

Aalto University  
School of Science  
Master's Programme in Security and Cloud Computing

Tomasz Czajęcki

# Privacy Enhancing Data Reporting System For Participatory Sensing

Master's Thesis  
Espoo, July 24, 2022

**Supervisors** Prof. Panagiotis Papadimitratos, KTH Royal Institute of Technology  
Prof. Tuomas Aura, Aalto University  
**Advisors** Cihan Eryonucu, KTH Royal Institute of Technology

Copyright Tomasz Czajęcki 2022

<b>Author:</b>	Tomasz Czałajęcki		
<b>Title:</b>	Privacy Enhancing Data Reporting System For Participatory Sensing		
<b>Date:</b>	July 24, 2022	<b>Pages:</b>	54
<b>Major:</b>	Security and Cloud Computing	<b>Code:</b>	SCI3113
<b>Supervisors:</b>	Professor Panagiotis Papadimitratos Professor Tuomas Aura		
<b>Advisors:</b>	Cihan Eryonucu, KTH Royal Institute of Technology		
<p>Privacy is a crucial aspect of any system involving user-supplied data. There exist multiple approaches to protecting the identity and secrecy of users in data submission systems. In this thesis I consider the case of privacy-enhancing of data reporting in Participatory Sensing systems. I conducted an extensive literature overview to explore privacy-oriented enhancements to data submission that are applicable in the PS systems. I designed a protocol for proximity-based data aggregation that utilizes Multi-party Secure Computations over Bluetooth Low Energy. Users are divided into groups that perform sub-aggregations and report results to central entities, protecting themselves from honest-but-curious adversary threats. I present a mobile app and web servers for central entities that follow the design of the protocol. I evaluated the achieved effectiveness and discuss the utility and privacy trade-offs. The implementation performs as one would expect for an MPC system with high communication overhead, and is implemented over Bluetooth, with the additional time needed for discovering and connecting devices. The overall performance of the system suggests that deployments targeting 1-second intervals of data submission are feasible. Main use cases are sensitive measurements, such as medical data or highly private user information.</p>			
<b>Keywords:</b>	Participatory sensing, data aggregation, MPC, Bluetooth		
<b>Language:</b>	English		

# Acknowledgements

I would like to thank professor Panagiotis Papadimitratos for his extremely deep and broad knowledge of the Participatory Sensing field and for asking the most important questions that shaped the progress of this thesis.

I am grateful to professor Tuomas Aura for his insightful help in the process and for being available in the most important moments.

I also want to thank Cihan Eryonucu, my direct supervisor at KTH, whom I owe the most. His intense cooperation with me throughout the whole semester was invaluable. Only thanks to his guidance the project was moving so swiftly and effectively.

Finally I would like to thank my girlfriend, friends and family, who supported me in this challenging time.

Espoo, July 24, 2022

Tomasz Czajęcki

With the support of the  
Erasmus+ Programme  
of the European Union



# Contents

<b>1</b>	<b>Introduction</b>	<b>7</b>
1.1	Research Question . . . . .	7
1.2	Research Method . . . . .	7
1.3	Aim and objective . . . . .	8
1.4	Contributions . . . . .	8
1.5	Simplified description of the protocol . . . . .	8
1.6	Reasoning behind the choice of the solution . . . . .	8
1.7	Use cases of the protocol . . . . .	9
<b>2</b>	<b>Background</b>	<b>10</b>
2.1	Participatory Sensing . . . . .	10
2.2	Problem of anonymizing data . . . . .	11
2.3	Privacy-Utility Trade-off . . . . .	11
2.4	Related work . . . . .	12
2.5	Data-modifying . . . . .	12
2.5.1	Pre-aggregation . . . . .	12
2.5.2	Obsfucation . . . . .	12
2.6	Preventing tracing of users . . . . .	13
2.7	TLS . . . . .	13
2.8	WebSockets . . . . .	14
2.9	Bluetooth . . . . .	15
2.10	MPC . . . . .	15
2.11	Group division . . . . .	15
<b>3</b>	<b>Method</b>	<b>17</b>
3.1	System model . . . . .	17
3.2	Protocol . . . . .	18
3.3	Entities . . . . .	18
3.4	Security and privacy targets . . . . .	19
3.5	Adversary model . . . . .	19
3.5.1	Bluetooth security . . . . .	19
3.6	Zones . . . . .	20
3.7	Protocol operations . . . . .	20

3.8	Procedures . . . . .	20
3.8.1	register . . . . .	21
3.8.2	formGroups . . . . .	22
3.8.3	joinGroup . . . . .	23
3.8.4	onDisconnect . . . . .	24
3.8.5	onRemove . . . . .	25
3.8.6	checkConnections . . . . .	25
3.8.7	fixGroup . . . . .	26
3.8.8	swapRepresentatives . . . . .	26
3.8.9	runTask . . . . .	27
3.9	Example use case . . . . .	29
<b>4</b>	<b>Implementation</b>	<b>36</b>
4.1	Mobile app . . . . .	37
4.1.1	React Native . . . . .	37
4.2	Servers . . . . .	37
4.3	Group division . . . . .	37
4.4	Random number splitting . . . . .	39
<b>5</b>	<b>Results</b>	<b>40</b>
5.1	Overall system performance . . . . .	40
5.1.1	Observations . . . . .	41
5.2	Impact of group representative optimization . . . . .	42
5.3	Simulation of adversary detection methods . . . . .	43
5.3.1	Observations . . . . .	46
<b>6</b>	<b>Conclusions</b>	<b>49</b>
6.1	Future work . . . . .	49
6.2	Sustainable development . . . . .	50
6.3	Ethical considerations . . . . .	50

# List of Acronyms and Abbreviations

- API** Application Programming Interface.
- BLE** Bluetooth Low Energy.
- Bluetooth SIG** Bluetooth Special Interest Group.
- DP** Differential Privacy.
- DS** Data Server.
- FHE** Fully Homomorphic Encryption.
- GAP** Generative Adversarial Privacy.
- GDPR** General Data Protection Regulation.
- HTTPS** Hypertext Transfer Protocol Secure.
- LTCA** Long-Term Certificate Authority.
- LTE** Long-Term Evolution.
- MAC** Media Access Control.
- MPC** Secure multi-party computation.
- OSI** Open Systems Interconnection.
- PM** Policy Manager.
- PS** Participatory Sensing.
- TLS** Transport Layer Security.

# Chapter 1

## Introduction

Participatory Sensing (PS) is a model of conducting experiments where users use their mobile devices, are highly equipped with various sensors, and contribute sensed data as participants in experiments. PS is a very broad term, which includes an extremely wide range of systems with different characteristics and needs.

As much as users embrace the aspect of contributing to scientific research, there is a valid concern about privacy following the submission of data. Users strongly prefer that the information they share cannot be used to identify them, link them with other available data sets and leak any sensitive information. This leads to a need for a compromise between the utility of collected samples and the protection of users[1].

### 1.1 Research Question

The goal was to design a protocol for a secure and privacy-enhancing data reporting system which should not reveal the identity of the participants while maintaining data quality.

The research question of the thesis is:

*What would be the performance, correctness, and feasibility of such protocol and how it could be integrated in larger PS systems?*

### 1.2 Research Method

The research method involved an extensive literature study, followed by the design and implementation of the protocol, which was concluded by conducting experiments and evaluating the observations.

### 1.3 Aim and objective

The goal of this thesis was to develop a protocol for data submission in PS systems in a privacy-enhancing way. This was followed by an implementation of a proof of concept system that follows the protocol and evaluation of its utility.

After careful consideration of available options, the plan included developing a design of the protocol using Bluetooth as a proximity-based transport layer for Secure multi-party computation (MPC), which would be further used to aggregate data among local participants before uploading it to the central Data Server (DS).

### 1.4 Contributions

The main contribution of the thesis is the specification and the implementation of a protocol for data submission in PS systems leveraging MPC over Bluetooth Low Energy (BLE).

What follows is the evaluation and analysis which estimates the feasibility of adopting the protocol, considering the trade-off between the overhead of MPC calculations and BLE communication, and the observed benefits for privacy.

### 1.5 Simplified description of the protocol

The protocol defines specifics of communication between DS collecting submissions, Policy Manager (PM) supervising the system and multiple users. Users conduct MPC based calculations of results of their groups and then designated users, group representatives, report the results to the DS.

This way there is no single entity besides the user that has access to the raw input data. Group representatives have access to a total sum of values submitted by all users, similarly to the DS, which uses those results to compute the total sum or average. PM has no direct access to data at any point of protocol execution.

### 1.6 Reasoning behind the choice of the solution

As presented in further chapters and sections, there are multiple approaches to enhancing user privacy. The assumption that the system should be protected from honest but curious adversary effectively reduces selection of methods to the ones based on distributed calculation. From that, I picked MPC combined with proximity-based groups as a mature paradigm with years of research, yet not fully explored in the context of PS. Following that, I selected Bluetooth as a transport layer of enormous industry adoption and cross platform availability. Other

choices included Wi-Fi Aware[2] and Multipeer Connectivity[3], but those, unlike Bluetooth, are respectively Android and iOS-specific solutions.

## 1.7 Use cases of the protocol

The protocol comes with an undeniable added cost of both communication and computation overhead. MPC algorithm is more expensive than simple submission to the server, even when the latter includes additional security measurements such as rotation of pseudonyms.

However, the quality of protection of all user input from every other entity in the system is a characteristic not available in most of the other schemes.

Situations in which the trade-off is reasonable are mainly use cases in which system is gathering medical records of users, or there is a significant risk of using data for identifying users.

## Chapter 2

# Background

One of the aims of this thesis is providing valuable contributions. Therefore, I conducted an extensive study of previous academic progress to discover the current state of the art in the field. I searched for solutions to the problems of privacy enhancing in data submission systems, with special attention to the Participatory Sensing.

### 2.1 Participatory Sensing

PS is a field dedicated to building systems where users both contribute and benefit from the sensed data. The ability to provide information comes from the vast number of sensors that modern smartphones are equipped with, and the sheer amount of participants that can bring a substantial statistical knowledge on its own[4].

As depicted in Figure 2.1, users interact with PS systems in two ways, both as providers of data and as consumers, typically only after a processing step and a visualization process. Although, from the point of scientific interest, the former is given significantly more attention, as proper incentive and safety guarantees are crucial to attract participants.

The rise of PS systems, also called Mobile Sensing, or Crowdsensing[5], was enabled by the rapid technological advances in the mobile industry, followed by developments in cellular communication infrastructure and standards such as 3G and Long-Term Evolution (LTE) standards. Progress allowed vendors to include much more effective and diverse sensors in their devices, and companies developing operating systems were competing to attract the most mobile developers to their platforms by equipping their operating systems in the best possible Application Programming Interface (API).

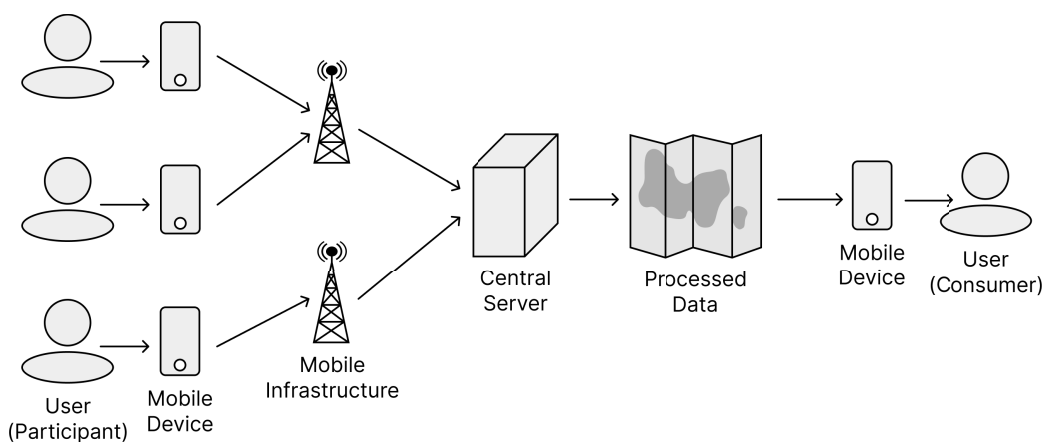


Figure 2.1: Illustration of key concepts of PS systems.

## 2.2 Problem of anonymizing data

From the research perspective, the preferable structure of data is one that is as close as possible to the raw sample of some measurement of an event. Any aggregation or compression hinders the ability of researchers to observe different aspects of said data, making it less useful.

However, recording numerous samples and storing them directly in a database creates a potential risk to the privacy of users. If, for example, the target of the experiment is to identify the most frequently used bus lanes in the city, and stored data is information about each travel of every registered holder of a season ticket, then the attacker would be able to connect users with locations of their homes and workplaces.

Apart from considerations about storing data, the final concern is the safety against an adversary known as an "honest but curious" server. It means that the data collection server (or any other part of central infrastructure) is following the protocol in regards to the data collection duties, but, apart from that, gathers additional information about users from the samples they provide.

This means that if a user sample at any point reaches the collection server in unmodified form, there is a risk of the server taking advantage of this data. It leads to a surprising conclusion, that this seemingly not dangerous adversary is, in fact, especially hard to avoid in usual server architectures.

## 2.3 Privacy-Utility Trade-off

As mentioned previously, there is a trade-off between protecting user privacy and keeping the high utility of collected data [6]. It was extensively studied and resulted in the creation of various data aggregation methods, such as Differential Privacy

(DP) [7].

## 2.4 Related work

There are multiple Participatory Sensing System designs that propose methods for improving user privacy (SPPEAR [8], PiRi [9], PEPSI [10] [11]) and the comparison of those systems was subject to a research itself [12].

I would divide said systems into two categories, based on whether the system targets protecting user identity, or if its concerns focus on privacy through anonymization of the data, with either cryptographic or data perturbation methods.

## 2.5 Data-modifying

Many methods achieve higher privacy by modifying data before or as part of the submission process. Modification of data before it reaches a central entity is desirable for protection from the honest-but-curious adversary, but offer suffers from lower quality of available methods.

### 2.5.1 Pre-aggregation

Methods focusing on modification of the input set include  $\ell$ -diversity [13], which is a modern successor of  $\kappa$ -diversity [14].

### 2.5.2 Obsfucation

**Differential Privacy** is based on modifying data adding noise from Laplace Distribution [15]. Resulting change prevents adversaries from learning user details in case of data leakage, although does not prevent honest but curious server from abusing the access to data.

There exists variations of the method such as **Local Differential Privacy**, which secures against adversary with access to the personal responses of individuals in the database [16].

**Generative Adversarial Privacy (GAP)** is a method where the optimal privacy mechanism is formulated as a constrained minimax game between a "privatizer" and an adversary [17].

**Random perturbations** is a method where random modifications are applied to the data to reduce the chance of attacker being able to link information to participants [18].

**Negative survey** [19] is a method which presents participant with  $n$  possible answers and asks to eliminate one of the  $t - 1$  answers that they think are wrong. It had successful implementations in PS systems [20] [21].

A modification called **Gaussian Negative Surveys** exists, which uses Gaussian distribution to modulate data collection and has most use cases in spatial data sampling [22].

## 2.6 Preventing tracing of users

**Group signatures** is a method which allows users to anonymously sign messages on behalf of the group, achieving the separation of user identity and identifier within the system [23]. There are multiple group signatures schemes developed over the years and combined into more advanced systems. One of them is SPPEAR [8], which offers a complex solution for managing participants, assigning tasks and revoking credentials for misbehaving users.

**Mix networks** are routing protocols that protect communication by shuffling messages from multiple senders, making it significantly harder for a specific class of adversaries to link participants to the data [24].

Yonglei Yao et al. propose a system which claims to successfully break the link between a reserved data slot and the participant, achieving privacy [25].

## 2.7 TLS

An important portion of the underlying security of the proposed protocol comes from building on top of existing Transport Layer Security (TLS) capabilities.

TLS operates in the application layer of the Open Systems Interconnection (OSI) model. It defines a stateful connection between two hosts. It starts with a handshake procedure, based on an asymmetric cipher used to establish a session-specific key. The communication itself is later encrypted using a symmetric key shared using the asymmetric one. It comes from the fact that asymmetric cryptography allows establishing a connection without prior off-channel communication, while symmetric cryptography has better encryption performance.

TLS provides privacy (confidentiality), integrity and authentication through the use of certificates.

Some PS systems, for example SPPEAR [8], include an entity responsible for managing certificates (Long-Term Certificate Authority (LTCA)). Authentication of communication is then mutual, unlike in usual web traffic where user checks the authenticity of the server, but the server does not have any interest in verifying the user.

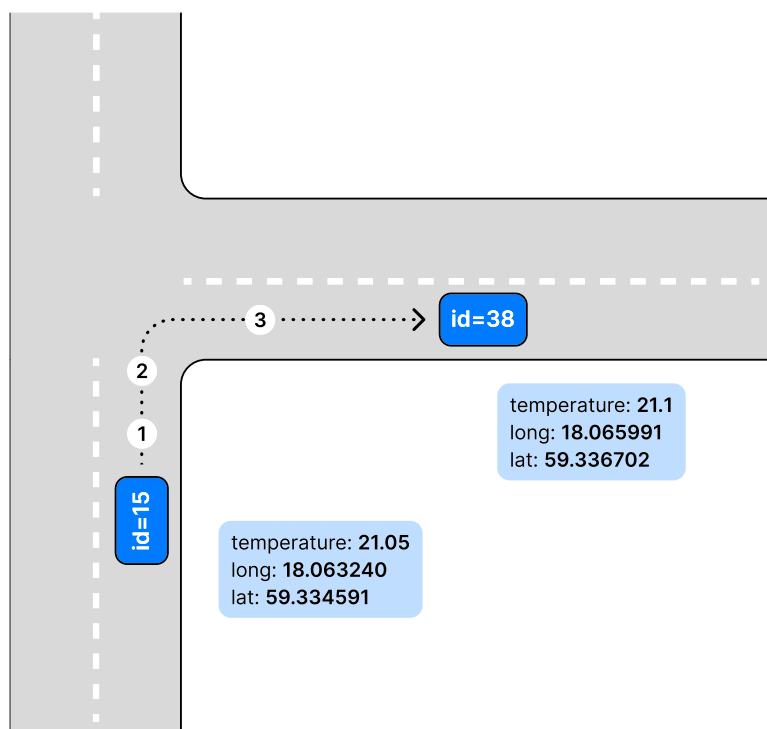


Figure 2.2: One of the mechanisms for preventing adversaries from linking user identities in PS systems is to use rotation of pseudonyms. In the presented example, the user has their ID changed conveniently in the middle of a street crossing. Therefore the sample from before entering and after leaving the crossing are submitted using a different identifier. If there happen to be more system participants using the same crossing, linking the identities of users before and after the crossing becomes significantly harder.

## 2.8 WebSockets

WebSocket API is a web-related technology allowing users to open two-way communication sessions between a client and a server. Unlike Hypertext Transfer Protocol Secure (HTTPS) is commonly used for one way communication from user to server, WebSockets allow server to push messages to the users.

WebSocket API has functionality that can resemble UNIX sockets, however, they are not related in use cases, API or implementation.

## 2.9 Bluetooth

Bluetooth was invented in 1989 and came into broader adoption around the year 2000 as a standard for short-range radio communication. There is no single Bluetooth protocol, but a group of protocols under one specification. It is managed by Bluetooth Special Interest Group (Bluetooth SIG).

Bluetooth operates using ad-hoc networks consisting of two or more devices. One device is central while the remaining are called peripheral. This ad-hoc network is called a piconet. Peripheral devices have a direct connection to the central device. A Bluetooth device might join several piconets simultaneously.

Bluetooth devices are assigned unique Media Access Control (MAC) addresses. However, they can choose to not share it. As an example, iOS devices advertise random MAC address until they are paired with another device, as a method for increasing privacy.

## 2.10 MPC

Secure Multi-party Computations [26] is a field of cryptography researching methods to compute functions in a way where multiple parties contribute inputs, which remain secret, while all participants have access to the result.

The beginning of multi-party computation is attributed to Andrew Yao[27], who suggested a method for secure two-party computation as a solution for the Millionaire's Problem.

Since then, there were many schemes proposed. Many of them focus on solving circuit logic problems, while others are based off Shamir's Secret Sharing[28], including this protocol.

Relevant problem domain is considered in the Fully Homomorphic Encryption (FHE), where users are allowed to perform computations on encrypted data without decrypting it first[29]. While MPC suffers from a large overhead in communication, FHE requires significantly more computations, making this method impractical for larger operations in current state of the art.

## 2.11 Group division

The situation of the system described in this thesis can be represented as a directed graph where edges symbolize that one user (source node) found another user (target node) through a BLE peripheral scan.

Constraints for the group splitting process is to keep group size between  $MIN$  and  $MAX$ , where those parameters are decided by the PM.

The goal is to find cliques, maximal complete subgraphs not subsumed by any other subgraph that will serve as groups.

Since the problem of finding cliques is NP-complete [30], this derivative presumably also is.

# Chapter 3

## Method

### 3.1 System model

The system model follows typical Participatory Sensing System architecture. In particular, compatibility with SPPEAR [8] system is ensured.

The full SPPEAR system has a scope significantly broader than this thesis, including, for example, the problem of incentivizing users to participate and administering tasks.

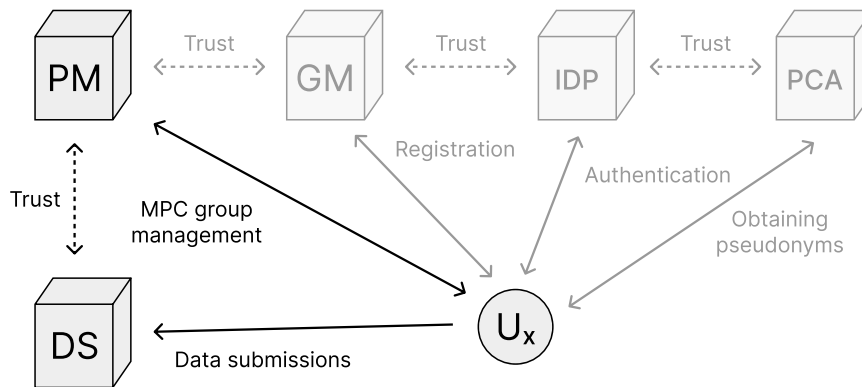


Figure 3.1: Diagram of the SPPEAR system entities, including addition of PM in the protocol. Highlighted entities are the ones present in the proof of concept implementation of the protocol. Although the other ones are not present there, they remain an important consideration in all design decisions.

## 3.2 Protocol

The protocol encapsulates communications using different networking protocols. Namely: HTTPS, WebSocket, Bluetooth.

## 3.3 Entities

The system consists of several independent entities taking different roles in the system, based on separation of concerns.

**PM** – responsible for invoking experiments. Controls parameters of the system. Web server capable of establishing WebSocket connections and handling HTTPS requests.

- Maximum and minimum group size, based on:
  - Total number of users.
  - Signal strength.
  - Geometry (spatial positioning of users).
  - Conclusions from previous experiment outcomes.
  - Specifics of current task.
- Number of group representatives ("leaders").
- Group representative election method:
  - Reputation based – users with the most past submissions that were successful.
  - Volunteer based – users can prefer to, or avoid to be GR.
  - Random – randomly selected users.
- Maximum time to wait for group joining reports.
- Discarding policy:
  - If some representative does not report back collected data, PM decides if other results reported by the group are credible.

**DS** – collects data coming from users. Web server capable of receiving HTTP requests.

**Participants** – users equipped with mobile phones supporting version 5 of the Bluetooth protocol.

## 3.4 Security and privacy targets

Following the SPPEAR architecture, this protocol targets:

1. **Communication integrity, confidentiality, authentication** – communication within the system should be authenticated and protected from unauthorized access or alteration. Two parties exchanging information should have guaranteed the safety of their communication.
2. **Non-repudiation and accountability** – all system entities are responsible for their actions and cannot disprove having sent a message or any other action.
3. **Anonymity** – users are anonymous in principle and ideally no observer should be able to link the same user as a participant of two different groups. However, this anonymity might be conditional based on the implementation of the Bluetooth protocol on the device of the user.

## 3.5 Adversary model

In this work, I consider both internal and external adversaries to the system. However, the main concern is the honest but curious server, which is interested in exploiting access to the raw communication from users, trying to learn facts about individual participants.

External threats involve passive attacks such as eavesdropping, and traffic analysis, or active attacks such as replay and modification vectors. They are not the main interest of the protocol, but defending against them is a necessity dictated by the usage of network protocols [31].

Section 5.3 discusses simulated situations showing the resilience of the system to various degrees of penetration by the active adversaries intending to disturb experiment results.

### 3.5.1 Bluetooth security

As described in the 2.9 section, real Bluetooth MAC address of a device can be learnt by connecting devices.

The implication of this fact for the protocol is that in adversary model of 'Honest but curious server', the adversary will still not be able to track users because it's not part of the protocol to share MAC addresses with the server. Unfortunately, malicious users can potentially link the fact that they took part in an experiment with some users by tracking the real MAC addresses of connected devices.

However, as a matter of a privacy risk, this vulnerability is not significant, since users often carry multiple devices which are advertising BLE. This means that as a vector of attack, this method is already available even in a much broader scope.

### 3.6 Zones

The protocol introduces a notion of zones. They can be configured to one of two modes: polygon-based shapes as in 3.2 (for representing real-world divisions such as city districts) or latitude and longitude-based grid as in 3.3. They can be also turned off.

If zones are enabled, each group is associated with a single zone, even if some members are spread between multiple zones. Due to the proximity nature of the group splitting, it's impossible for users to occupy distant zones, and therefore associating each group with one zone is sufficient.

The zone of the group is determined by calculating the average position of users in the group using MPC. Such computed value is then reported to the PM and used to assign zone.

I considered other methods such as selecting random users to give up location, but there was no plausible solution that would protect both the server and the representatives from the learning location of the user.

### 3.7 Protocol operations

The protocol can be divided into several phases of execution. Each of them consists of multiple cryptographic, or otherwise potentially computationally expensive operations.

Registration	Group forming	MPC	Submission
signing, key generation	BLE scan, BronKerbosch*	Sample splitting, Exchange of BLE messages, sample addition	signing†

Table 3.1: Table of operations in different phases of the protocol.

\* – algorithm for group splitting (Section 4.3).

† – Only for individual submissions.

### 3.8 Procedures

There are multiple procedures which model the exchange of messages in the system in response to various events.



Figure 3.2: Demonstration of custom zones.

Users need the ability to join the system and report readiness to receive requests for data, which is implemented by *register* 3.8.1. PM, prior to expecting results distributes users into groups within *formGroups* 3.8.2. Users, provided with information about the group they are meant to join, follow *joinGroup* 3.8.3. If a determined group appears impossible to function due to problems with establishing a peer-to-peer connection with every other member, *checkConnection* 3.8.6 is used. If user loses connection, *onDisconnect* 3.8.4 is used to proceed. *onRemove* 3.8.5 is meant for situations following removal of user.

### 3.8.1 register

Executed for each user  $i$  connecting to the PM.

1. User generates pseudonymous ID  $p$  and sends it to PM (Figure 3.5):  
 $U_i \rightarrow PM : \{id : register, name : p\}$ .
2. PM replies with signed ID  $s$  (Figure 3.6).  
 $PM \rightarrow U_i : \{id : registerAck, signature : sign_{PMK}(p)\}$ .

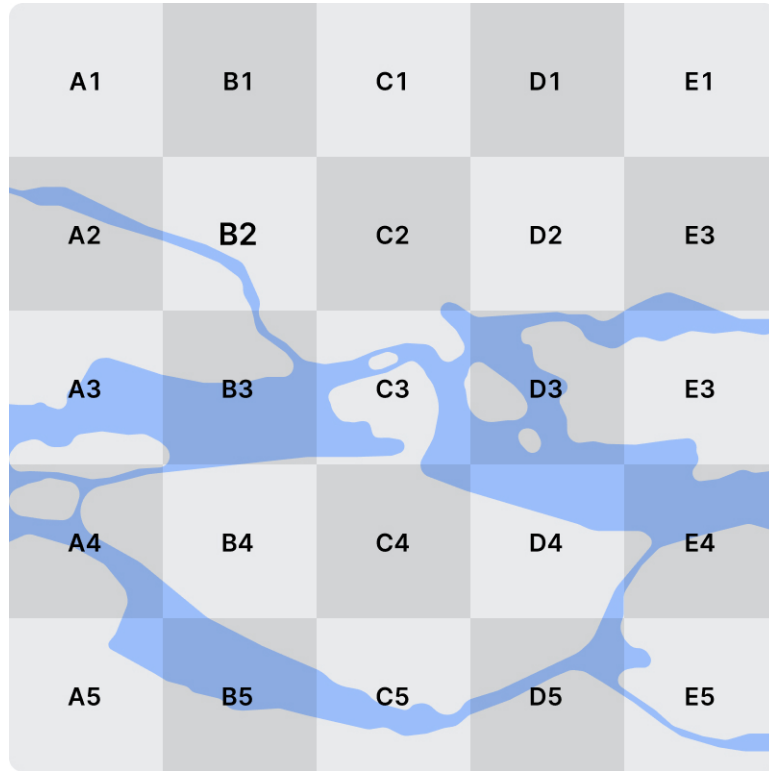


Figure 3.3: Demonstration of grid-based zones.

### 3.8.2 formGroups

PM sends request to all users to form groups.  $t$  is specified timestamp (1s later for example) when users are supposed to take final snapshot of the scanned data.

1.  $PM \rightarrow U_{1,\dots,n} \{id : formGroups, timestamp : t\}$
2.  $U_x \rightarrow PM : \{id : formGroupsAck\}$
3. Each user:
  - (a) Starts broadcasting their pseudonym (Figure 3.7).
  - (b) Scans surroundings for BLE (Bluetooth Low Energy) peripherals.
  - (c) Prepares a list of detected user ids at time  $t$ .
  - (d) Reports with the list of detected users (Figure 3.8):  
 $U_x \rightarrow PM : \{id : detected, detected : [p]\}$ .
  - (e) PM confirms receiving list:  
 $PM \rightarrow U_x : \{id : detectedAck\}$ .
4. PM waits until first of the conditions becomes true:

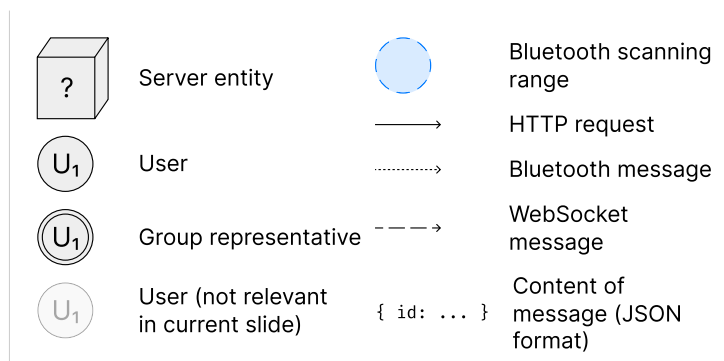


Figure 3.4: Legend of the figures included in this chapter. Server is represented by a box put in perspective view. User is a circle. Group representatives are additionally marked with an inner circle. Entities not relevant to the current step are presented in lower opacity. Arrows represent different connection methods, including HTTPS, BLE, WebSocket connections.

- (a) It receives all responses.
  - (b) 2 seconds pass from the timestamp  $t$ .
5. PM divides users into groups (Figure 3.9).
    - (a) PM splits users into groups by finding maximal complete subgraphs in size range between predefined  $MIN$  and  $MAX$ .
    - (b) PM invokes *joinGroup* 3.8.3 on all groups.
  6. Depending on receiving all user responses:
    - (a) If all arrive then PM finishes this procedure.
    - (b) If there is any response missing then PM excludes user from experiment and calls *checkConnections* 3.8.6 for their group. It repeats the process until state of all users is decided.

### 3.8.3 joinGroup

Executed for each potential group.

1. User receives group ID  $g$ , a list of group members  $m$  and a list of group representatives  $r$  (Figure 3.10):  
 $PM \rightarrow U_{1,\dots,n} : \{id : \text{groupInfo}, g, m, r\}$ .
2. Each user establishes peer-to-peer connection with all other group members (Figure 3.11).

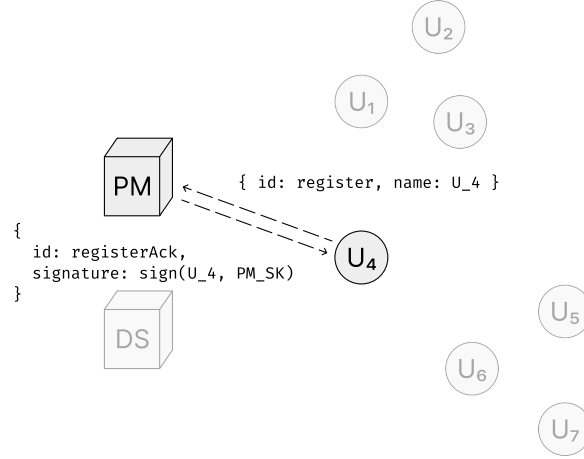


Figure 3.5: Each user registers to the PM by sending their user ID and receiving it signed.

3. User reports missing connections to the server (Figure 3.12):  
 $U_x \rightarrow PM : \{ id : groupReport, [u] \}$
4. If user reports empty list, PM continues:  $PM \rightarrow U_x : \{ id : groupReportAck \}$ 
  - (a) If all users report empty lists, group is considered as successfully formed.
  - (b) If there are some not connected users, PM invokes *checkConnections* 3.8.6.

### 3.8.4 onDisconnect

When user loses connection to another user while experiment is active, they try 3 times to re-establish connection. If this fails they report problem to PM. If connection is re-established, no action is needed.

1. If *fixGroup* 3.8.7 is currently under execution, abort.
2. 3 times:  $U_a \rightarrow U_b : \{ id : ping \}$ .
3.  $\forall i \in G : U_i \rightarrow PM : \{ id : missing \}$ .
4.  $\forall i \in G : PM \rightarrow U_i : \{ id : missingAck \}$ .

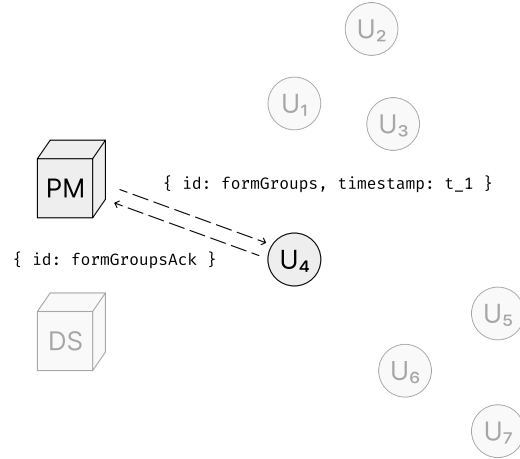


Figure 3.6: PM begins group-forming procedure by sending request to each user. Timestamp  $t_1$  is point in time (2 seconds in future) when users are supposed to take snapshot of devices detected until then and report that list to the PM.

### 3.8.5 onRemove

Executed when user is removed from a group, for example, as a result of losing connection to the PM.

1. PM removes information about user's pseudonymous ID and removes group association.
2. PM notifies group members that user is removed from a group:  
 $\forall i \in G : PM \rightarrow U_i : \{id : userRemoved, userId\}$ .

### 3.8.6 checkConnections

Used for removing users from a group if they cannot establish connections.

1. PM calculates how many other users can't see each user in a group.
2. If the number is greater than 1, the user is removed from the group.
3. If the number equals one, it means that there is a situation where two users have problem seeing each other but still see other users. In this case both users are removed.

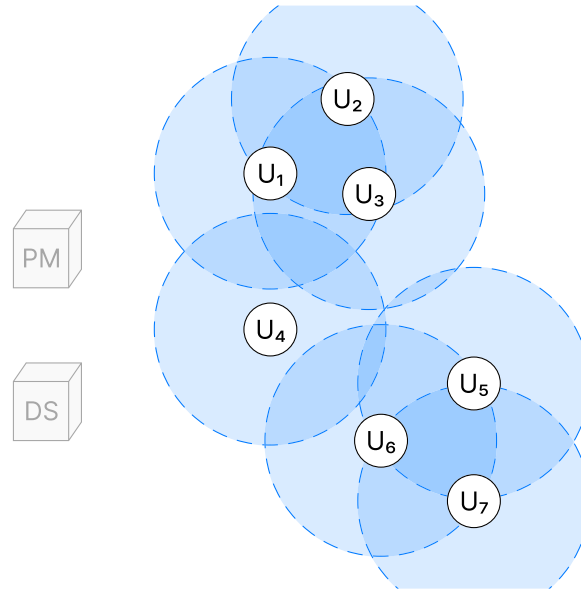


Figure 3.7: Users start advertising as BLE peripherals and scanning.

4. If after removing users the number of group members falls below minimum group member limit, group is dismantled.

### 3.8.7 fixGroup

This is run following *onDisconnect* 3.8.4 reporting problems.

1. PM asks each user  $i$  for list of users they cannot see:  
 $\forall i \in U : PM \rightarrow U_i : \{id : \text{sendMissingList}\}.$
2. Each user sets a “lock” which means that won’t attempt *onDisconnect* 3.8.4 until this procedure is over.
3. Users reply with lists:  
 $\forall i \in U : U_i \rightarrow PM : \{id : \text{missingList}, [u]\}.$
4. PM invokes *checkConnections* 3.8.6.
5. PM invokes *joinGroup* 3.8.3 on remaining users.
6. The “onDisconnect lock” is released.

### 3.8.8 swapRepresentatives

As a security measure, PM can change representatives of a given group. If previous set of representatives consisted entirely of malicious users and they attempted to

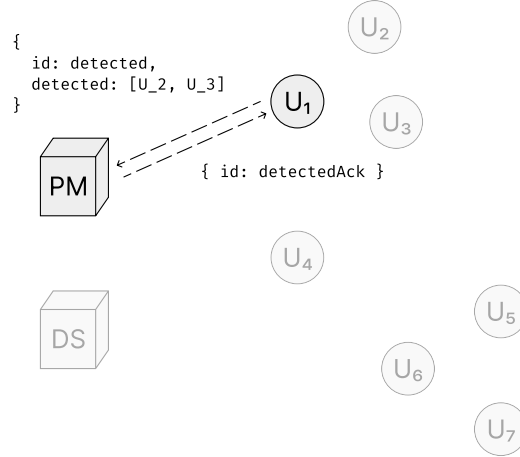


Figure 3.8: Each user sends list of detected users to the PM and receives confirmation.

submit incorrect result, the new representatives might uncover them (or, what is also possible, new representatives might be adversaries and undermine the effort of benign users).

1. PM decides that in a group  $G$ , representatives must be swapped. PM elects new representatives by randomly electing ones that do not have this role currently. If this is not possible, representatives might repeat.
2. PM notifies group members of the change:
 
$$\forall i \in G : PM \rightarrow U_i : \{id : representativesChanged, [representatives]\}.$$
3. Users reply with acknowledgement:
 
$$\forall i \in U : U_i \rightarrow PM : \{id : representativesChangedAck\}.$$

### 3.8.9 runTask

PM when it wants to execute a task.

1. Invoke *formGroups* 3.8.2.
2. PM sends request to users participating to take sample at timestamp  $t$  (Figure 3.13):
 
$$PM \rightarrow U_{1,\dots,n} : \{id : takeSample, timestamp : t\}.$$
3. Each user takes sample  $X$  (Figure 3.14).

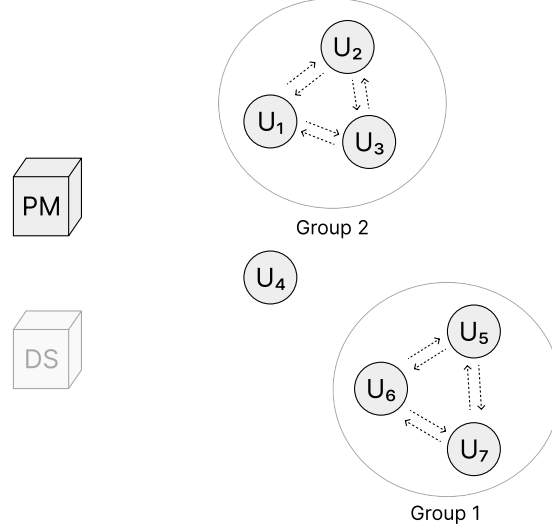


Figure 3.9: PM analyses the directed graph of scanning detections and calculates cliques (subgraphs where all vertices are connected).

4. Users begin MPC calculation.
  - (a) Each user  $i$  splits sample  $x$  into  $n - 1$  subsamples  $x_{i1}, \dots, x_{i(n-1)}$  in such way that  $\sum_{i=1}^n x_{ij} = X$ , where  $n$  is size of the group.
  - (b) Each User  $i$  sends samples to each user  $j$ , skipping  $x_{ij}$ .  
 $\forall i \in U : U_i \rightarrow U_1, \dots, U_{m-1} : \{id : x, x\}$  (Figure 3.15).
  - (c) Each user  $i$  calculates  $n - 1$  values  $s_j = x_{1j} + \dots + x_{nj}$ , skipping  $x_{ij}$ .
  - (d) Then, each user  $i$  sends their  $s$  values to group representatives (Figure 3.16).  
 $\forall i \in U : U_i \rightarrow U_{1, \dots, k} : \{id : s, s\}$ .
  - (e) Each representative  $r \in R$  sums the received values and report to DS (Figure 3.17).  
 $\forall r \in R : U_r \rightarrow DS : \{id : data, groupId, e, n, value, p, s\}$ .
  - (f) After sending sum to the DS, representatives inform each user  $i$  in their group that experiment concluded (Figure 3.21):  
 $\forall r \in R : U_r \rightarrow U_i : \{id : done\}$  (Figure 3.20).
  - (g) Users disconnect their Bluetooth connections.
5. If some user decided to submit data individually, the request they send has the following format (Figure 3.18):  
 $U_k \rightarrow DS : \{id : data, e, v, p, s\}$ .

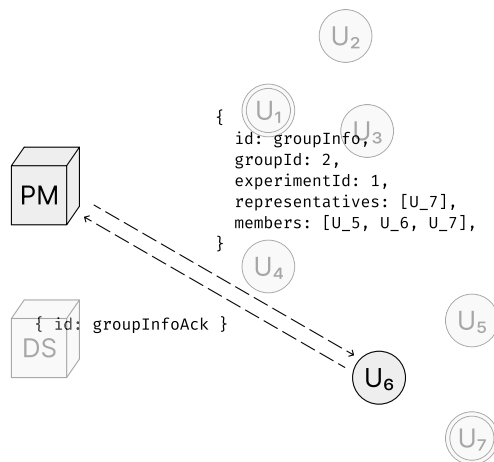


Figure 3.10: User receives information about the group and experiment the PM assigned them to. It includes a list of members of the group and list of group representatives.

6. DS obtains samples.
  - (a) If in some group there are less received samples than group leaders, then act according to PM policy.
7. DS informs PM that experiment is concluded.
8. PM discards all information about groups.
9. DS calculates the result of the experiment (Figure 3.21).

### 3.9 Example use case

An example of a system using the protocol might be a PS system measuring the heart rate of users across Stockholm city. The scientific value is a search for correlation between heart rate measurements in different districts of the city.

Due to the sensitivity of medical data, the team prefers to not directly process raw user measurements to avoid the risk of data leak in an event of a breach. Because the data samples do not reach the server in their original form, there is no possibility that any malicious actor can track any participant in the experiment. This simplifies the problem of management of the server and data, especially in accordance to General Data Protection Regulation (GDPR) or other privacy-protecting laws.

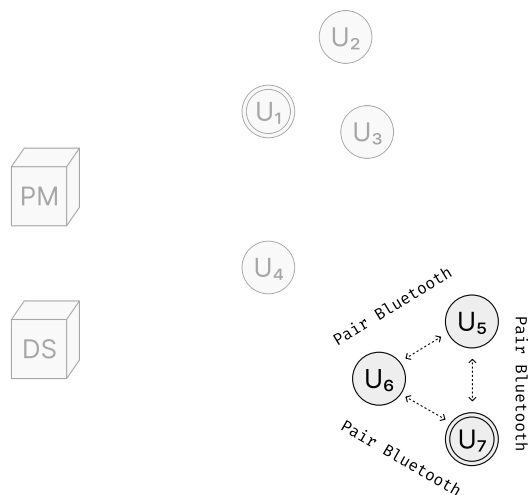


Figure 3.11: Users establish peer to peer connections in groups.

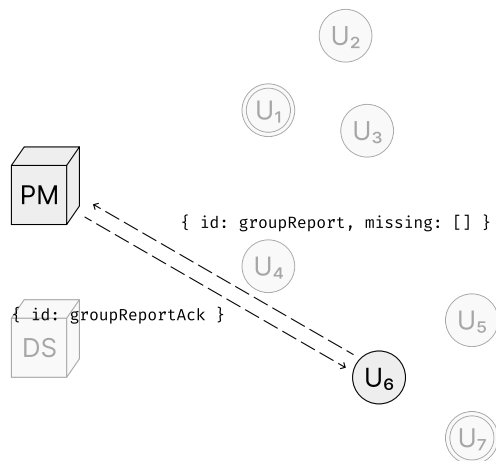


Figure 3.12: Each user reports users with which they could not establish connection.

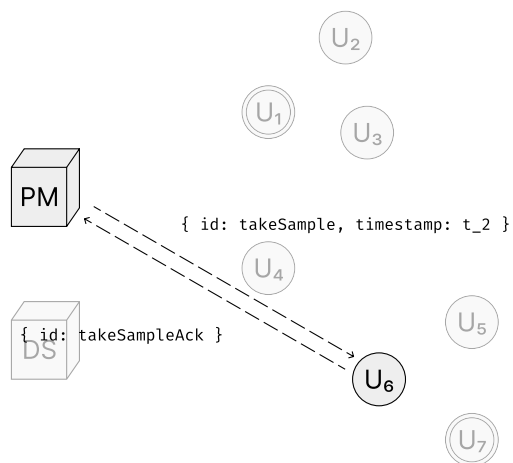


Figure 3.13: PM asks users to take samples at time  $t_2$ .

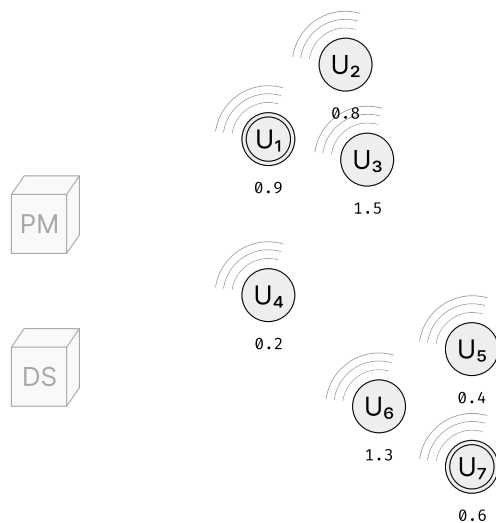


Figure 3.14: Users collect data at time  $t_2$ .

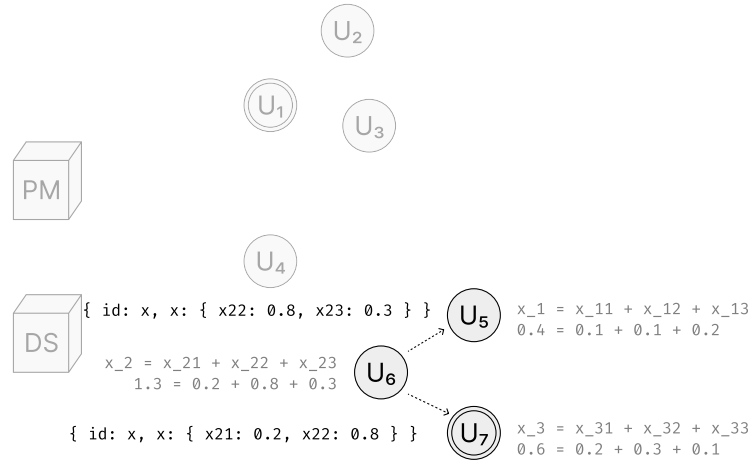


Figure 3.15: Each user  $i$  in a group of  $n$  users splits the sample into  $n - 1$  subsamples  $x_{i1}, \dots, x_{i(n-1)}$  and sends to each other user  $j$  all samples but skipping  $x_{ij}$ .

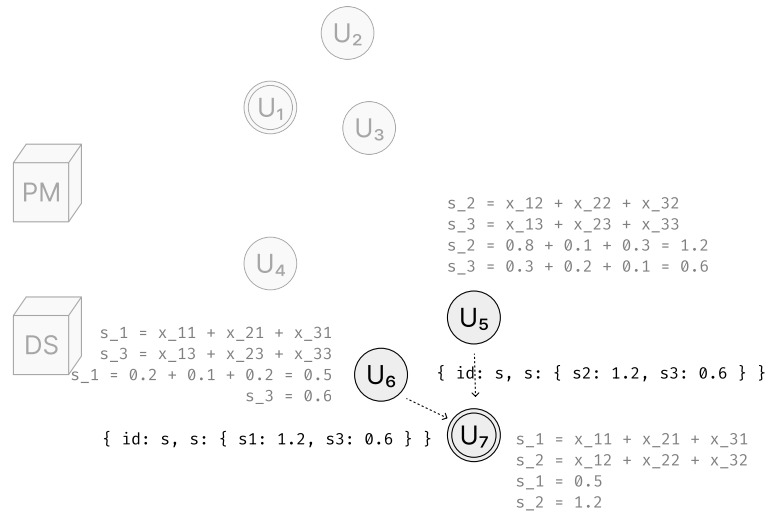


Figure 3.16: Each user  $i$  calculates  $n - 1$  values  $s_j = x_{1j} + \dots + x_{nj}$ , skipping  $j = i$ . Then, each user  $i$  sends their  $s$  values to group representatives.

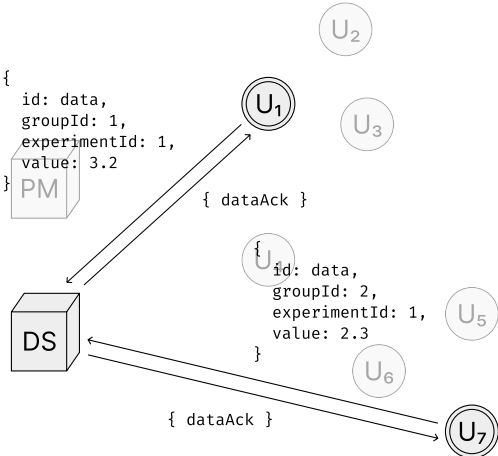


Figure 3.17: Group representatives report results of their groups to the DS.

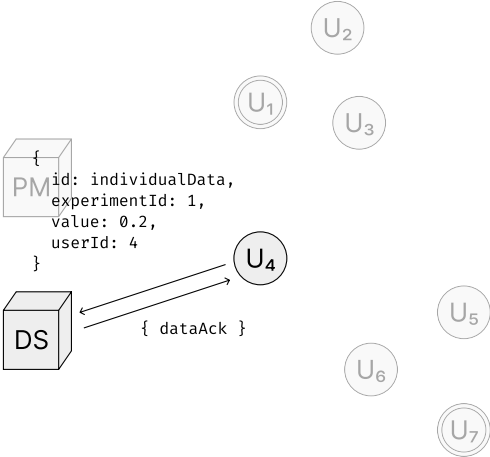


Figure 3.18: Users that were not assigned to group can, based on their preference and policies of PM, provide data individually (compromising their privacy).

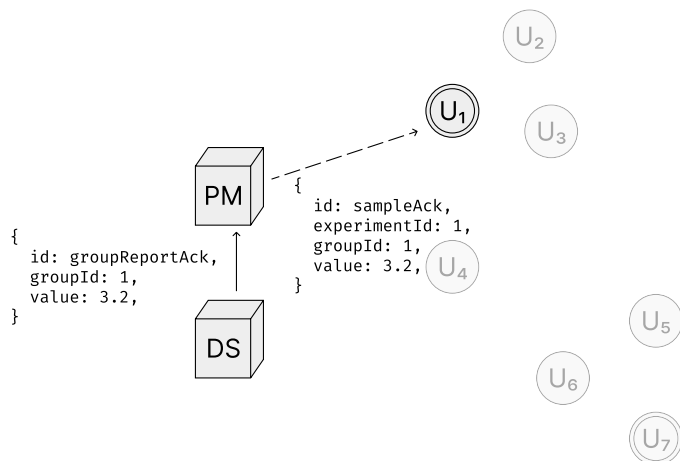


Figure 3.19: DS confirms to PM that the group reported their result. PM then notifies each user in the group.

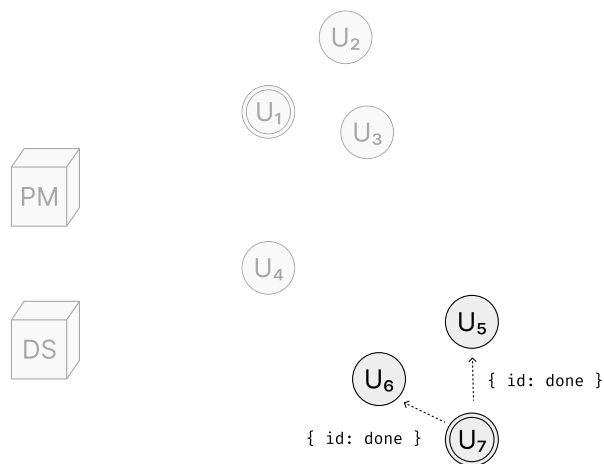


Figure 3.20: Group representatives inform users that experiment is over. Users stop their Bluetooth connections.

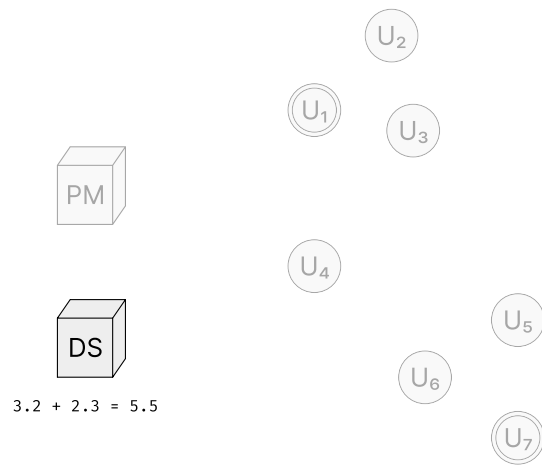


Figure 3.21: DS obtains final result by summing partial ones.

# Chapter 4

## Implementation

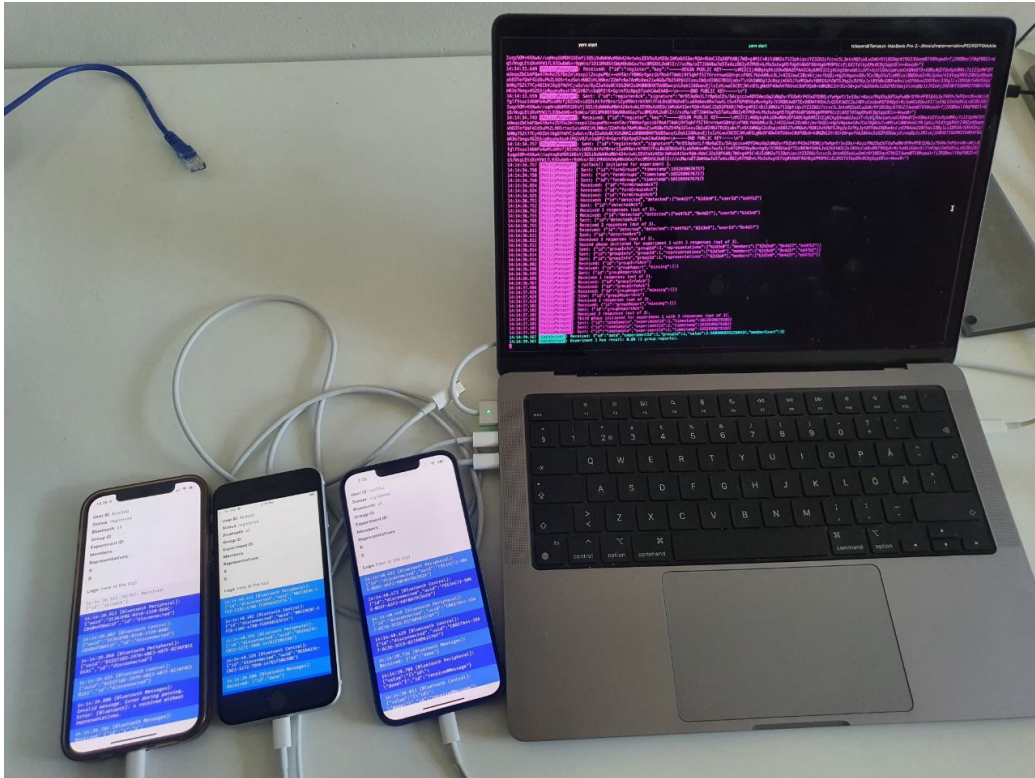


Figure 4.1: Example picture of a working system with 3 mobile devices contributing to an experiment and one laptop acting as a host of PM and DS.

The project was implemented in a top-down philosophy, aiming to achieve fully functional modules across the whole protocol stack, while not focusing, for instance, on broad platform support.

## 4.1 Mobile app

The mobile app is implemented in React Native[32] framework. Therefore, it supports both iOS and Android platforms, and with minor adjustments can support desktop platforms like Windows and MacOS (which is potentially important as laptops have Bluetooth hardware support and can take part in the MPC phase the same way as phone users).

The platform-specific Bluetooth module is implemented for iOS platform using BluetoothCore[33] platform library.

### 4.1.1 React Native

React Native is a mobile development framework initially created in 2015 to allow for building mobile apps for Android and iOS using shared JavaScript code.

The simplified idea of the framework is that the resulting app includes a JavaScript compiler (either Apple's JavaScript Core[34] compiler or Facebook's Hermes[35]) and bundle with the JavaScript code. After the app starts, the compiler initializes, compiles the JavaScript code and starts the application. Communication between the JavaScript thread and the parts native to the platform is done mainly via the so-called bridge[36], which is an asynchronous communication channel.

Nowadays React Native offers support for additional platforms, such as Windows and MacOS (as mentioned above)[32].

## 4.2 Servers

Each server, PM and DS, is implemented as NodeJS[37] server. ExpressJS library is used for handling HTTPS connections, while `ws` library is used for the WebSocket server management.

There is another, third server implementation, device emulator, which can be used for emulating the whole system within one machine for testing purposes, and was used to accelerate development in early phases. It similarly uses `ws` library as a WebSocket client and does not participate in the MPC computations, only pretends so for testing purposes.

## 4.3 Group division

As an approximation, I use Bron-Kerbosch algorithm for finding cliques in an undirected graph [38], as shown in 4.1.

---

Listing 4.1: BronKerbosch.

```

1  func BronKerbosch(R, P, X):
2      if P and X are both empty then
3          report R as a maximal clique
4          choose a pivot vertex  $u \in P \cup X$ 
5          foreach vertex  $v \in P \setminus N(u)$  do
6              BronKerbosch( $R \cup \{v\}$ ,  $P \cap N(v)$ ,  $X \cap N(v)$ )
7               $P \leftarrow P \setminus \{v\}$ 
8               $X \leftarrow X \cup \{v\}$ 

```

---

Following that, function 4.2 shows next parts of the algorithm. I sort resulting graphs by size from smallest to biggest, and in each graph, sort them alphabetically by labels of nodes.

As nodes can be assigned simultaneously to multiple cliques, the algorithm continues with the loop which removes nodes from a clique if it appeared previously. Because of the sorting, nodes are meant to stay in smaller cliques, to not risk falling below the minimum size, while being removed from bigger cliques. Finally, cliques that fail to reach size *MIN* are discarded, while cliques that have a size exceeding *MAX* are further split into smaller cliques.

Listing 4.2: SplitGroups.

```

1  func SplitGroups(G):
2      C ← BronKerbosch({}, G, {})
3      C ← sort(C)
4      A ← {}
5      foreach clique  $c \in C$  do
6          R ← {}
7          foreach  $v \in c$  do
8              if  $v \notin A$  then
9                   $R \leftarrow r \cup \{v\}$ 
10             if  $|r| < MIN$  then
11                  $c \leftarrow []$ 
12             return
13         foreach  $v \in R$  do
14              $A \leftarrow A \cup \{v\}$ 
15             if  $|R| > MAX$  then
16                 SplitIntoSmaller(R)
17             return
18          $c \leftarrow [R]$ 
19         return

```

---

Splitting cliques is responsibility of the SplitIntoSmaller function, as shown in 4.3.

Listing 4.3: SplitIntoSmaller.

```

1  func SplitIntoSmaller(G):
2      if  $|G| \leq MAX$  then
3          return [G]
4
5      S ← []
6
7      for  $i; i < \lfloor \frac{|G|}{MIN} \rfloor; i += 1$  do
8           $s+ = MIN$ 
9
10     if  $|G| - |S| \cdot MIN < MIN$  do
11          $S[|S| - 1] += |G| - |S| \cdot MIN$ 

```

---

## 4.4 Random number splitting

Division of samples into random numbers is done the following way: for each of the required  $n - 1$  parts, there is a random unsigned 32-bit integer generated, through hardware accelerated generation of random bytes. Then the final part is calculated

$$\text{as } p_n = s - \sum_{i=1}^{i < n-1} p_i.$$

# Chapter 5

## Results

The final phase of the thesis included an evaluation of the performance of the implementation and simulations verifying the resilience of the protocol.

### 5.1 Overall system performance

To evaluate system performance, I executed the full system flow 10 times. Test devices were: iPhone SE 2020, iPhone 13, iPhone 12 Pro and as the host for servers: MacBook Pro M1 32GB RAM.

Results are presented in the Figure 5.1.

1. *Registration* is the time between the user sending a request to register and receiving signed credentials.  
*Form groups signal* is the time between receiving and confirming group formation request from the PM.
2. *BLE advertising* is the time between the start of advertising as BLE peripheral and the last discovery of a new user within the range.
3. *Finalizing groups* is the message exchange between user and PM including reporting discovered devices, receiving instructions on who to connect to.
4. *BLE connection* is the time between attempting the first connection to another user and the moment when the last user becomes available for receiving messages.
5. *Message exchange* is the time spent on sending  $x$  and  $s$  values to conduct MPC calculation.
6. *Submission* is the time between sending HTTPS request to the DS and receiving response.

Registration can be expected to be a very quick process (with occasional reasonable slowdowns determined by, for example, network conditions). Group forming signal is a very quick procedure lasting between 20 and 30ms.

Bluetooth advertising is most of the time in a range of 120 to 180ms, with occasional outliers possible due to temporal lack of availability of access to system Bluetooth APIs, however, system is correctly recovering each time and none of the runs required a retry.

The phase of finalizing groups takes between 250 and 400ms, due to the synchronization of messages from all group members and the time needed to execute the group division algorithm.

Establishing peer-to-peer connection in the system lasts between 800 and 1000ms and is by far the most time-consuming process in the system.

Message exchange, which means the full MPC execution, lasts between 300 and 500ms.

Submission, as a simple HTTPS request, is relatively short and takes 100 to 225ms.

Phase	min	25th	mean	75th	max
Submission	52	99	195	224	438
Message exchange	246	296	390	481	579
BLE connection	637	824	948	1031	1537
Finalizing groups	243	254	351	376	660
BLE advertising	85	122	251	182	1253
Form groups signal	21	22	28	31	52
Registration	40	59	143	74	853

Table 5.1: Result of executing the system flow 10 times. Presents minimum value, 25th percentile, mean, 75th percentile and maximum. The unit is milliseconds.

### 5.1.1 Observations

Due to the resource constraints, available data does not reflect the expected situation in the system but merely presents a minimal case. However, with this information in mind, several observations can be done about the performance of the system.

Group forming, from the request originating from PM to the moment when groups become fully operational, takes 1200 to 1600ms of operations and additional timeouts for synchronization of phases, which depend on the protocol configuration.

The core part of calculating results is the MPC execution with the expected duration spanning 300 to 500ms. This means that for an experiment with multiple submissions, this can be the expected interval of processing samples. Group size

can possibly reach up to 8 users, which means less than a triple than in this experiment. It is hard to estimate an increase in the cost of sending triple the number of messages, as the message sending itself is asynchronous, but I expect it the increase to be lower than linear.

As the time for MPC execution does not depend on the total size of the system but rather the sizes of groups, I expect 1-second interval between result submission to be feasible.

## 5.2 Impact of group representative optimization

The total number of messages that need to be exchanged in the protocol can be precisely calculated.

Several important symbols:

- $g$  is number of group participants,  $\lfloor \frac{n}{s} \rfloor$ .
- $s$  is the size of each group (assuming they are equal size, for this particular calculation).
- $n$  is the total number of users.
- $k$  is number of group representatives in each group ( $k < s$ ).

The calculation:

The initial setup produces  $12n$  messages. Exchanging  $x$  values leads to  $gs(s-1)$ . Following that,  $s$  values bring  $g(s-k)k$ . Users who upload alone contribute another  $n - sg$ . Group representatives informing DS about results send  $2kg$  messages. Group representatives informing users that cooperation is done take  $g(s-k)k$  messages.

The final formula:

$$12n + gs(s-1) + gk(s-k) + n - gs + 2gk + gk(s-k) = 13n + gs(s-2) + 2gk(s-k+2)$$

Calculations for different  $n$ ,  $s$  and  $k$  are illustrated on a chart in the Figure 5.1.

It clearly shows that the optimization was successful as it greatly decreases the factor of growth for the function.

However, as presented in Table 5.2, the overhead of sending messages has minimal influence on the protocol execution time, as most of the time is spent on a Bluetooth connection.

$k$ (out of 3)	Time (ms)
1	580
2	595
3	683

Table 5.2: Time difference between the point in time specified by the PM in *takeSample* message and the moment the last result reached DS.

### 5.3 Simulation of adversary detection methods

As part of the effort to evaluate the security of the system, I conducted simulations of different scenarios of attempts to identify adversaries in the system.

In the following scenarios, the adversary is a malicious user attempting to become a group representative in order to report the incorrect result to the DS. If there is more than one representative per group, adversaries wait for a situation in which all representatives are adversaries.

It is important to note that the adversary cannot compromise the calculation as a regular user, as a two-phase calculation does not give enough power to a single user to consistently forge values to make the group believe the calculation was indeed correct.

However, it is also not possible to prevent individual users from providing arbitrary artificial values as their results and therefore this is not part of the adversary model in this simulation.

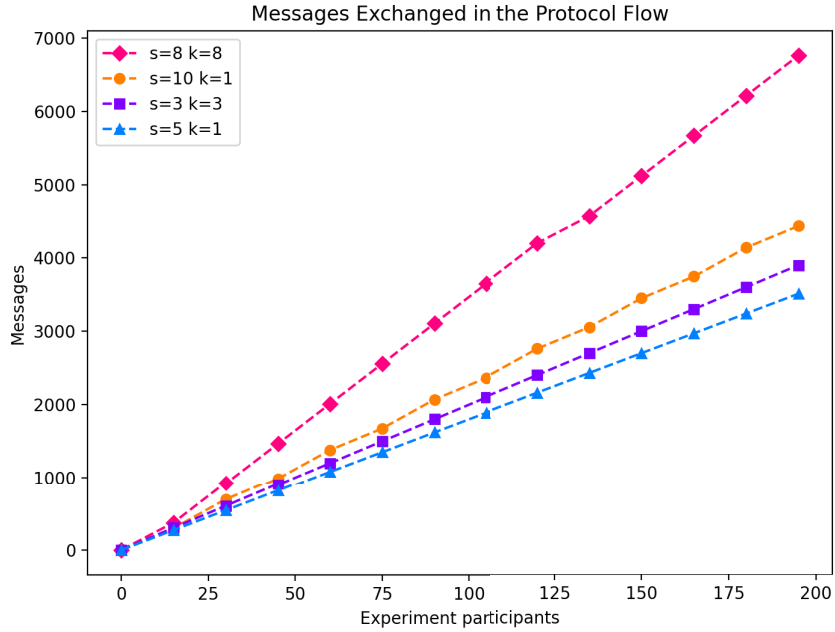


Figure 5.1:  $s$  is number of members in group.  $k$  is number of group representatives.

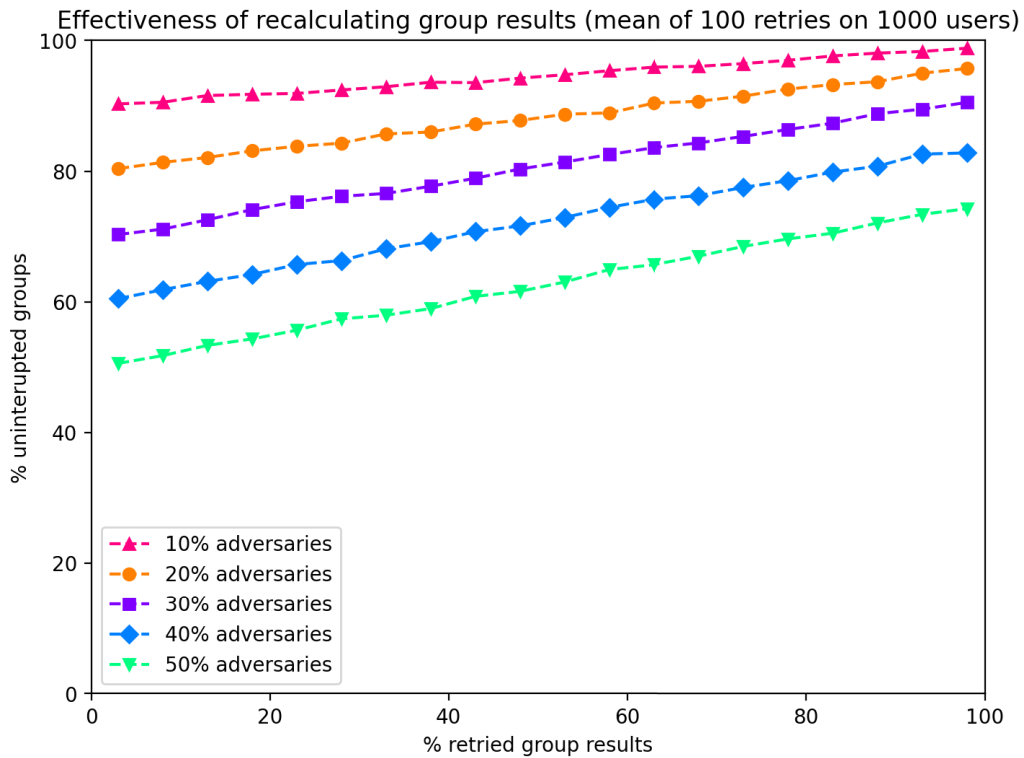


Figure 5.2: Retrying a selected percentage of experiments, given the specified amount of adversaries. It includes one submission (with optional retry).

The first discussed method is retrying randomly selected groups. PM selects some percentage of groups and in each of them, group representatives are switched. If possible, to a different one than in the previous submission. Both selections of groups and the following selection of new representatives is random. This guarantees that adversaries cannot rely on any pattern when deciding to attack.

As Figure 5.2 shows, retrying randomly selected groups is a method generally linear to the number of retried experiments. Although it never guarantees success, the newly elected representative might also be an adversary. The latter problem is increasingly severe with a rising percentage of adversaries in the system.

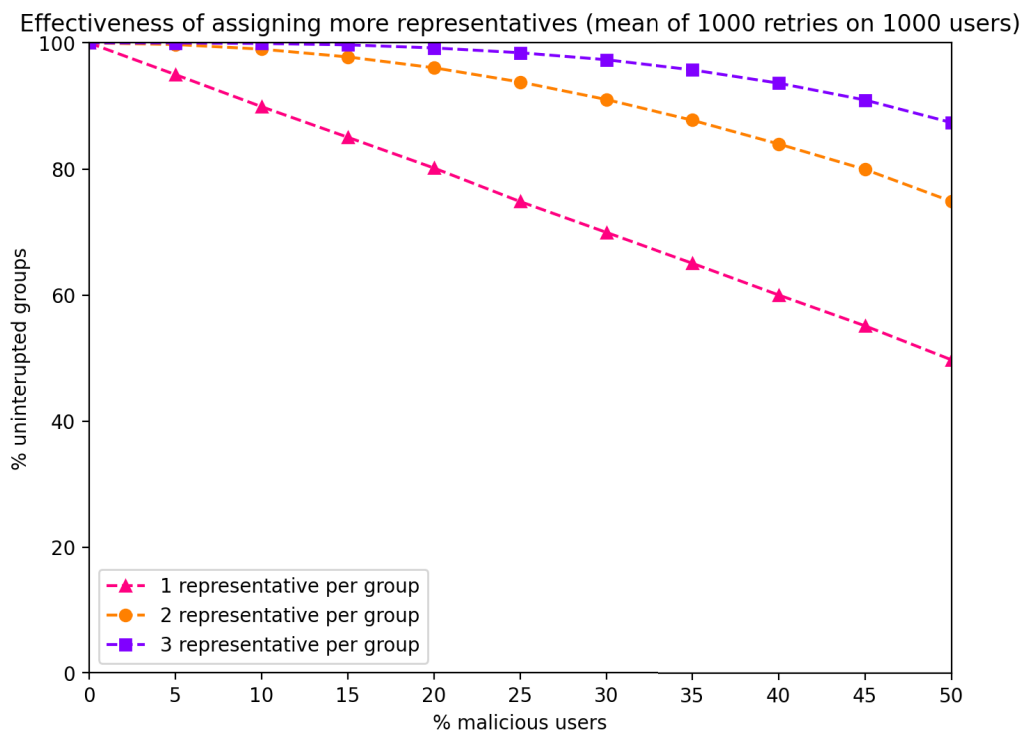


Figure 5.3: Electing more than one representative (two or three) makes the system more resistant to the increase in the percentage of adversary users.

An important tool for countering the risk of adversary attack is the number of representatives in each group. Just assigning more representatives naturally decreases the chances of adversaries compromising a group, as depicted in Figure 5.3. It is important to note that for one representative per group, the resilience decreases linearly with an increase of the percentage of malicious users. While for two and more representatives, the decrease is slower, resembling the square root function reflected on the X axis.

An increasing number of representatives also visibly impacts effectiveness of retries, as in Figure 5.4. System with 50% of adversaries which has two representa-

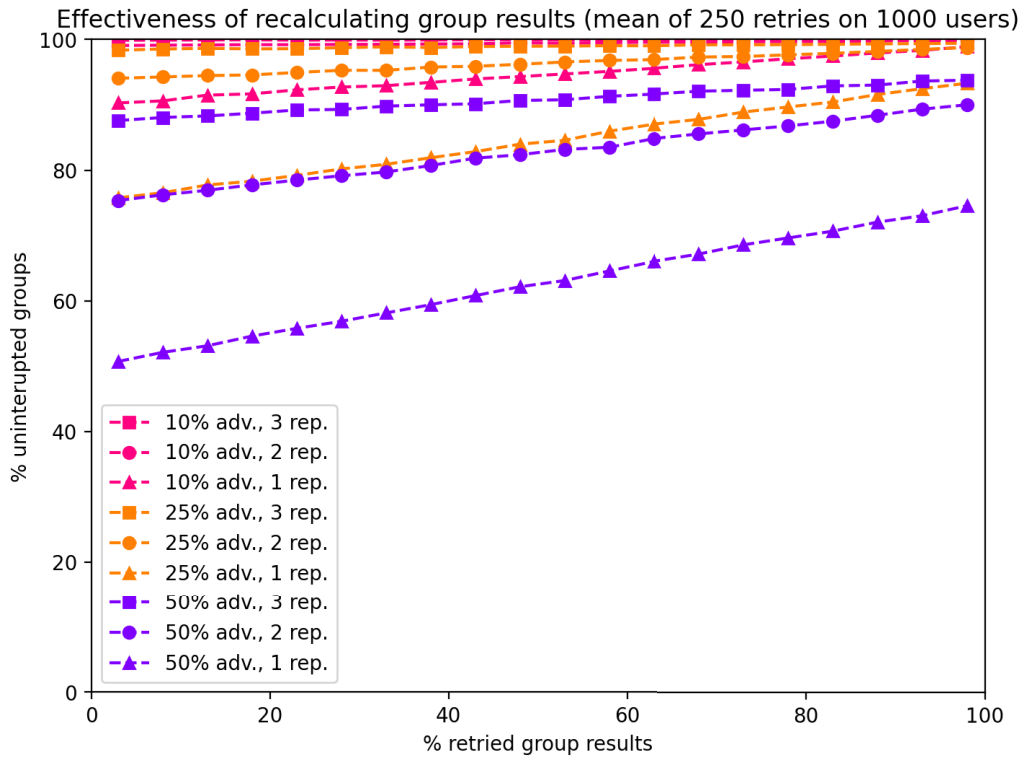


Figure 5.4: Effectiveness of catching adversaries appears to be linear to the increase of retry percentage.

tives per group, responds to an increase of retried experiments in a similar fashion to a system with 25% of adversaries and one representative per group.

Unsurprisingly, the success rate in evicting adversaries through retries grows linearly to the number of retries, as shown in Figure 5.5. The biggest impact on the number of caught adversaries is the percentage of users who were adversaries. The more adversaries the lower the possible eviction rate.

Another property of the system that aids in preventing attacks is ability to reuse groups for multiple submissions. Assuming 20% of users being malicious, it is possible to evict 80% of them within 20 submissions, as presented in Figure 5.6. It is worth noting that 20 submissions, depending on the experiment, might take as little as 20 seconds. This shows that any fraction of repeated experiments can have

### 5.3.1 Observations

Presented experiments show that there are three notable countermeasures against adversaries targeting group submissions: retrying a selected subset of experiments,

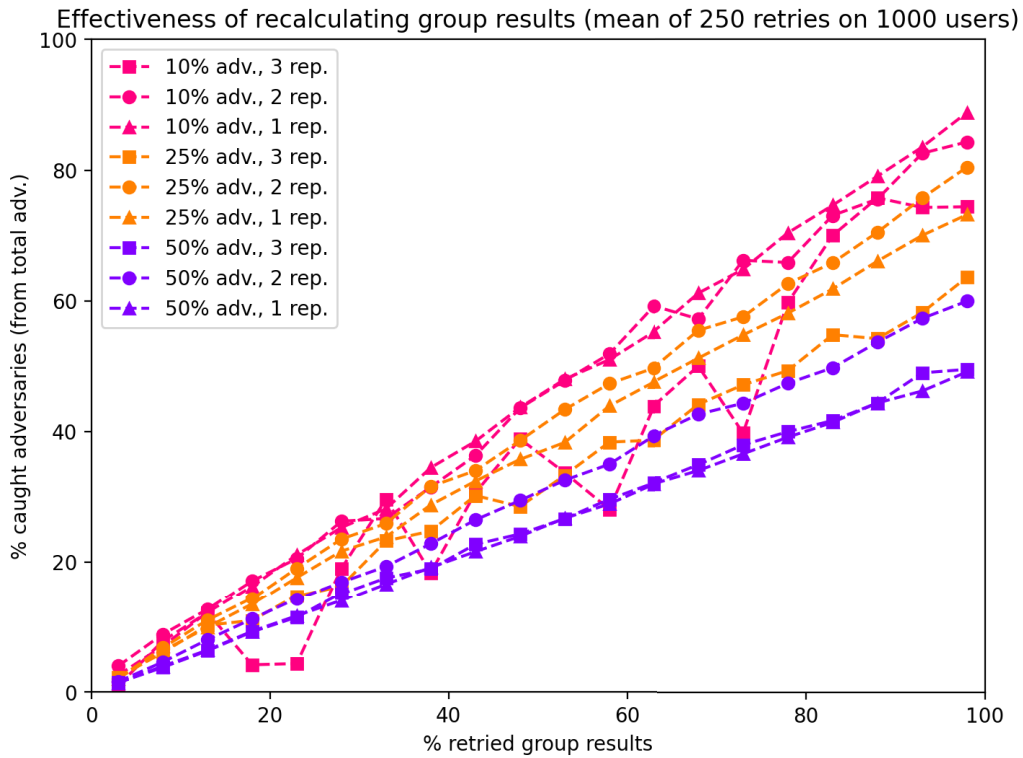


Figure 5.5: The chart presents effectiveness of recalculating group results as a solution for catching adversaries. It shows that effectiveness of evicting malicious users rises with the number of representatives, as it becomes harder to compromise a group.

increasing the number of group representatives, and increasing the number of submissions.

The number of submissions is a parameter solely depending on the specifics of the conducted experiment.

The number of group representatives in each group is a compromise between safety and optimization of message passing. The fewer representatives, the fewer messages need to be exchanged in the final phase of the protocol, not affecting accuracy in any way. However, as shown by the simulations, each group representative greatly increases the resilience of the group.

Finally, retrying experiments is an interesting approach which provides an additional tool for increasing security measurements in case other methods were used to the highest available extent. However, the effectiveness of this method on one-time measurements is linear to the percentage of retried experiments. This means that there is no natural hint on the value of this setting and in each case it needs to be arbitrarily selected.

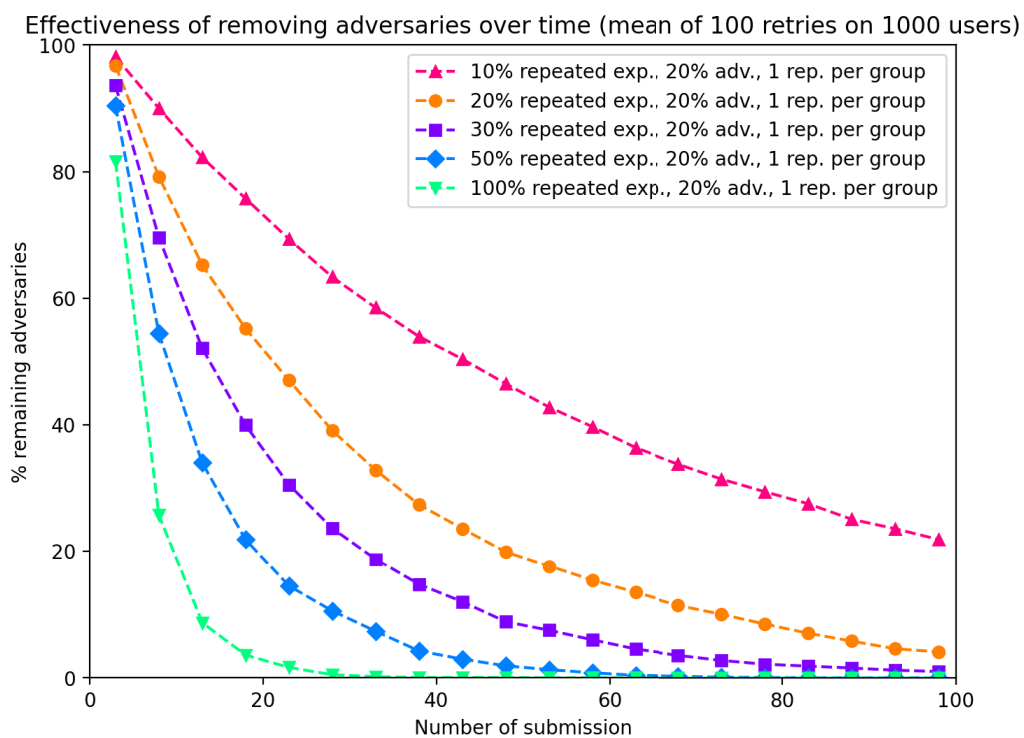


Figure 5.6: Effectiveness of evicting adversaries over multiple submissions.

On the other hand, removing adversaries over multiple submissions proves to be very effective, leading to the nearly total eradication of adversaries in any range between 20% and 100% retries over 100 submissions.

## Chapter 6

# Conclusions

This thesis project resulted in design and implementation of a protocol that allows users to share sensed data in a privacy-preserving manner. It comes with a considerable cost in terms of execution time and computational cost, but none of it exceeded reasonable expectations coming from the use of MPC and Bluetooth.

The conducted measurements show that the system initialization is expected to last 1-2 seconds for entities to be fully operational, and the feasible submission interval is expected to be 1 second.

The resulting protocol features a large communication and computation overhead over simple data submission. However, it offers large data privacy benefits that would otherwise be unavailable in simpler solutions. Additionally, this overhead is effectively insignificant in a scale of all operations and connections that modern smartphones are managing. In a real-world scenario, usage of the protocol should be unnoticeable to users even as background calculations, and achieved submission interval should be more than satisfying for the suggested medical data use cases.

### 6.1 Future work

There are several aspects that would benefit from future attention.

The algorithm for group division can be optimized based on further study of the NP-completeness of the underlying issue and possibly utilize a better heuristic algorithm.

Alternative modes of transport, apart from Bluetooth, could potentially offer lower delays, especially in comparison to rather lengthy Bluetooth device pairing.

The number splitting algorithm would benefit from proper formal analysis of the current and target quality of randomness.

As in most distributed systems, the protocol operates on time limits that require certain stages of execution to finish within specified time frames. As of now, those time limits are pre-calculated and constant between all protocol executions.

This particular aspect could benefit from adaptive timeouts and methods such as exponential backoff.

## 6.2 Sustainable development

The whole concept of the suggested protocol involves a heavy communication overhead compared to the most simple data aggregation. This involves cost in network traffic, congestion, the need for more computations, and use of cryptographic algorithms for verification of credentials.

I took steps to counteract the increase of cost, mainly the optimization coming from the concept of group representatives which lead to a decrease in the number of messages required to exchange. This reduces energy consumption on devices and decreases the number of messages required to be sent to DS, which can lead to less load and result in a simpler architecture that offers better scaling.

## 6.3 Ethical considerations

The aim of the project was to increase the privacy of users taking part in data collection for experiments. It comes with a potential risk of malicious actors taking advantage of the anonymity of their inputs. However, it is unavoidable and cannot be solved on a technical level if users are promised perfect privacy.

# Bibliography

- [1] Mykola Makhortykh, Aleksandra Urman, Teresa Gil-Lopez, and Roberto Ulla. To track or not to track: examining perceptions of online tracking for information behavior research. *Internet Research*, 32(7):260–279, December 2021.
- [2] Wi-Fi Aware. Accessed: 2022-07-22.
- [3] Multipeer Connectivity. Accessed: 2022-07-22.
- [4] Burke JA, Deborah Estrin, Mark Hansen, A Parker, R.Anitha Nithya, S Reddy, and M Srivastava. Participatory sensing. *Workshop on World-Sensor-Web (WSW): Mobile Device Centric Sensor Networks and Applications*, 01 2006.
- [5] Raghu K. Ganti, Fan Ye, and Hui Lei. Mobile crowdsensing: current state and future challenges. *IEEE Communications Magazine*, 49(11):32–39, 2011.
- [6] Hao Zhong and Kaifeng Bu. Privacy-utility trade-off, 2022.
- [7] Rim Ben Messaoud, Nouha Sghaier, Mohamed Ali Moussa, and Yacine Ghamri-Doudane. On the privacy-utility tradeoff in participatory sensing systems. In *2016 IEEE 15th International Symposium on Network Computing and Applications (NCA)*, pages 294–301, 2016.
- [8] Stylianos Gisdakis, Thanassis Giannetsos, and Panos Papadimitratos. Sppear: Security & privacy-preserving architecture for participatory-sensing applications. In *Proceedings of the 2014 ACM Conference on Security and Privacy in Wireless & Mobile Networks, WiSec '14*, page 39–50, New York, NY, USA, 2014. Association for Computing Machinery.
- [9] Leyla Kazemi and Cyrus Shahabi. A privacy-aware framework for participatory sensing. *SIGKDD Explor. Newsl.*, 13(1):43–51, aug 2011.
- [10] Emiliano De Cristofaro and Claudio Soriente. Participatory privacy: Enabling privacy in participatory sensing. *IEEE Network*, 27, 01 2012.

- [11] Emiliano De Cristofaro and Claudio Soriente. Extended capabilities for a privacy-enhanced participatory sensing infrastructure (pepsi), 2013.
- [12] Mulugeta K. Tefera and Yang Xiaolong. Trust and privacy in mobile participatory sensing: Current trends and future challenges. In *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*, pages 712–716, 2017.
- [13] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. L-diversity: privacy beyond k-anonymity. In *22nd International Conference on Data Engineering (ICDE'06)*, pages 24–24, 2006.
- [14] Kuan Lun Huang, Salil S. Kanhere, and Wen Hu. Preserving privacy in participatory sensing systems. *Comput. Commun.*, 33(11):1266–1280, jul 2010.
- [15] Rim Ben Messaoud, Nouha Sghaier, Mohamed Ali Moussa, and Yacine Ghamri-Doudane. Privacy preserving utility-aware mechanism for data uploading phase in participatory sensing. *IEEE Transactions on Mobile Computing*, PP:1–1, 09 2018.
- [16] Mengmeng Yang, Lingjuan Lyu, Jun Zhao, Tianqing Zhu, and Kwok-Yan Lam. Local differential privacy and its applications: A comprehensive survey, 2020.
- [17] Chong Huang, Peter Kairouz, Xiao Chen, Lalitha Sankar, and Ram Rajagopal. Generative adversarial privacy, 2018.
- [18] Haimonti Dutta, Hillol Kargupta, Souptik Datta, and Krishnamoorthy Sivakumar. Analysis of privacy preserving random perturbation techniques: Further explorations. pages 31–38, 01 2003.
- [19] Fernando Esponda. Negative surveys. *arXiv preprint math/0608176*, 2006.
- [20] Shunsuke Aoki, Masayuki Iwai, and Kaoru Sezaki. Limited negative surveys: Privacy-preserving participatory sensing. In *2012 IEEE 1st International Conference on Cloud Networking (CLOUDNET)*, pages 158–160, 2012.
- [21] Michael Groat, Benjamin Edwards, James Horey, Wenbo He, and Stephanie Forrest. Enhancing privacy in participatory sensing applications with multidimensional data. *2012 IEEE International Conference on Pervasive Computing and Communications, PerCom 2012*, 03 2012.
- [22] Hairuo Xie, Lars Kulik, and Egemen Tanin. Privacy-aware collection of aggregate spatial data. *Data & Knowledge Engineering*, 70(6):576–595, 2011.

- [23] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, pages 614–629, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.
- [24] Krishna Sampigethaya and Radha Poovendran. A survey on mix networks and their secure applications. *Proceedings of the IEEE*, 94(12):2142–2181, 2006.
- [25] Yonglei Yao, Laurence T. Yang, and Neal N. Xiong. Anonymity-based privacy-preserving data reporting for participatory sensing. *IEEE Internet of Things Journal*, 2(5):381–390, 2015.
- [26] Peter Bogetoft, Dan Lund Christensen, Ivan Damgård, Martin Geisler, Thomas Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, Michael Schwartzbach, and Tomas Toft. Secure multiparty computation goes live. In Roger Dingledine and Philippe Golle, editors, *Financial Cryptography and Data Security*, pages 325–343, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.
- [27] Andrew C. Yao. Protocols for secure computations. In *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, pages 160–164, 1982.
- [28] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, nov 1979.
- [29] Maya Mohan, M K Kavitha Devi, and V Jeevan Prakash. Homomorphic encryption-state of the art. In *2017 International Conference on Intelligent Computing and Control (I2C2)*, pages 1–6, 2017.
- [30] Panos Pardalos and Jue Xue. The maximum clique problem. *Journal of Global Optimization*, 4:301–328, 04 1994.
- [31] Y. Qian, F. Ye, and H.H. Chen. *Security in Wireless Communication Networks*. IEEE Press. Wiley, 2021.
- [32] React Native. Accessed: 2022-07-22.
- [33] Core Bluetooth. Accessed: 2022-07-22.
- [34] JavaScriptCore. Accessed: 2022-07-22.
- [35] Hermes – JavaScript engine optimized for React Native. Accessed: 2022-07-22.
- [36] Communication between native and react native. Accessed: 2022-07-22.
- [37] Nodejs. Accessed: 2022-07-22.

- [38] Coen Bron and Joep Kerbosch. Algorithm 457: Finding all cliques of an undirected graph. *Commun. ACM*, 16(9):575–577, sep 1973.