 7 discusses the ramifications of such collusions.

External adversaries are non-registered users without access to the system services. They can still mount clogging Denial of Services (DoS) attacks[1], eavesdrop on communication, send unauthorized/forged and replay legitimate contributions, and try to collect other users' rewards.

Internal adversaries are the users and TIs, all with valid credentials, or attackers who gain unauthorized access by other means, e.g., hacking devices, making them relatively stronger compared to external attackers. Such adversaries are interested in learning information about other contributors, such as identifying the task participation history, user sensor profiling, submitting inaccurate data to pollute the data collection process, and tracking user-submitted data. The adversaries may try to obtain task information they cannot participate in. They also try to obtain unfair payments, i.e., intentionally submitting faulty data, double submissions, and remuneration for someone else's submission.

[1] Such attacks are beyond the scope of this work.

3 Security and Privacy Requirements

To realize a secure and privacy-preserving MCS architecture, we consolidate the following requirements based on the literature [7,8,5].

R1 **Privacy-preserving participation:** Privacy preservation covers separate core requirements that make up the general design goal. In prior works, these core requirements are often addressed in isolation or in combination but seldom in their entirety. Core privacy requirements include **(a)** identity privacy, covering both digital and real user identities; **(b)** location privacy, concealing users' whereabouts throughout the process; **(c)** device privacy, hiding device-specific identifiers; **(d)** data privacy, ensuring the unlinkability of the submitted data; **(e)** network anonymity, keeping sender's networking identifiers (e.g., IP addresses) confidential, which can be integrated with device privacy; lastly, **(f)** TI privacy, protecting the task description and the TI identity from unsuited users; **(g)** task enrollment secrecy, hiding the tasks participants are involved in. Collectively, these facets of privacy form full privacy protection.

R2 **Fair and private incentives:** User contributions should be compensated. Incentives could take multiple forms, ranging from monetary rewards to reputation to access to resources i.e., querying task results. Incentives should not be linked to user contributions (privacy) and should be resilient against potential exploitation from malicious or self-serving users (fairness).

R3 **Communication integrity, confidentiality, and authenticity:** Communication among system entities and users should be across secure channels that guarantee data integrity, confidentiality, and authenticity.

R4 **Access control:** The system should define the roles of the actors and these roles should be adhered to. Users should be able to access data (aggregates) from different tasks, choose, and contribute to tasks they are authorized for based on the prerequisites outlined for each task by the TI. We additionally require user properties to be verifiable as a basis for user access.

R5 **Data verification:** The contributed data quality must be verified to ensure the formed knowledge truthfulness. Data quality is assessed based on the submitted data itself combined with auxiliary information on time and context (location, system, and environment conditions).

R6 **Accountability:** Users and infrastructure entities are responsible for their actions within the system. Misbehaving users should be evicted and their credentials revoked accordingly. Likewise, malicious system entities should be identified. Accountability acts as a bedrock for building trust among participants, TIs, and third-party stakeholders (e.g., IdPs, CAs), which strengthens the integrity of the system.

4 Related Work

Often, the literature claims a particular requirement is addressed when, in fact, separate, albeit related, issues are adressed. This is most evident in privacy-preserving participation (R1), with some studies defining privacy as user identity [5,23], while others define it as data privacy [4,29]. These two requirements deserve separate consideration. To illustrate the overview of the literature, table 1 shows the comparison between the most complete S&P MCS architectures, to the best of our knowledge.

AnonySense [23], one of the early architectures in the literature, aims to uphold privacy by severing the link between users and their submitted measurements. However, AnonySense does not provide a robust revocation and data verification mechanism, allowing adversaries to pollute the data collection persistently. Moreover, it does not consider the device and TI privacy. PEPSI [5] employs identity-based encryption to safeguard user privacy as well as TI privacy, but it does not address user location privacy and accountability. Furthermore, PEPSI is vulnerable to collusion attacks from users and TIs [13].

RPPTD [4], a privacy-preserving truth discovery framework without a trusted third party and non-colluding entities, has the users add noise to the sensed measurement before signing them. While the system assumes the existence of ground truth, it does not deal with incentives, user and TI privacy, and accountability. PRICE [29] offers another privacy-preserving truth discovery scheme emphasizing secure and privacy-aware incentivization, but it does not address accountability and anonymity.

Name	**Sybil Resilience**	**Accountability**	**Data Verification**	**Enrollment Secrecy**	**User Privacy**	**Location Privacy**	**Device Privacy**	**Data Privacy**	**TI Privacy**
This Work	●	●	●	●	●	●	●	●	●
SPPEAR [8]	●	●	●	●	●	●	●	●	○
SPOON [18]	○	○	●	○	●	●	◐	●	●
PRICE [29]	○	○	●	○	○	●	○	●	●
ZebraLancer [16]	◐	◐	○	●	●	○	○	●	●
EPTSense [25]	●	●	●	●	●	○	○	●	○
RPPTD [4]	◐	○	◐	○	○	○	○	●	○
PEPSI [5]	○	○	○	○	●	○	●	○	●
AnonySense [23]	○	○	○	●	●	○	◐	○	○

Table 1: Features in secure and privacy-preserving MCS architectures.
●: Feature present. ◐: Not fully addressed/considered. ○: Feature missing

A number of studies use blockchains as service providers for MCS [26,30]. ZebraLancer [16] allows users to participate anonymously in tasks while ensuring data confidentiality and fair incentives. Although it facilitates revocation, it limits users to submit only a single measurement, creating constraints for real-world applications and hardening compatibility with other systems.

SPOON [18] focuses on precise task allocation paired with reputation management while preserving user privacy. Tasks are allocated to users based on their credits (reputation) and location without revealing any. However, the service provider links users to the tasks, and accountability is not considered. Furthermore, users are assumed not to be able to spoof their device locations.

SPPEAR [9,8], in conjunction with SHIELD [10], is a comprehensive S&P MCS architecture in the literature. SPPEAR enables anonymous contribution to the tasks, supports revocation, and fair remuneration. However, it does not consider TI privacy and user control.

The aforementioned works do not consider modularity and user control as design goals for their system. Although regularly addressed in isolated works, TI privacy and enrollment secrecy are usually of secondary importance in works and seldom addressed together with other requirements [18,8,5,25,23].

5 Architecture Overview

In this section, we give an overview of the system entities in Fig. 1 comprising our architecture. Infrastructure entities are modular; they do one thing and satisfy one requirement, autonomous; operate independently, and handle individual requests without any collaboration from other entities, and flexible; they support multiple policies for actors to pick based on their needs, can change over time, and are involved in different ways (e.g., supplying keys to vehicular communication systems). Consequently, the architecture endorses the separation of duties principle [22].

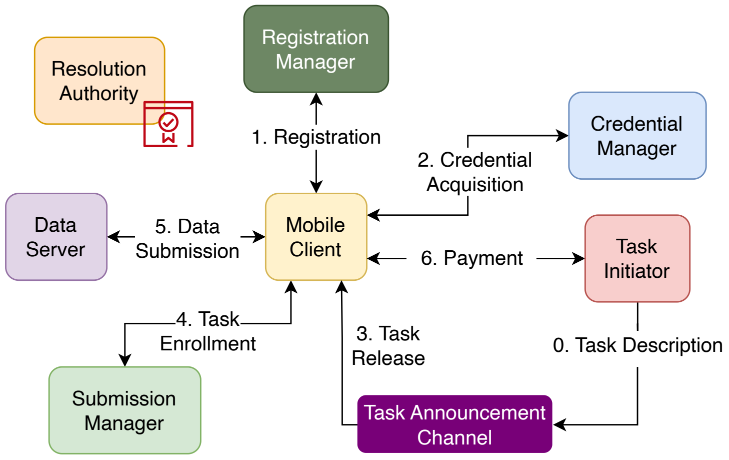


Fig. 1: System model

Task Initator (TI) and Task Announcement Channel (TAC): TIs create tasks, and users select the ones that interest them. TAC facilitates the announcement and browsing of various tasks. The publishing process for tasks can take multiple forms: Tasks are either evaluated by the system or another trusted entity before they are published, or they can be published without restrictions, placing the responsibility of evaluation solely on individual users. For enhanced privacy and security, a *task release* procedure is utilized, configuring tasks to be visible only to users satisfying specific preconditions, e.g., users with a specific type of sensor.
Mobile Client: Users interact with the MCS system and contribute to the tasks with a mobile app on their mobile devices. The app enables users to register, select tasks, collect rewards, and participate in tasks. The platform of the mobile client is not only smartphones, as such apps can be utilized by a broad range of devices, e.g., vehicular systems, UAVs, etc.
Registration Manager (RM): RM serves as the initial point of contact between the mobile client and the infrastructure. With two primary responsibilities, the RM registers new users and validates the presence of devices and users entering the system.
Credential Manager (CM): Issues and manages long-term credentials (LTCs) for registered participants. CM issues two types of credentials: Anonymous authentication based schemes and public key cryptography (PKC) based primitives. CM also handles credential exchange between external PKIs and identity providers (i.e., Google).
Submission Manager (SM): Tasked with providing users with essential tools and privacy-enhancing technologies (PETs) to submit data in a privacy-preserving manner, most frequently, pseudonyms, digital certificates with anonymized identities. SM can also procure tools required in PETs, i.e., primitives for differential privacy, secure multi-party computation (MPC), etc. SM handles task enrollment requests and, consequently, issues pseudonyms/tools for users.
Data Server (DS): It collects, stores, and aggregates user-collected data. DS also has data verification functionality, sifting the user-submitted data based on the perceived quality and accuracy. Deemed maliciously and wrongly submitted data are excluded and are not included in the task aggregates. DS issues receipts to users upon their data submission.
Resolution Authority (RA): Eviction of users and revocation of their credentials are mediated by the RA. The revocation procedure starts when a user misbehaves and detected by the system entities. The system then revokes the detected users' credentials. The accountability of the potentially compromised or deviant entities is also overseen by the RA.

6 Protocols

The architecture utilizes PKC and attribute-based cryptography (ABC) as protocol building blocks. Each entity has its own key pair and certificate signed by the CM, as well as the public parameters of ABC constructions, to be able to verify attribute-based signatures. Only mobile clients have the attribute-based keys (ABKs) to decrypt task descriptions and authenticate themselves to enroll for tasks. The network connection between entities and clients is secured via server-authenticated transport layer security (TLS). We describe cryptographic preliminaries for protocols before we describe them. Table 2 provides notations and abbreviations.

Notation	**Description**
RM	Registration manager
CM	Credential manager
SM	Submission manager
DS	Data server
TI	Task initiator
RA	Resolution authority
$ABK_{pattern}$	Attribute-based key with *pattern*
t, t_s, t_e	Current, start, and end timestamp
λ	Attribute universe size
l	$User_{id}$ length
n	Number of active tasks
k	Number of IDs in the CRL
$[m]_{pattern}$	Message m encrypted with *pattern*
$\{\cdot\}_{\sigma_{pattern}}, \{\cdot\}_{pattern}$	Signature σ generated with *pattern*
$h(\cdot)$	Hash function
$E_{pattern}(\cdot)$	Encryption function utilizing a *pattern*
$D_k(\cdot)$	Decryption function with key k

Table 2: Table of notations

6.1 Preliminaries

We utilize ABC as a main form of LTC. We use the JEDI scheme [15] as a building block based on identity-based encryption WKD-IBE [1]. First, we define *patterns*, a λ-sized list of integers and *wildcard* symbols with the following format:

$$pattern = \{P(i); P(i) \in \mathbb{Z}_p^* \cup \{*\}, i \in \{1..\lambda\}\}$$

ABKs, signatures, and ciphertexts have patterns. Patterns play a crucial role in authentication and decryption, facilitated by a 'match' function. We say $match(pattern_x, pattern_y)$ returns *True* iff $\forall i \in \{1..\lambda\} : (P_x(i) = P_y(i)) \vee (P_x(i) = *)$. The asterisk $(*)$ denotes an optional value in patterns, indicating that the field is capable of matching any value. The optional values can be delegated to be transformed into fixed values, generating a brand-new pattern. Once an optional value is delegated and assigned a specific value, the field becomes final and loses its optional status.

Delegation of patterns gives control to users as it enables them to modify their attributes. Users may use this feature not to disclose some of their existing attributes. In our scheme, we allocate 25 fields specifically for the most common types of sensors typically employed in crowdsensing, i.e., $i = 1$ for Bluetooth, $i = 2$ for GPS, etc. We assign 6 fields to represent the date, time, and week number to specify the ABK validity and facilitate passive revocation. Finally, we designate 64 fields for the $User_{id}$ to aid credential revocation. Owned attributes, such as a microphone, can be proven by generating a signature using those attributes. The patterns reinforce user control, ensure secure task enrollment by matching them with clients' sensory capabilities, establish credential expiration by denoting valid time periods as attributes, and aid in revocation procedures by incorporating $User_{id}$ into the keys.

The utilized functions of the ABC scheme are as follows:

- **setup**(λ): Takes pattern universe size (λ) as input, then initializes and returns the master secret key (msk) and public parameters $(params)$.
- **keygen**$(msk, params, pattern)$: Generates a key $ABK_{pattern}$ with given *pattern* using public parameters and master secret. If the input is another key instead of a *msk*, a new key, $ABK_{pattern}$, is generated (if the key pattern *matches* the input *pattern*); this is also called key delegation.
- **encrypt**$(params, pattern, m)$: Encrypts the given message (m) with the *pattern*. Returns the ciphertext as $[m]_{pattern}$. Note that one does not need a key to encrypt a message.
- **decrypt**$(ABK_{pattern_u}, [ciphertext]_{pattern})$: Decrypts the ciphertext using the given ABK. $pattern_u$ must *match* the ciphertext pattern for successful decryption.
- **sign**$(params, ABK_{pattern_u}, pattern, m)$: Generates a signature $(\sigma_{pattern})$ for a message (m) with a specific *pattern*. ABK $pattern_u$ must *match* the signature *pattern*.
- **verify**$(params, pattern, \sigma_{pattern})$: Returns true if the signature $(\sigma_{pattern})$ is generated using *pattern*, otherwise false. One does not need an ABK to verify signatures.

6.2 High-level Overview

We discuss first, in brief, the overall system operations corresponding to Fig. 1, then present in more detail each protocol. Initially, the mobile client registers with the RM, and it undergoes what we term as *Sybil probing* test for verification. Once successfully verified, the RM issues a short-term token to certify the client's sensors and capabilities (Step 1). Using this token, the client acquires an ABK with defined capabilities and a task enrollment ticket from the CM (Step 2). The client then fetches available tasks, decrypts only the ones it can participate in, and picks the desired task (Step 3). The client proves it is capable of carrying out the task by authenticating itself using the ABK and presenting the ticket to enroll in a task. After successful enrollment, the client receives pseudonyms from SM (Step 4). Finally, the client collects measurements and submits data, signing it with a private key corresponding to a pseudonym and receiving a receipt as proof of submission (Step 5). This receipt is then used to request payments from the TI (Step 6).

6.3 Device Registration

The registration manager is the first point of contact between mobile clients and the S&P MCS architecture every time they want to use the system. Its responsibilities encompass user management and coordinating the *Sybil-probing* process.

The purpose of *Sybil-probing* is two-fold: Determining the mobile client's sensory capabilities and safeguarding the system against Sybil devices. The mobile application gathers information about the device's capabilities while simultaneously validating its functioning. The app does not collect unique identifiers associated with the device but rather performs sanity checks [2], detecting the potential sensors that will be utilized by the client. The sensor condition can be established through various steps, including collecting sample data, validating calibration information, and analyzing diagnostic data. It is important to note that only the sensor condition, which involves the presence and capabilities of the sensor, is shared with the RM. No private data pertaining to the device is disclosed. Subsequently, $User_{id}$, condition of the sensors, validity period, token's scope, and a random number, r, are then bundled together and signed by RM to create a short-term token. We define the token format as follows

$$token = \{User_{id}, t_s, t_e, scope, sensors, r\}_{\sigma_{RM}}$$

The scope can be *ABK* for acquiring ABK, *ticket* for using the token as a *task-joining ticket*, and *authenticate* for utilizing the token as a system-wide credential. Specifically, an *ABK* token is used for authenticating the client to the CM, with a short validity period. Once verified, the CM issues an ABK for sensors defined in the token field $sensors$. Only one *ABK* token can be active at any given time per client. The *authenticate* token provides access and authentication for system operations, e.g., obtain pseudonyms, make contributions, etc., allowing clients to engage with different parts of the system without specialized cryptography, which may be crucial for some types of clients (i.e., low-power devices).

The issued tokens are in OAuth2 format [19], the industry-standard framework enabling parties to authenticate to third parties without using user credentials. OAuth2 enables compatibility with existing services and makes adoption easier with external parties easier. Sybil-probing occurs every time a user starts the mobile client and also periodically to verify the client is an actual device.

6.4 Credential Acquisition

The CM is tasked to issue Long-term credentials (LTCs) and task-joining tickets. LTCs are typically issued as ABKs but can be issued as ordinary digital certificates as well. We utilize tickets to enable revocation and regulate access to the tasks. Depending on the use case, both may have extended validity periods, e.g., lasting several days to weeks, compared to tokens issued by the RM.

With an ABK token from the RM, a mobile client can request an ABK. The token, encoded with a sensor list and $User_{id}$, becomes the blueprint for creating the key's pattern. Each sensor field value represents the version number of the corresponding sensor. For instance, if a client possesses Bluetooth 5.1, this would be represented as $P(1) = 51$. If the client does not have a particular sensor, the value for that field is assigned as 0.

The CM has the authority to define the key's validity period, allowing users to utilize the key over an extended timeframe. This flexibility can be demonstrated by, e.g., leaving the key's pattern's day field as optional, ensuring the key remains valid for an entire month. Alternatively, the week number field can be employed to extend the key's validity to span a full week. Finally, the $User_{id}$ fields are set in binary format.

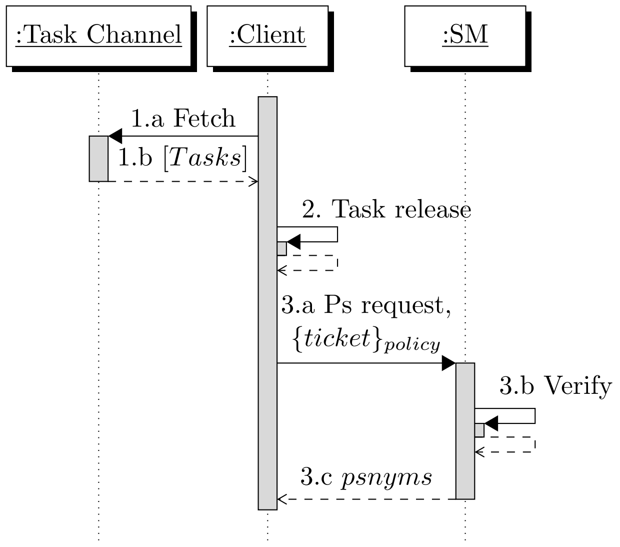
We utilize task-joining tickets as the building block of the revocation process. This is facilitated through the use of authenticated tickets, which securely link clients to their pseudonyms while maintaining their anonymity. The ticket structure is defined as follows:

$$ticket = \{h(User_{id}, t_s, t_e, r), (t_s, t_e)\}_{\sigma_{CM}}$$

The user ID remains confidential as it is hashed together with a random number and the ticket's start and end times. This obfuscates the user's identity while still allowing the CM to disclose it when necessary. The CM dispenses tickets to clients, provided they haven't exceeded the permitted number of tasks. Tickets can be provisioned ahead of time for later usage to enhance user control. One can also utilize tokens with a *ticket* scope instead of tickets.

6.5 Task Release and Enrollment

The system stores task definitions in the task channel. Depending on the TI policies, tasks can be encrypted using their specific policies (patterns) or can be in plaintext. Clients without the capabilities for a particular task cannot decrypt or authenticate with the patterns defined in their ABK, so they can neither access the definitions nor participate in those tasks. We outline the *task release* protocol as follows:



Step 1. The client retrieves encrypted tasks and their associated patterns from the task announcement channel. The client refrains from querying tasks with its preferences as it would reveal its attributes. The TAC returns encrypted tasks bundled with their policies, $[Tasks] = \{[task_i]_{policy_i}; i \in \{1..n\}\}$.

Step 2. The client decrypts all suitable tasks with its ABK whilst disregarding tasks with incompatible patterns. More formally, the client operation can be denoted as $Client : D_{ABK_u}([task_i])$ if $match(ABK_u, policy_i)$ for all $i \in \{1..n\}$. Upon selecting a task to join, the client generates key pairs for the subsequent certificate signing requests (CSRs), with the number and lifetime set as per the task specifics.

Step 3. Upon selecting a task to join, the client generates key pairs for the subsequent certificate signing requests (CSRs), with the number and lifetime set as per the task specifics. The client sends a request for pseudonyms, coupled with the signed task-joining ticket and CSRs. The ticket is signed with the task's policy, validating the client's capability to join the task, while the ticket anonymously binds the client to pseudonyms. CM verifies whether the signature pattern matches the task policy and whether the ticket is authenticated and valid. After the verification, the CM issues and returns pseudonyms to the client.
In step 3.a, the mobile client can also utilize a token with a *ticket* scope to authenticate itself. If the task has a sensory requirement, the token should also encapsulate the sensors. Tokens provide an alternative authentication means for clients, enhancing user control and at the cost of their privacy.

6.6 Data Submission and Remuneration

The mobile client contributes to the tasks by leveraging pseudonyms or tokens. We define each sample authenticated by pseudonym as $s = \{data, loc, radius, t\}_{\sigma_{ps_i}}$, where $data$ represents the required sensing for the task, loc signifies the location of the sensing, $radius$ designates the precision of the position, and t stands for the timestamp. These fields are hashed and subsequently signed using the private key that corresponds to the pseudonym ps_i. Using pseudonyms provides maximum privacy.

Clients can alternatively submit their contributions using a token with a "submit" scope is denoted as $s = \{data, loc, radius, t, token_{\sigma_{RM}}\}$. This approach circumvents the necessity of performing any cryptographic operations, such as generating signatures, on the mobile client, as the tokens are authenticated by the RM. Although using tokens impinges upon the mobile client privacy, it offers better performance, compatibility, and, more importantly, flexibility.

After the verification of the submission, DS creates a receipt for the mobile client formalized as $receipt = \{ID_{receipt}, quality, t\}_{\sigma_{DS}}$ where quality states the assessment of the contributed data. Clients can then send the receipts to TI to redeem them. Alternatively, TI can pay users through forwarding payments over the infrastructure.

6.7 User Revocation

Our architecture revokes the credentials of misbehaving any client (i.e., contributing incorrect measurements to corrupt the collected data), irrespective of the type of credentials they use. Misbehavior detection schemes [10,17], designed specifically for the PS, can be employed to identify such attacks[2]. The inclusion of ABC contributes to the integration of data verification and truth discovery schemes.

Revocation can be tailored to the system needs and can take several forms, ranging from barring clients from a single task to expulsion from MCS. Our architecture allows the revocation of all issued credentials, with the exact revocation strategies being left to the discretion of the system operators.

Upon detection of misbehavior, the RA is alerted to initiate the process. The revocation request includes the malicious client credentials, a pseudonym set, or a token. This is then forwarded to the issuing entity to pinpoint the misbehaving user. The identification process is straightforward if the data submission involves a token, as the $User_{id}$ is tied to the token. For submissions with pseudonyms, the RA consults the SM to obtain the ticket used during the task enrollment. This request also prompts the SM to revoke any remaining pseudonyms linked to the user. This ticket is sent to CM to resolve the $User_{id}$. Finally, CM adds the $User_{id}$ to the credential revocation list (CRL) and informs RM to halt any further credential issuance.

Revoked credentials can no longer be used. This is apparent in the case of revoked tokens, which include the bearer's ID, allowing the verifier to compare the ID with the CRL directly. However, with ABK signatures being anonymous,

[2] The construction of these misbehavior detection methods is beyond the scope of this work and therefore is not discussed here.

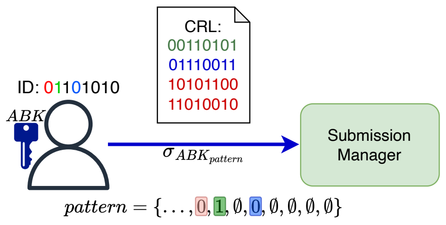


Fig. 2: A client proving it is not revoked. Each colored bit in the signature pattern affirms the user is not on the CRL

a mechanism is required to enable entities to verify whether the signer credentials were revoked.

To demonstrate their credentials are still valid, clients disclose a number of bits of their $User_{id}$, an *unrevoked subset*, to the authentication pattern before signature generation. Given that every $User_{id}$ is unique, if a user is not listed on the CRL, then at least one bit differs from any ID on the list. Revoked clients cannot find such a bit or subset and thus cannot authenticate.

We provide an illustration of the authentication process depicted in Fig. 2. For simplicity, we assume $User_{id}$ is 8-bits long, and four IDs are listed on the CRL. The client examines the CRL to assemble its *unrevoked subset*. The first bit, shaded in red, eliminates the last two entries, also colored red, from the list. The bit colored in green eliminates the top entry, and the blue bit effectively cancels out the remaining entry. Note that the revealed patterns ultimately leak information about the $User_{id}$. The extent of the leakage depends on the ID length, number of revoked IDs in the CRL, and *unrevoked subset* construction. We further explore these relationships in Sec. 8.

We design two primary strategies to devise *unrevoked subsets*: random and greedy bit selection methods. The random method randomly selects $User_{id}$ indices in each iteration, adding them to the subset if they eliminate any $User_{id}$. Then, it updates the list, eliminating the IDs in CRL based on the last added bit, and proceeds to the next iteration until no IDs are left in CRL. However, there is a chance that the method might choose an index that does not eliminate any IDs, exemplified by the 3rd bit of the $User_{id}$ in Fig. 2, and it may not always yield the optimal subset.

We can employ a greedy method to find the least revealing subset. In this method, the client selects the bit that results in the highest ID elimination from the CRL until all IDs are eliminated. The selection process involves identifying the index i that $\min_{i \in \{1..l\}} f_i(ID(i))$ where $f_i(bit)$ finds the frequency (occurrence) of the *bit* in index i of the CRL and $ID(i)$ returns the $User_{id}$ for the index i. However, there is a drawback with performance as the entire CRL, involving l columns and k rows, is scanned to find the index, remove it, and repeat for the reduced CRL. On average, every selected bit halves the IDs in the CRL. Only the residual list is examined for subsequent iterations, making a logarithmic growth rate and expected subset size of random method $O(logk)$

with k representing the CRL size. The greedy method examines all indices (l columns) in every iteration, resulting in a $O(l * logk)$ complexity. For small-sized lists, revealing a single bit could be sufficient for the subset.

7 Security and Privacy Analysis

We informally analyze how the proposed protocols and architecture achieve the defined requirements. Note that the collusions between entities do not break the requirements; e.g., if some parties need to collude to break an actor's privacy, we still consider the requirement as addressed.

User identities are shared only with the RM during the initial registration, and then the user is assigned to its system ID, i.e. $User_{id}$. The RM only knows about this binding. The $User_{id}$ is revealed only to CM when obtaining tickets and ABKs. Apart from this, the CM does not know anything about the client and its actions.

Clients authenticate anonymously via attribute-based signatures, preventing the SM from linking pseudonym requests revealing a small subset of their IDs during authentication. Further, the presented ticket has its $User_{id}$ masked. All in all, the SM does not know which client requested the pseudonyms for a specific task, nor can it link any two requests to a client. Lastly, the CM issues the tickets without any task information, so task privacy is protected.

Entities	**Information Exposure**	**Possible Ramifications (if any)**
RM	$User_{id}$	The RM knows the user is registered with *id*.
CM	*sensors*, $User_{id}$	The CM infer that $User_{id}$ has the *sensors*.
SM	*psnyms*, $Task_{id}$, *ticket*	The SM infer that an anonymous user has pseudonyms for the $Task_{id}$.
DS	*s*, *receipt* *psnyms*, $Task_{id}$	The DS know that submissions come from some user for a specific task.
RM, CM	*sensors*, $User_{id}$	No new information gained by this collusion
RM, SM	$User_{id}$, *psnyms* $Task_{id}$, *ticket*	The RM and the SM cannot link any credential with $User_{id}$.
RM, DS	$User_{id}$, *s*, *receipt* *psnyms*, $Task_{id}$	$User_{id}$ do not have any connection to the pseudonyms, submissions, and receipts.
CM, SM	*psnyms*, $Task_{id}$, *ticket* *sensors*, $User_{id}$	The entities can learn $User_{id}$ with *sensors* obtained pseudonyms for a task.
CM, DS	$User_{id}$, *s*, *receipt* *psnyms*, $Task_{id}$	There are no ways to link $User_{id}$ with the *psnyms*.
SM, DS	*psnyms*, $Task_{id}$, *ticket* *s*, *receipt*	They can infer and track the submissions made by an anonymous user.
CM, SM, DS	*all*	Entities can identify a $User_{id}$'s task participation history and submissions.

Table 3: Colluding entities and their combined intelligence

Mobile clients use pseudonyms for data submission, ideally fresh for each submission. Using the same pseudonym for multiple data submissions allows a curious DS to trivially link the submissions. Issued credentials (tickets, tokens, ABKs, pseudonyms) have non-overlapping validity times to prevent Sybil-based attacks. This limits data submission to one per client at any given time. Unless clients abandon their anonymity by using tokens, making it trivially detectable if they misbehave.

Device identifiers are validated locally on the client devices and are never collected by any entity. Network identifiers can be hidden by the utilization of the TOR network. Tasks can be encrypted to block unwanted parties from seeing the task details. TI identity is only known to the RM at the time of registration. Task creation is done through the TAC using tokens with one-time user IDs **(R1)**.

Receipts are signed by the DS, making it impossible for malicious users to forge them. Furthermore, no receipts can be used twice - they are consumed when used **(R2)**. Communication among any entity is over server-authenticated TLS, except while using pseudonyms during the data submission, thus achieving communication authenticity, integrity, and confidentiality **(R3)**.

Scopes defined on the issued tokens denote the access rights of the clients. Only the CM can generate ABKs with patterns. The task selection will always be limited to the defined sensors on ABK because clients cannot see or join tasks that require sensors they don't have. Tickets provide authorization for participation in tasks **(R4)**.

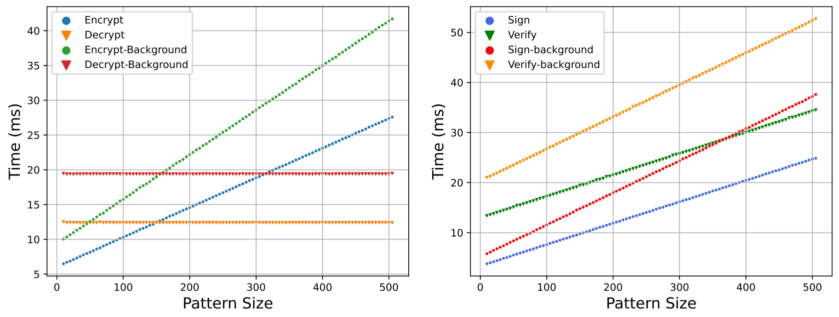
Data verification schemes deserve their own independent inquiries, so we do not design a new data verification scheme for this work. However, we make our system to be compatible and integrable with such schemes like [10] and utilize them **(R5)**.

The RA resolves any suspected misbehaviors and, if found guilty, initiates the revocation process. Evicted users are prevented from doing any operation within the system since their credentials are revoked, and they cannot get new credentials **(R6)**.

Table 3 provides insight into what MCS colluding entities infer about the users and the implications. No pair of colluding entities can completely deanonymize users. Moreover, the majority of the two-entity collusions do not result in new information leaks compared to their non-colluding states. The only time the system learns everything about the users is when CM, SM, and DS collude. Any other combination of three entities cannot achieve complete de-anonymization. Some of the entities can be run by the same organizations, given there are no privacy/security conflicts. For example, a TI can also employ its own DS with adjustments to the remuneration process, as users should be remunerated. The collaboration of CM, SM, and DS is needed for complete revocation. By adhering to the principle of separation of duties, a system of checks and balances needs to be in place to detect and handle any misbehavior.

8 Implementation and Evaluation

We implemented system entities in JavaScript and Python and developed a mobile client app for Android devices. We use OpenSSL for cryptographic operations, i.e., to generate ECDSA key pairs for entity digital certificates and pseudonyms. We implemented a wrapper library for ABC to utilize the core JEDI pairing library [15] in the mobile environment. We conduct experiments on smartphones in Xiaomi Redmi 9, released in 2020, an entry-level device with modest resources, and we host our server entities in an HP Z440 workstation with 96 GB of RAM. All the plotted values are averaged over 200 measurements and fall within a 95% confidence interval, but we do not show the intervals because the intervals are too tight (i.e., interval sizes are in the orders of microseconds).



(a) (b)

Fig. 3: Impact of pattern size over cryptographic operation latency.

We experiment with protocols using ABC, i.e., task encryption, release, and enrollment. We wish to understand how the ABC affects the protocols and how it scales with larger pattern sizes. We additionally investigate how these operations perform in the background, i.e., when the phone is locked. Fig. 3a shows the execution times for encryption and decryption, averaged over 200 measurements with an increasing number of patterns. Realistically, tasks are encrypted with patterns less than size 15, as they only need the required sensors and validity time. We see that a larger pattern size increases the encryption time; still, the operation is highly efficient in a mobile environment, even in the background. Encryption is a rare operation, only used when TAC requires task details from TIs. Comparatively, decryption is a frequent operation; all suitable tasks are decrypted by mobile clients when they want to access the tasks. The number of active tasks at a given time, thus the number of decryptions, varies, but the decryption time (latency) is low, allowing for many descriptions in a short time, e.g., 50 task descriptions per second on the background power. This is further reinforced by the fact that decryption times are unaffected by the pattern size on the ciphertext, as seen in Fig. 3a.

Fig. 3b illustrates the signature generation and verification performance, with varying numbers of patterns. Execution time grows linearly with larger patterns. We can see that the *sign* operation is cheaper than *verify*. This is advantageous for mobile clients, which are usually resource-constrained compared to the server. When in the background, both operations take 1.5 times longer but still grow linearly. Nevertheless, they perform comparably to widely employed PKC schemes [24] and create no major performance issues in the mobile environment.

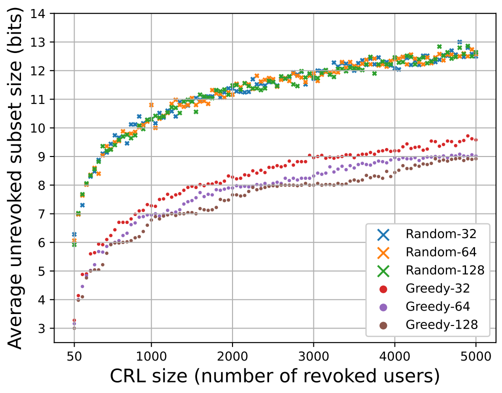


Fig. 4: Impact of CRL size on the number of revealed bits during authentication.

We assess the privacy exposure of the *unrevoked subset* during the task enrollment process. An increase in the number of clients in the CRL requires more bits to validate the IDs's absence in the CRL. We investigate the effect on the number of disclosed bits, considering varying $User_{id}$ bit lengths, for both bit selection methods in Fig. 4. Both method's effect on privacy exposure exhibits a logarithmic trend with respect to the CRL size. The greedy bit selection method discloses 30% fewer bits than the random method. It is noteworthy that the $User_{id}$ length, l, does not influence the random selection method as the unrevoked subset is considerably smaller than l. However, for the greedy method, the larger l means more indices to look for the optimal index, which can yield better results.

9 Conclusion

We presented an S&P MCS architecture focusing on relatively overlooked design goals of TI privacy, user control, and relevance. We brought attention to modularity and interoperability for the MCS architecture entities focusing on their autonomy, flexibility, and usability as a fresh design goal. We evaluated how we addressed the discussed requirements and made experiments to demonstrate the efficiency and practicality of the proposed protocols. Future research directions can include experimentation on tokens, formal analysis, rigid sybil-probing, alternative data submission policies, and security/privacy exposure labels.

References

1. Abdalla, M., Kiltz, E., Neven, G.: Generalized key delegation for hierarchical identity-based encryption. Cryptology ePrint Archive, Paper 2007/221 (2007)
2. Borsub, J., Papadimitratos, P.: Hardened registration process for participatory sensing. In: Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks. Association for Computing Machinery (2018)
3. Brown, A., Franken, P., Bonner, S., Dolezal, N., Moross, J.: Safecast: successful citizen-science for radiation measurement and communication after fukushima. Journal of Radiological Protection (2016)
4. Chen, J., Liu, Y., Xiang, Y., Sood, K.: Rpptd: Robust privacy-preserving truth discovery scheme. IEEE Systems Journal (2022)
5. De Cristofaro, E., Soriente, C.: Extended capabilities for a privacy-enhanced participatory sensing infrastructure (PEPSI). IEEE Transactions on Information Forensics and Security **8**(12), 2021–2033 (2013)
6. Eryonucu, C., Papadimitratos, P.: Sybil-based attacks on google maps or how to forge the image of city life. In: Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks (2022)
7. Giannetsos, T., Gisdakis, S., Papadimitratos, P.: Trustworthy people-centric sensing: Privacy, security and user incentives road-map. In: 2014 13th Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET) (2014)
8. Gisdakis, S., Giannetsos, T., Papadimitratos, P.: Security, Privacy, and Incentive Provision for Mobile Crowd Sensing Systems. IEEE Internet of Things Journal (2016)
9. Gisdakis, S., Giannetsos, T., Papadimitratos, P.: SPPEAR: Security and Privacy-preserving Architecture for Participatory-sensing Applications. In: ACM Conference on Security and Privacy in Wireless and Mobile Networks (ACM WiSec) (2014)
10. Gisdakis, S., Giannetsos, T., Papadimitratos, P.: SHIELD: A Data Verification Framework for Participatory Sensing Systems. In: ACM Conference on Security & Privacy in Wireless and Mobile Networks (ACM WiSec) (2015)
11. Google: Popular times, wait times, and visit duration, `https://support.google.com/business/answer/6263531`, [Online; accessed 19-June-2013]
12. Grubeša, S. and Petošić, A. and Suhanek, Mia and Durek, I.: Mobile crowdsensing accuracy for noise mapping in smart cities. Automatika (2018)
13. Günther, F., Manulis, M., Peter, A.: Privacy-enhanced participatory sensing with collusion resistance and data aggregation. In: Cryptology and Network Security: 13th International Conference, CANS 2014. Springer (2014)
14. Help, W.: How does waze work?, `https://support.google.com/waze/answer/6078702`, [Online; accessed 17-June-2023]
15. Kumar, S., Hu, Y., Andersen, M.P., Popa, R.A., Culler, D.E.: JEDI: Many-to-Many End-to-End encryption and key delegation for IoT. In: 28th USENIX Security Symposium (USENIX Security 19). USENIX Association (2019)
16. Lu, Y., Tang, Q., Wang, G.: Zebralancer: Private and anonymous crowdsourcing system atop open blockchain. In: 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS) (2018)

17. Luo, T., Huang, J., Kanhere, S.S., Zhang, J., Das, S.K.: Improving iot data quality in mobile crowd sensing: A cross validation approach. IEEE Internet of Things journal (2019)
18. Ni, J., Zhang, K., Xia, Q., Lin, X., Shen, X.S.: Enabling strong privacy preservation and accurate task allocation for mobile crowdsensing. IEEE Transactions on Mobile Computing (2020)
19. OAuth2: Oauth2 access tokens, `https://oauth.net/2/`
20. Pournajaf, L., Garcia-Ulloa, D.A., Xiong, L., Sunderam, V.: Participant privacy in mobile crowd sensing task management: A survey of methods and challenges. ACM SIGMOD Rec. (2016)
21. Restuccia, F., Ferraro, P., Sanders, T.S., Silvestri, S., Das, S.K., Re, G.L.: First: A framework for optimizing information quality in mobile crowdsensing systems. ACM Transactions on Sensor Networks (TOSN) (2018)
22. Saltzer, J.H., Schroeder, M.D.: The protection of information in computer systems. Proceedings of the IEEE (1975)
23. Shin, M., Cornelius, C., Peebles, D., Kapadia, A., Kotz, D., Triandopoulos, N.: Anonysense: A system for anonymous opportunistic sensing. Pervasive and Mobile Computing (2011)
24. Tschofenig, H., Pegourie-Gonnard, M.: Performance of state-of-the-art cryptography on arm-based microprocessors. In: NIST Lightweight Cryptography Workshop. vol. 2015 (2015)
25. Wu, H., Wang, L., Xue, G., Tang, J., Yang, D.: Enabling data trustworthiness and user privacy in mobile crowdsensing. IEEE/ACM Transactions on Networking (2019)
26. Wu, H.T., Zheng, Y., Zhao, B., Hu, J.: An anonymous reputation management system for mobile crowdsensing based on dual blockchain. IEEE Internet of Things Journal (2022)
27. Yan, X., Zeng, B., Zhang, X.: Privacy-preserving and customization-supported data aggregation in mobile crowdsensing. IEEE Internet of Things Journal (2022)
28. Zhang, X., Lu, R., Ray, S., Shao, J., Ghorbani, A.A.: Spatio-temporal similarity based privacy-preserving worker selection in mobile crowdsensing. In: 2021 IEEE Global Communications Conference (GLOBECOM) (2021)
29. Zhao, B., Liu, X., Chen, W.N., Liang, W., Zhang, X., Deng, R.H.: Price: Privacy and reliability-aware real-time incentive system for crowdsensing. IEEE Internet of Things Journal (2021)
30. Zou, S., Xi, J., Xu, G., Zhang, M., Lu, Y.: Crowdhb: A decentralized location privacy-preserving crowdsensing system based on a hybrid blockchain network. IEEE Internet of Things Journal (2022)