Vulnerability Research of Mobile Applications Commonly Used in Sweden

How Secure Are Mobile Applications Commonly Used in Sweden Against Vulnerabilities?

Yamini Balannagari

Communication Systems
School of Information and Communication Technology
KTH Royal Institute of Technology
Stockholm, Sweden

07 January 2025

Supervisor:
Examiner: Guanciale Roberto

Abstract

This project explores the implementation and evaluation of an efficient and scalable cryptographic solution for Mobile Crowdsensing (MCS) systems, focusing on addressing key challenges related to security, privacy, and computational efficiency. The study begins with a baseline system using the Elliptic Curve Digital Signature Algorithm (ECDSA) for individual message signing and verification. This initial implementation establishes a foundation for understanding the cryptographic workflows and performance metrics in MCS. Building on this, the project introduces an enhanced architecture employing the Boneh-Lynn-Shacham (BLS) aggregated signature scheme, which allows multiple signatures to be combined into a single compact signature, significantly improving scalability and reducing verification overhead.

The analysis highlights the performance improvements achieved by transitioning from individual verification to aggregated verification, particularly in reducing computational costs as the number of participants grows. The results demonstrate that the BLS aggregated scheme provides substantial efficiency gains while maintaining strong security guarantees. The study concludes by proposing future enhancements, including incorporating anonymity-preserving mechanisms to further protect user privacy in MCS applications.

Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

BLS	Boneh-Lynn-Shacham
CA	Certification Authority
DS	Data Server
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
MCS	Mobile Crowdsensing
PKI	Public Key Infrastructure
RA	Registration Authority

Chapter 1

Introduction

Mobile applications have become an indispensable part of modern life, offering convenience and efficiency in various sectors such as banking, transportation, healthcare, and food delivery. In Sweden, a country recognized as one of the most digitally advanced societies in the European Union, the adoption of mobile applications for essential services has grown significantly [1]. However, this widespread usage has also introduced critical security challenges. Mobile applications often handle sensitive personal, financial, and health data, making them prime targets for cyberattacks. Despite advancements in security measures, vulnerabilities such as weak encryption, poor session management, and insufficient authentication mechanisms continue to pose significant risks to users and organizations alike [2].

This research builds upon previous studies, such as the 2022 evaluation of Swedish mobile applications by Ekenblad and Garrido, which identified persistent vulnerabilities like brute-force attacks and session hijacking [2]. By focusing on five widely used applications in Sweden—Foodora, Handelsbanken, SL, Kronans Apotek, and Voi- e scooter hire. This study aims to identify and address security gaps, ultimately contributing to a safer mobile ecosystem.

1.1 Research Problem

1.1.1 Original problem and definition

The increasing reliance on mobile applications for critical services has exposed users to potential security risks. Vulnerabilities in these applications can lead to unauthorized access, data breaches, and financial fraud, compromising user privacy and trust. Despite the implementation of industry-standard security frameworks, such as the OWASP Mobile Security Testing Guide (MSTG), many

applications still exhibit significant security flaws [3]. These vulnerabilities are often a result of inadequate encryption, weak authentication mechanisms, poor session management, and insufficient API security [4].

1.1.2 Scientific and engineering issues

This research addresses key scientific and engineering challenges relevant to mobile application security. These include:

- **Runtime Vulnerability Detection:** Vulnerabilities such as logic flaws and insecure runtime permissions often appear only during execution. Static analysis alone is insufficient to uncover these flaws, making dynamic analysis essential [5].

- **Reverse Engineering of Obfuscated Applications:** Many production applications use obfuscation and anti-debugging techniques to hinder analysis. Analyzing such apps requires tools like Frida, Jadx, and APKTool to bypass protections and inspect internal logic [2].

- **Session and Authentication Security:** Poor session management, token leakage, and flawed authentication flows are common attack vectors. Identifying these issues involves inspecting session handling, login/logout mechanisms, and token storage [5].

- **Cryptography and Key Management:** Insecure encryption algorithms, hardcoded keys, and improper key storage can compromise confidentiality. This requires assessment of cryptographic implementations and API usage [5].

- **API and Backend Communication Security:** Mobile applications often rely on APIs for backend communication. Testing for insecure endpoints, lack of authorization, or weak transport encryption is essential to identify potential breaches [5].

- **Ethical and Legal Considerations:** Testing real-world applications requires strict adherence to ethical guidelines and legal boundaries. The methodology avoids accessing user data or violating application terms, ensuring responsible and compliant research practices.

1.2 Research Question

The core objective of this research is to assess the real-world effectiveness of security measures implemented in popular mobile applications. Accordingly, the central research question is formulated as follows:

How effective are existing mobile application security mitigation strategies in preventing runtime vulnerabilities, and how can dynamic analysis and reverse engineering be systematically applied to evaluate their adequacy and enhance their effectiveness?

This question seeks to explore the effectiveness of advanced techniques like dynamic analysis and reverse engineering in identifying vulnerabilities that may not be detected through static analysis alone. By focusing on runtime vulnerabilities, the study aims to provide a comprehensive understanding of the security gaps in mobile applications, ultimately contributing to the development of more secure systems [6].

1.3 Purpose

The purpose of this thesis is to systematically identify and analyze security vulnerabilities in five widely used Swedish mobile applications: **Foodora**, **Handelsbanken**, **SL (Storstockholms Lokaltrafik)**, **Kronans Apotek**, and **Voi (e-scooter hire)**. The study employs industry-standard methodologies, including the **OWASP Mobile Security Testing Guide (MSTG)**, to assess the adequacy of implemented security measures. Techniques such as static and dynamic analysis, reverse engineering, and penetration testing are used to uncover potential weaknesses in areas such as authentication, data storage, session management, and **API integration**.

1.3.1 Degree Project Purpose

The broader purpose of the degree project is to demonstrate the ability to apply advanced cybersecurity techniques to real-world applications in a systematic, ethical, and legally compliant manner. It also aims to contribute practical insights to the mobile security domain, reinforcing secure development practices and raising awareness of persistent threats in the mobile ecosystem [3].

1.3.2 Benefits

If the goals of this project are achieved, the following stakeholders will benefit:

- **Users**: Enhanced security of mobile applications will protect users sensitive personal, financial, and health data from unauthorized access and cyberattacks. This will increase user trust in digital services, leading to greater adoption of mobile applications for essential services [4].

- **Developers and Organizations**: Insights from this research will help developers and organizations identify and address security vulnerabilities in their applications, reducing the risk of data breaches and financial losses. This encourages ethical practices by prioritizing user privacy and data security [6].

- **Regulatory Bodies**: The findings of this study can inform policymakers and regulatory bodies about the current state of mobile application security, helping them develop more effective regulations and standards. This promotes the development of sustainable security practices that can be applied across the industry [1].

- **Academic Community**: The research contributes to the academic body of knowledge on mobile application security, providing a foundation for future studies and innovations in the field. This encourages further research and collaboration to address emerging security challenges in the digital age [2].

1.3.3 Ethical, Sustainability, and Social Issues

The research involves identifying vulnerabilities in actively used applications, which raises ethical concerns about potential misuse of the findings. To address this, the study will follow responsible disclosure practices, ensuring that vulnerabilities are reported to the respective application developers and organizations before public disclosure [3].

From a sustainability perspective, promoting secure development practices contributes to the sustainability of digital ecosystems by reducing the environmental and economic costs associated with data breaches and cyberattacks [1].

Socially, the research highlights the importance of security in applications that handle sensitive user data, addressing concerns about privacy and data protection in an increasingly digital world. By raising awareness of these issues, the study aims to foster a culture of security-conscious development and usage [4].

1.4 Goals

The primary goals of this research are:

- To systematically identify and analyze security vulnerabilities in widely used Swedish mobile applications using advanced techniques such as static and dynamic analysis, reverse engineering, and penetration testing.
- To uncover runtime vulnerabilities that may not be detectable through static analysis alone, focusing on areas such as authentication, data storage, session management, and API integration.
- To provide actionable insights and recommendations for developers and organizations to address identified vulnerabilities, thereby enhancing the security of mobile applications and protecting sensitive user data.

1.5 Delimitations

The scope of this thesis project is bounded by the following delimitations, which define what is explicitly included and excluded from the study:

- **Application Selection**: The study focuses on five widely used Swedish mobile applications: **Foodora**, **Handelsbanken**, **SL (Storstockholms Lokaltrafik)**, **Kronans Apotek**, and **Voi (e-scooter hire)**.
- **Platform**: The research is limited to Android-based applications.[7].
- **Time Constraints**: Given the limited timeframe for the research, the study primarily focuses on reverse engineering and dynamic analysis techniques to identify and validate vulnerabilities.
- **Mitigation Strategies**: The study focuses on identifying and analyzing vulnerabilities but does not include the development or evaluation of mitigation strategies. Recommendations for addressing vulnerabilities are provided, but their implementation is outside the scope of this project.

These delimitations ensure the study remains focused and manageable within the given constraints.

1.6 Structure of the report

The report is organized into five major chapters: Introduction, Background, Methodology, Analysis, and Conclusions. The Introduction outlines the project's objectives, delimitations, and research questions, providing a clear starting point for understanding the scope and purpose of the study. The Background

chapter covers the necessary technical concepts and includes a literature review, offering context and insight into related work in the field. Following this, the Methodology is divided into two parts: the baseline implementation and the enhanced implement-ation, detailing the approaches used to achieve the project goals. The Analysis section assesses the performance of both implementations, providing a thorough evaluation of the results. Finally, the Conclusions summarize the key findings and implications of the study, closing with reflections on the outcomes and potential future directions.

Chapter 2

Background

This section provides an in-depth exploration of the foundational technical concepts that underpin this project. Understanding these concepts is essential for comprehending the challenges and solutions proposed in the development of secure and efficient Mobile Crowdsensing systems. These topics span the principles of MCS, cryptography, and their intersection to address real-world issues of security, privacy, and scalability. The key concepts include:

- **Mobile Crowdsensing**: An overview of MCS, focusing on its ability to leverage user-contributed data for applications like environmental monitoring, urban planning, and health tracking. This subsection highlights the critical challenges faced by MCS systems, including maintaining security and privacy while ensuring scalability in large-scale deployments.
- **Public Key Infrastructure**: A detailed look at PKI and its role in providing authentication, integrity, and non-repudiation within MCS. This subsection examines how PKI supports secure interactions between users and the system while addressing its limitations in dynamic, high-frequency environments like MCS.
- **Cryptographic concepts**: An explanation of relevant cryptographic topics such as digital signatures, aggregated signatures, batch verification, and elliptic curve cryptography (ECC). These concepts form the backbone of the cryptographic optimizations employed in this project to enhance the efficiency and security of data handling in MCS systems.

2.1 Mobile Crowdsensing (MCS)

Mobile Crowdsensing is a data collection paradigm that leverages the capabilities of smart devices, such as smartphones, to gather and share data for diverse

applications, including environmental monitoring, urban planning, and health tracking. By engaging users as active data contributors, MCS facilitates large-scale, cost-effective sensing that would otherwise require significant resources to achieve. This paradigm not only democratizes data collection but also provides highly granular and real-time insights that drive informed decision-making.

As participation scales, however, MCS systems encounter significant challenges. Ensuring efficiency and scalability becomes increasingly complex, particularly when handling vast amounts of data contributed by a growing number of users. At the same time, safeguarding the privacy of sensitive user information, such as locations and behavioral habits, is paramount to maintaining trust and widespread adoption. Secure and efficient mechanisms for data signing, verification, and transmission are critical to addressing these challenges.

These demands underscore the need for scalable cryptographic solutions that balance robust security, privacy preservation, and operational efficiency. Advanced techniques, such as aggregated signature schemes and batch verification, play a pivotal role in overcoming these obstacles, enabling MCS to achieve its full potential in real-world scenarios.

2.2 Public Key Infrastructure (PKI)

A framework that facilitates secure communication through the use of cryptographic key pairs: a public key (shared with others) and a private key (kept confidential by the owner). Public Key Infrastructure leverages these keys to provide three fundamental security services: authentication, integrity, and non-repudiation.

1. **Authentication**: ensures that the parties involved in communication are who they claim to be, helping prevent impersonation attacks.

2. **Integrity**: guarantees that the data being transmitted has not been altered or tampered with during transmission, providing protection against data corruption or unauthorized modifications.

3. **Non-repudiation**: ensures that the sender of a message cannot deny having sent it, which is crucial for accountability in secure communications.

PKI is widely employed in various applications, including secure email, online banking, and virtual private networks (VPNs). In the context of MCS, PKI plays a significant role in ensuring the security and trustworthiness of data collected from distributed mobile devices. It helps secure the communication channels between users and the central servers, ensuring that the sensed data is authentic,

untampered with, and that participants cannot deny their contributions. This is especially important in MCS applications like environmental monitoring, traffic analysis, and public health data collection, where data integrity and trust are crucial for decision-making and analysis.

This cryptographic system relies on trusted Certificate Authorities (CAs) and Registration Authorities (RAs) to issue digital certificates, validating the ownership of public keys, further strengthening the security infrastructure.

2.3 Cryptographic concepts

Cryptographic techniques are essential for securing communication and ensuring data integrity, especially in environments where sensitive information is transmitted over potentially insecure channels. Among the various cryptographic methods, digital signatures and elliptic-curve cryptography (ECC) play pivotal roles in providing security services such as authentication, integrity, and non-repudiation. These concepts are widely used in applications like secure messaging, online transactions, and Mobile Crowdsensing, where resource constraints and security are both critical concerns.

2.3.1 Digital signatures

Digital signatures are cryptographic mechanisms used to verify the authenticity and integrity of messages. They allow a sender to sign data using their private key, generating a unique signature. This signature can then be verified by the recipient or any third party using the sender's public key. Digital signatures provide three essential security services:

1. **Authentication**: Ensures the identity of the sender, confirming that the message originated from the claimed source.

2. **Integrity**: Guarantees that the message has not been altered during transmission.

3. **Non-repudiation**: Prevents the sender from denying the authenticity of the message, providing proof of origin.

These properties make digital signatures a cornerstone of modern cryptographic systems, securing communications in applications such as secure email, digital contracts, and financial transactions.

2.3.2 Elliptic-curve cryptography

Elliptic-curve cryptography is a modern cryptographic approach that uses the mathematical properties of elliptic curves over finite fields to provide robust security with significantly smaller key sizes than traditional algorithms like RSA. ECC's efficiency makes it ideal for constrained environments such as mobile devices, where processing power, memory, and storage are limited. In MCS, ECC is particularly valuable because it allows for secure and lightweight cryptographic operations, which are critical for maintaining performance in large-scale, distributed sensing applications.

ECC supports various cryptographic operations, including key exchange, encryption, and digital signatures. Its widespread use has made it a preferred choice for modern security protocols, including those used in secure communications, blockchain technologies, and IoT devices.

2.3.2.1 Elliptic Curve Digital Signature Algorithm (ECDSA)

The Elliptic Curve Digital Signature Algorithm (ECDSA) is a widely used cryptographic algorithm that generates and verifies digital signatures based on ECC. It offers strong security with smaller key sizes compared to traditional algorithms, making it ideal for environments with limited computational resources, such as mobile devices and IoT systems. ECDSA is particularly well-suited for Mobile Crowdsensing applications, where devices need to verify the authenticity of data while maintaining low power and processing overhead. Key features of ECDSA include:

- **Authentication**: ECDSA ensures that the signer's identity is verified using their public key, protecting against impersonation attacks.
- **Data integrity**: The algorithm guarantees that the data has not been altered during transmission by generating a unique signature for each message.
- **Efficiency**: ECDSA uses smaller key sizes to achieve the same level of security as larger key-based algorithms, resulting in faster computations and lower storage requirements, which is essential for mobile and IoT devices.

Figure 2.1 illustrates the key steps of ECDSA, from the initial message hashing to the process of signature generation, transmission, and verification. This visual representation demonstrates how ECDSA ensures secure communication while minimizing computational load.

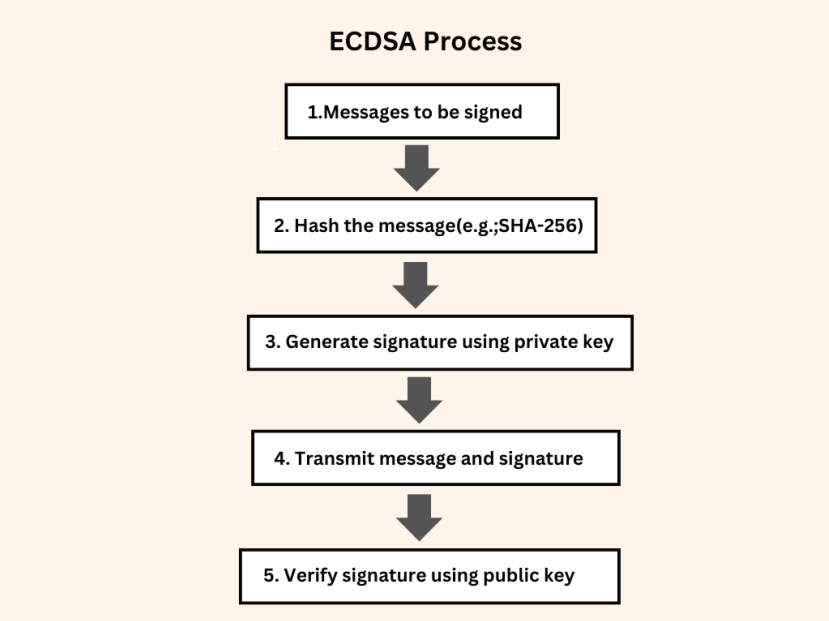

Figure 2.1: Overview of the ECDSA Process

2.4 Advanced cryptographic techniques

As cryptographic systems continue to evolve, the demand for more efficient and scalable techniques has become increasingly critical, particularly in environments characterized by large volumes of data and numerous users. Techniques such as aggregated signatures and batch verification have emerged as pivotal advancements, offering substantial improvements in scalability and efficiency. These techniques are indispensable for applications like Mobile Crowdsensing, where resource constraints, dynamic user participation, and high data throughput pose significant challenges.

Aggregated signatures and batch verification optimize cryptographic operations by minimizing computational overhead, reducing communication costs, and maintaining robust security properties. Their application ensures that even resource-constrained environments can achieve the necessary levels of authentication, integrity, and privacy.

In this section, we provide a detailed explanation of both aggregated signatures and batch verification. These concepts are crucial for understanding the cryptographic framework developed in this project, as they form the foundation of the proposed solutions to enhance scalability, efficiency, and security.

2.4.1 Aggregated signatures

Aggregated signatures are a powerful cryptographic technique that combines multiple individual signatures into a single compact signature. This process not only reduces the size of transmitted data but also minimizes the computational cost of signature verification. Instead of verifying each individual signature separately, the server only needs to perform one verification operation for the aggregated signature, improving both efficiency and scalability. First introduced by Boneh et al. in their seminal work on aggregate and verifiably encrypted signatures, this technique has become a cornerstone of efficient cryptographic protocols in systems with high signer volumes, such as MCS, where numerous devices need to authenticate their data [**?**].

The aggregation process involves several steps:

- **Signers and their keys:** Each signer has a private key (s_i) and a corresponding public key ($pk_i = s_i \cdot Q$), where Q is a base point in the elliptic curve group G_2. The signer creates their signature (sig_i) by hashing the message ($H = hashToG1(msg)$) and then multiplying it by their private key: $sig_i = s_i \cdot H$.

- **Message hashing:** The message is hashed into an element of a group (G_1) using a function such as $hashToG1(msg)$. This step ensures that the message becomes a fixed-size input for cryptographic operations, preventing the possibility of collision attacks.

- **Signature aggregation:** All individual signatures ($sig_1, sig_2, \ldots, sig_N$) are combined into a single aggregated signature ($aggSig = \Sigma sig_i$). Similarly, the public keys ($pk_1, pk_2, \ldots, pk_N$) are aggregated into a combined public key ($aggPK = \Sigma pk_i$).

- **Verification:** The verifier uses bilinear pairing to verify the aggregated signature. Instead of verifying each individual signature, the verifier computes:

  $$e(aggSig, Q) \stackrel{?}{=} e(H, aggPK)$$

  If this equation holds true, it confirms the validity of the aggregated signature for the given message and the combined public keys. This method ensures both correctness and integrity without the need for individual verifications.

To conclude, this technique significantly reduces computational and communication over-head by consolidating multiple signatures into one compact signature. In systems like MCS, where there are large numbers of signers, aggregated

signatures improve scalability and efficiency, enabling secure data collection with minimal resource usage.

2.4.2 Batch verification

Batch verification is an advanced cryptographic technique that enables the simultaneous verification of multiple digital signatures in a single operation. Rather than verifying each signature individually, this approach processes all signatures together, significantly reducing computational overhead. This optimization is particularly advantageous in scenarios with a high volume of signatures, such as Mobile Crowdsensing, where large numbers of signatures must be verified efficiently to maintain system performance.

The concept of batch verification was first introduced by Fiat in the context of digital signatures, demonstrating how to verify multiple RSA signatures concurrently without compromising security [**?**]. By leveraging algebraic properties and advanced cryptographic primitives, batch verification combines individual verification checks into a single streamlined operation. This drastically enhances the verification process, making it feasible to handle large-scale mobile sensing applications with limited resources.

Figure 2.2, sourced from Klaytn's explanation of BLS signatures [**?**], provides a clear summary of the batch verification process, illustrating how individual verifications are aggregated into a single computation step to achieve efficiency and scalability. This visualization emphasizes the practicality of batch verification in reducing computational cost while maintaining the integrity of the verification process.

The reduced computational cost and improved efficiency make batch verification an essential tool for applications requiring scalable and resource-efficient crypto-graphic mechanisms, such as processing data from numerous participants in MCS systems.

2.5 Boneh-Lynn-Shacham (BLS) Signatures

As cryptographic demands evolve, innovative techniques like the Boneh-Lynn-Shacham (BLS) signature scheme have emerged to address challenges of scalability, efficiency, and security in modern applications. BLS signatures, based on bilinear pairings over elliptic curves, offer a compact and flexible framework for digital authentication, particularly in environments requiring lightweight and scalable operations. This section introduces the key aspects of BLS signatures, the cryptographic curve BLS12383, and the underlying concept of bilinear pairing,

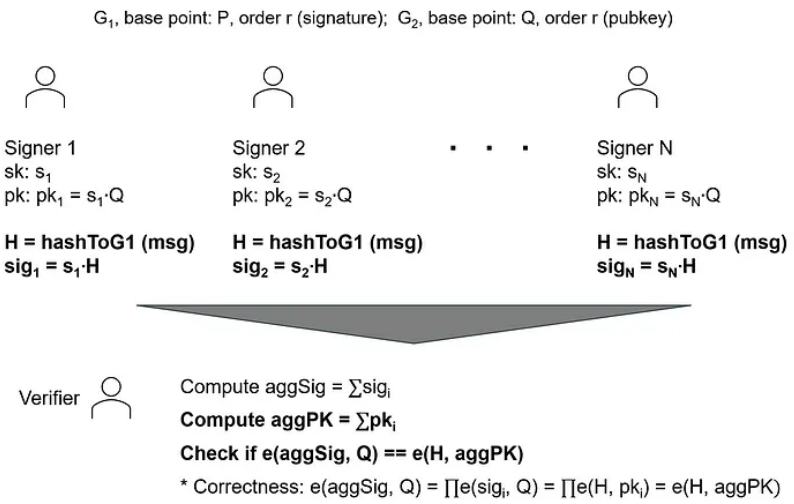

Figure 2.2: Schematic diagram of the aggregate signature scheme [**?**]

.

all of which are instrumental in ensuring efficient and secure data verification in Mobile Crowdsensing and similar domains.

2.5.1 Introduction to BLS signatures

BLS signatures are a cryptographic scheme leveraging bilinear pairings on elliptic curves to produce compact, non-interactive, and highly efficient digital signatures. Their unique properties have positioned them as a powerful solution in applications demanding low storage and transmission overhead, such as MCS.

The key characteristics of BLS signatures include:

- **Compactness**: The signature size in the BLS scheme is significantly smaller compared to traditional cryptographic methods. This efficiency minimizes storage and bandwidth requirements, making BLS signatures ideal for large-scale systems.

- **Aggregation**: Multiple BLS signatures can be combined into a single aggregated signature, greatly reducing the computational effort required for verification. This is especially beneficial in MCS, where signatures from numerous participants must be authenticated.

- **Non-Interactivity**: BLS signatures allow individual entities to generate their signatures independently, without requiring any coordination with

others. This simplifies distributed signing processes and enhances usability in decentralized systems.

2.5.2 BLS12383 Curve

The BLS12383 curve is an elliptic curve specifically designed for pairing-based cryptographic operations. As part of the Barreto-Lynn-Scott (BLS) family of curves, it plays a pivotal role in the efficiency and security of the BLS signature scheme.

This curve is notable for the following features:

- **Pairing efficiency**: The BLS12383 curve is optimized for fast and efficient bilinear pairing operations, which are essential for verifying aggregated signatures and supporting other cryptographic constructs.
- **Security level**: Offering 128-bit security, the curve provides strong resistance against modern cryptographic attacks, ensuring that its use in large-scale systems remains robust.
- **Application in aggregate signatures**: The curve's properties make it well-suited for scenarios requiring compact and efficient signature aggregation, a critical requirement for privacy-preserving systems like MCS.

The inclusion of the BLS12383 curve in this project facilitates secure and efficient cryptographic operations such as signing and verification. Its adoption ensures the scalability and reliability of the system while maintaining the integrity of data submitted by users in large-scale crowdsensing environments.

2.5.3 Bilinear pairing

Bilinear pairing is the mathematical operation that underpins pairing-based cryptographic schemes like BLS signatures. It enables advanced functionalities by mapping elements from two elliptic curve groups into a third target group while preserving specific linear properties.

The mathematical structure of bilinear pairing allows for the following cryptographic benefits:

- **Support for aggregated verification**: Bilinear pairings enable simultaneous verification of aggregated signatures, reducing computational load.
- **Advanced cryptographic constructs**: Pairings are integral to schemes like identity-based encryption, attribute-based encryption, and group signatures, expanding the versatility of cryptographic systems.

- **Enhanced security mechanisms**: The unique mathematical properties of bilinear pairing ensure that operations like signature verification remain secure while providing efficiency gains.

In this project, bilinear pairing serves as the cornerstone of cryptographic operations, facilitating efficient and scalable authentication processes. Its application ensures that the system can manage the high demands of data verification inherent in large-scale MCS environments while maintaining the highest standards of security and performance.

2.6 Related work

The use of aggregated signature schemes in Mobile Crowdsensing has garnered significant attention due to their potential to address challenges related to scalability, efficiency, and security. Traditional centralized authentication mechanisms, while robust in providing authentication, integrity, and non-repudiation, suffer from scalability issues and lack inherent privacy-preserving features, particularly in dynamic environments like MCS as highlighted by Khodaei et al.[**?**]. To address these limitations, researchers have explored various cryptographic techniques, including certificateless and aggregate signature schemes.

Several studies have demonstrated the advantages of aggregate signature schemes in enhancing efficiency and scalability [**?**] [**?**] [**?**] [**?**]. For instance, Liu et al. propose a lightweight authentication mechanism combining blockchain technology with sequential aggregate signatures, leveraging BLS short signatures to improve key length and verification efficiency in large-scale MCS scenarios [**?**]. Similarly, Wang et al. present an approach for efficient and anonymous batch verification of large-scale concurrent data in Mobile Healthcare Crowd Sensing (MHCS) using an improved certificateless aggregate signature, ensuring both data integrity and user privacy while addressing high data volumes [**?**].

Kim et al. extend these advancements by introducing an aggregate one-time signature (Agg-OTS) scheme, which leverages elliptic curve cryptography to enable efficient message aggregation and signature verification. Their approach minimizes the computational overhead associated with large-scale data aggregation and verification, making it highly suitable for dynamic environments like MCS [**?**].

Privacy-preserving and scalable authentication mechanisms are essential in resource-constrained environments like MCS. For example, Sai et al. introduce a certificateless double authentication preventing aggregate signature, which ensures anonymous sensing data authentication and scalability, mitigating fraudulent activities and supporting batch authentication efficiently [**?**]. Building on this,

Rajkumar et al. propose a certificateless signature aggregation scheme employing elliptic curve cryptography, specifically designed for environments like vehicular ad hoc networks (VANETs). This scheme reduces computational overhead and communication costs while preserving privacy, offering insights applicable to MCS scenarios facing similar challenges [**?**].

Gisdakis et al. emphasize the importance of balancing security and privacy in MCS through cryptographic solutions that integrate accountability with privacy-preserving mechanisms. Their work underscores the role of advanced signature schemes in constructing holistic security architectures [**?**]. Additionally, Zhang et al. propose a blind signature scheme based on the SM2 algorithm, demonstrating its relevance for lightweight and secure authentication protocols in MCS, despite its original focus on financial and transactional systems [**?**].

Other noteworthy contributions include a distributed signing protocol for mobile devices by Khodaei et al., which employs identity-based cryptography to address scalability and security issues in resource-constrained environments [**?**]. The protocol's practical implementation highlights its potential for MCS applications. Similarly, Kim et al. introduce a randomized hybrid pseudonym scheme that ensures user anonymity and unlinkability in vehicular systems, offering adaptable methodologies for dynamic and privacy-sensitive MCS environments [**?**].

These studies collectively illustrate the potential of aggregated and certificateless signature schemes in addressing the dual challenges of scalability and privacy in MCS. By leveraging advanced cryptographic techniques, these methodologies establish a robust framework for efficient, secure, and privacy-preserving MCS deployments, paving the way for future innovations in large-scale data collection and transmission.

Chapter 3

Method

The implementation of this project is divided into two major stages, each contributing to the overall functionality and efficiency of the final system. This section explains each stage, focusing on the system's goals, tools used, and results achieved.

In the first stage, we implemented a system for an MCS scenario using ECDSA, where messages were individually signed by clients and verified by the server one at a time. In the second stage, we extended the system to use the BLS aggregated signature scheme, enabling multiple signatures to be combined and verified in a single operation, significantly improving efficiency.

3.1 Tools

During the implementation of this project, several tools and programming languages were utilized to ensure efficient development and functionality. Table 3.1 summarizes the tools and their respective purposes in the project.

Table 3.1: Tools and technologies used in the implementation

Tool/Technology	**Component**	**Purpose**
Android Studio	Client-side Application	Integrated Development Environment (IDE) for building the Android client using Java. Handles network operations and cryptographic tasks.
MIRACL Library	Client, RA, DS	Provides cryptographic functionalities such as key pair generation, encryption, digital signature creation, and verification.
MongoDB	RA, DS	Primary database to store client public keys (RA) and data records (DS).
Retrofit2	Client, Servers	HTTP client library used for seamless communication between the client and servers.
GitHub	Project Repository	Version control system for collaborative development and maintaining project history.

3.2 Baseline system implementation: ECDSA with Individual signature verification

In this section, we describe the baseline implementation of the system, which served as the foundational framework for the project. This initial implementation utilized ECDSA for message signing and verification, where each signature was generated and verified individually. The primary goal of this stage was to establish a functional system to serve as a starting point for further enhancements.

The baseline system involved developing the client application in Android Studio, and two server entities, the RA and the Data Server, which acted as the core platform for authentication, secure data collection, message signing and verification. During this phase, we focused on understanding the interaction

flows between the client and server, integrating cryptographic operations using the MIRACL library, and familiarizing ourselves with its features. This stage also helped identify challenges associated with implementing and managing cryptographic operations in MCS scenarios.

In addition to laying the groundwork specially for the client base, this implementation provided a reference point to later demonstrate the advantages of employing an aggregated signature scheme. By building upon this system, we were able to transition to a more advanced design, incorporating BLS aggregated signatures to improve efficiency and scalability.

Figure 3.1 illustrates the workflow of the baseline system, which proceeds as follows:

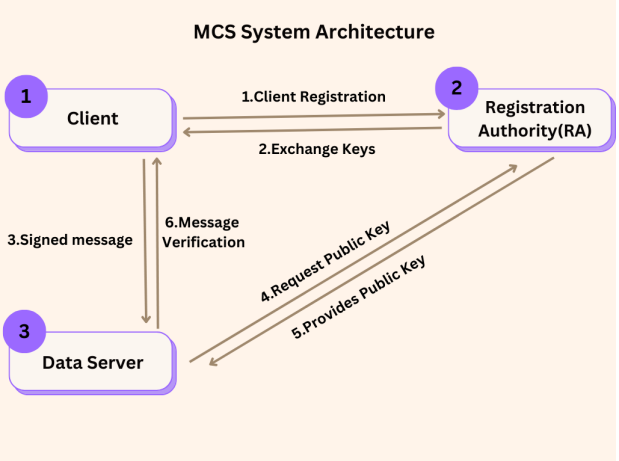

Figure 3.1: Baseline system architecture using ECDSA for an MCS use case

1. **Client registration**: The client begins by registering with the Registration Authority, by sending its username to the RA.

2. **Key exchange**: The RA generates the key pair for the client, and provides them to the client. While saving the Public key in its own data base as well.

3. **Message signing:** After registration, the client generates a message, signs it using its private key via the ECDSA algorithm, and sends the signed message and the signatures to the data server.

4. **Public key request**: Upon receiving the signed message, the data server contacts the RA to request the corresponding public key for the sender.

5. **Public key provision:** The RA responds by providing the requested public key to the data server.

6. **Message verification:** The data server verifies the authenticity and integrity of the message using the public key and the signature. If the signature is valid, the server confirms the message is from the registered client and processes it accordingly.

Table 3.2 provides a detailed explanation of the implemented system, highlighting the algorithmic workflow and the underlying cryptographic operations based on the Elliptic Curve Digital Signature Algorithm. The figure outlines key processes, including the generation of private and public keys, signature creation, and signature verification. The equations described in the figure are derived from the ECDSA specification, as detailed in the SEC 1 standard [**?**] and NIST FIPS 186-4 [**?**].

Table 3.2: Algorithmic flow and cryptographic details for ECDSA-based system

Step	**Description and Cryptographic Details**
Input	Message M, Client's Private Key sk, Public Key Database, System Parameters.
Output	Verification Result (Valid/Invalid).
1. Client-Side: Message Signing	The client generates a signature for the message M using the ECDSA as follows: • Hash the message M using a secure hash algorithm (e.g., SHA-256) [**?**]. • Compute the signature (r,s) using the private key sk, the hash of M, and a random nonce k: $$r = (k \cdot G)_x \mod q, \quad s = k^{-1} \cdot (H(M) + r \cdot sk) \mod q$$ These equations are part of the ECDSA signature generation process, as defined in the SEC 1 standard [**?**] and NIST FIPS 186-4 [**?**]. The client sends the message M, the signature (r,s), and its username to the Data Server (DS) via HTTPS.
2. Data Server: Message Reception	The DS receives the signed message and stores it temporarily. It prepares for verification by retrieving the corresponding public key for the user [**?**].
3. Public Key Retrieval	The DS requests the public key pk of the client (identified by the username) from the Registration Authority (RA). The RA retrieves and sends the public key pk from its database to the DS [**?**].
4. Signature Verification	The DS verifies the signature (r,s) using the ECDSA verification process: • Compute the elliptic curve points: $$u_1 = H(M) \cdot s^{-1} \mod q, \quad u_2 = r \cdot s^{-1} \mod q$$ $$V = u_1 \cdot G + u_2 \cdot pk$$ These equations are part of the ECDSA signature verification process, as described in SEC 1 [**?**] and NIST FIPS 186-4 [**?**]. • Check if the x-coordinate of V matches r: $$V_x \mod q \stackrel{?}{=} r$$ If the condition holds, the signature is valid.
Output	The system responds with a success message if the signature is valid. Otherwise, an error message indicating invalidity is returned.

3.3 Enhanced system implementation

The enhanced system builds upon the baseline architecture by incorporating the BLS aggregated signature scheme to address the scalability and efficiency challenges of Mobile Crowdsensing. This implementation introduces key improvements, including a shift in key generation to the client-side, the use of aggregated signatures, and batch verification on the server. The interaction flow between the client, Registration Authority, and Data Server (DS) is illustrated in Figure 3.2 and proceeds as follows:

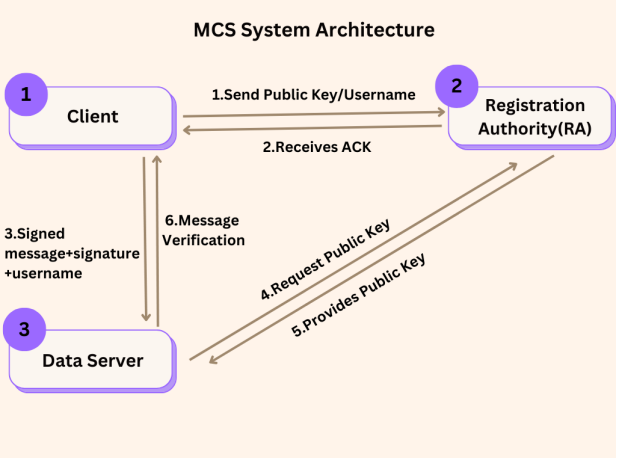

Figure 3.2: Enhanced system architecture using BLS aggregated scheme for an MCS use case

1. **Client registration**: The client starts by registering with the RA. During this process, the client generates its private-public key pair locally to ensure the private key is never exposed or transmitted. The client then sends its public key along with its username to the RA.

2. **Key exchange and acknowledgment**: The RA securely stores the client's public key and associates it with the respective username. An acknowledgment is sent to the client to confirm successful registration.

3. **Message signing and transmission**: The client signs its data using the private key and the BLS signature scheme. It then transmits the signed message, the signature, and the username to the Data Server.

4. **Public key retrieval**: Upon receiving signed messages, the Data Server temporarily stores them in a buffer. The system includes a configurable parameter to define the number of buffered messages required to trigger

aggregation. Once the buffer reaches this threshold, the Data Server requests the public keys for the clients associated with these messages from the RA.

5. **Public key provision**: The RA responds by providing the corresponding public keys for the requested clients, enabling the server to proceed with verification.

6. **Batch verification**: The Data server aggregates the signatures from the buffered messages into a single compact signature. It then performs batch verification using the aggregated signature and the retrieved public keys, significantly reducing computational overhead compared to individual verification.

Table 3.3 provides a detailed explanation of the implemented aggregated signature scheme algorithm. The algorithm is based on the Boneh-Lynn-Shacham signature scheme, which leverages bilinear pairings for efficient and compact digital signatures. The equations mentioned in the table are derived from the foundational work presented by Linn et al. in "Short signatures from the weil pairing" [**?**].

Table 3.3: Algorithmic flow and cryptographic details of the proposed system

Step	**Description and Cryptographic Details**
Input	Message M, Client's Private Key sk, Public Key Database, System Parameters (e.g., Aggregation Count).
Output	Verification Result (Valid/Invalid).
1. Client-Side: Message Signing	The client generates a private-public key pair locally using the BLS (Boneh-Lynn-Shacham) signature scheme [**?**]: • Private Key (sk): A random scalar $s \in Z_q^*$. • Public Key (pk): Computed as $pk = s \cdot G$, where G is the generator of the elliptic curve group G_2. To sign the message M: • Hash the message M onto the elliptic curve group G_1: $H_M = h(M)$, where h is a secure hash function. • Compute the signature as $\sigma = s \cdot H_M$. These steps are derived from the BLS signature scheme, which provides short and efficient digital signatures using bilinear pairings [**?**]. The signed message, signature σ, and username are sent to the Data Server (DS) via HTTPS.
2. Data Server: Message Reception	The Data Server buffers received messages $\{M_1, M_2, \ldots, M_n\}$ and their corresponding signatures $\{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ until the buffer reaches the predefined aggregation count n.
3. Public Key Retrieval	The Data Server requests public keys $\{pk_1, pk_2, \ldots, pk_n\}$ for the corresponding usernames from the RA. The RA retrieves and sends the requested public keys from its database to the DS.
4. Signature Aggregation	The Data Server aggregates the signatures $\{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ into a single compact signature σ_{agg}: $$\sigma_{agg} = \sum_{i=1}^{n} \sigma_i$$ This aggregation step, unique to the BLS signature scheme, leverages the additive property of elliptic curve points [**?**, **?**]. Aggregation ensures that multiple signatures are combined into a single, verifiable signature, minimizing storage and computation overhead.
5. Batch Verification	The aggregated signature σ_{agg} is verified using bilinear pairings: $$e(\sigma_{agg}, G) \stackrel{?}{=} \prod_{i=1}^{n} e(H_{M_i}, pk_i)$$ • e is the bilinear pairing function that maps $G_1 \times G_2 \to G_T$. • The left-hand side (LHS) computes the pairing of σ_{agg} with G. • The right-hand side (RHS) computes the product of pairings for each hashed message H_{M_i} and its corresponding public key pk_i. This batch verification process, a feature of BLS signatures, ensures computational efficiency by verifying multiple messages in a single operation [**?**].
Output	The system returns a success message if verification is valid; otherwise, it returns a failure message.

3.4 Comparison of baseline and enhanced architectures

There are significant differences between the baseline and enhanced architectures, which are essential to highlight for a clear understanding of the systems, especially when evaluating their performance and scalability. Table 3.4 summarizes these differences concisely.

Table 3.4: Comparison of baseline and enhanced architectures

Aspect	Baseline Architecture	Enhanced Architecture
Programming Language for RA and DS	Python (MIRACL integrated in Python)	Java (MIRACL integrated in Java)
Signature Scheme	Individual signing and verification using ECDSA	Aggregated BLS with batch verification
Key Generation	Performed in the RA and shared with the client	Performed on the client side for improved security
Server-Side Operations	Verification of individual signatures	Aggregation and batch verification of signatures
Transition to Java for Servers	Not applicable	Avoided compatibility issues in cryptographic operations between client and servers using MIRACL
Rationale for Key Generation on Client Side	Private keys were generated in the RA and sent to the client, which posed security risks	Private keys are generated directly on the client side to prevent exposure or sharing

3.4.1 Challenges of the implementation

The implementation of this project presented several challenges, stemming primarily from the complexity of integrating cryptographic libraries, managing cross-platform compatibility, and developing a robust client-server architecture. Below are the key challenges faced during the implementation:

- **MIRACL library**: The MIRACL library was a central component for implementing cryptographic functionalities, such as key generation, message signing, and verification. However, a significant challenge was the lack of comprehensive documentation and resources for using the library.

- **Compatibility issues between Python and Java version of MIRACL**: Initially, in the baseline implementation the server was developed in Python, but compatibility issues arose when attempting to verify messages in the enhanced implementation. Specifically, the BLS classes differed between the Python and Java implementations of the MIRACL library, making it impossible to verify aggregated signatures across platforms. To resolve this, the server implementation was transitioned to Java, ensuring compatibility with the client application and enabling seamless integration of the BLS aggregated signature scheme.

- **Android Studio Client Development**: Developing the client in Android Studio introduced its own set of challenges. Updating layouts and debugging cryptographic library implementations within the Android environment proved to be particularly time-consuming. Android Studio's build process and dependency management, combined with the inherent complexities of mobile app development, added to the difficulty of implementing and testing cryptographic operations on the client side.

Chapter 4

Analysis

This chapter evaluates the performance and scalability of the implemented system by analyzing key metrics collected during both the baseline and enhanced implementations. These metrics are crucial for understanding the trade-offs and benefits of transitioning from an individual signature verification system to an aggregated signature scheme.

4.1 Methodology and setup

The experiments were conducted using the system specifications outlined in Table 4.1 for the server, and for the client-side, an emulated device was set up in Android Studio with the specification of Table 4.2. Metrics were collected on a per-client basis for the client-side operations (message signing) and on the server side (signature aggregation and verification). Each experiment was repeated multiple times to ensure reliability, and the average times were calculated to minimize the impact of outliers.

System Component	Details
Processor	12th Gen Intel i7-12700H
Memory	16 GB LPDDR5 RAM
Operating System	Microsoft Windows 11 Home

Table 4.1: System Specifications for the Experiments

Device Specification	Details
Device	Google Pixel 6a
API Level	TiramisuPrivacySandbox
RAM	2 GB
CPU Cores	4

Table 4.2: Emulated Smartphone Specifications

4.2 Baseline system analysis: ECDSA with individual verification

At the time of implementing the baseline system, the aggregated signature scheme was not yet available. Consequently, an ECDSA-based system was analyzed to evaluate its performance and identify areas for improvement. The baseline implementation focused on two different elliptic curves, BLS12383 and BN254, within the ECDSA framework. Key performance metrics, including signing and verification times, were measured to compare the efficiency of these curves and establish a foundation for future enhancements. The analysis was conducted using file-based transmission, wherein data was securely transmitted to the DS.

The decision to evaluate and compare different cryptographic curves and signature schemes was driven by their distinct properties. BLS12383 and BN254 offer different trade-offs in terms of security and computational efficiency.

By understanding these trade-offs, valuable insights can be gained into the balance between robustness and performance. To ensure findings were applicable to diverse MCS deployments, the methodology involved simulating various conditions, including variations in message size and client numbers.

Metrics were calculated as averages over 200 runs for each message size to ensure reliability and statistical significance. The results were stored in a database for further analysis, with ECDSA signatures guaranteeing the integrity and authenticity of transmitted messages. Notably, file-based transmission was used in this implementation instead of real data sensed by devices, focusing the evaluation on comparing the performance of the two elliptic curves.

This comparative analysis highlights the importance of selecting appropriate cryptographic primitives based on system requirements. The findings lay the groundwork for subsequent exploration of advanced schemes, such as aggregated signatures, which are better suited for large-scale and resource-constrained environments typical of MCS systems.

4.2.1 Signature time analysis

Figure 4.1 visualizes the signature generation times for both curves across different message sizes.

- **Curve 1 (BLS12383)**
  - A noticeable initial spike in signature times is observed for smaller message sizes (up to 3,000 bytes), likely caused by the system's warm-up effect, where operations such as cache initialization introduce delays.
  - After stabilizing beyond 4,000 bytes, the average signature time hovers around 15 ms.
- **Curve 2 (BN254)**:
  - Similarly, the BN254 curve experiences an initial peak but stabilizes faster, maintaining an average signature time of approximately 10 ms, making it more efficient than BLS12383.

Results indicated that both BLS12383 and BN254 curves showed stable signature times across message sizes. However, BN254 consistently outperformed BLS12383. BN254's superior performance is attributed to its smaller field size (254 bits vs. 383 bits), resulting in lower computational complexity during elliptic curve operations. Although BLS12383 provides stronger security (128-bit vs. 100-bit), this comes at the cost of increased computation, especially

during point multiplication and hashing operations. To mitigate the warm-up effect in future experiments, a pre-warming approach was adopted, wherein a few sample signatures are generated to initialize resources and caches before collecting performance metrics.

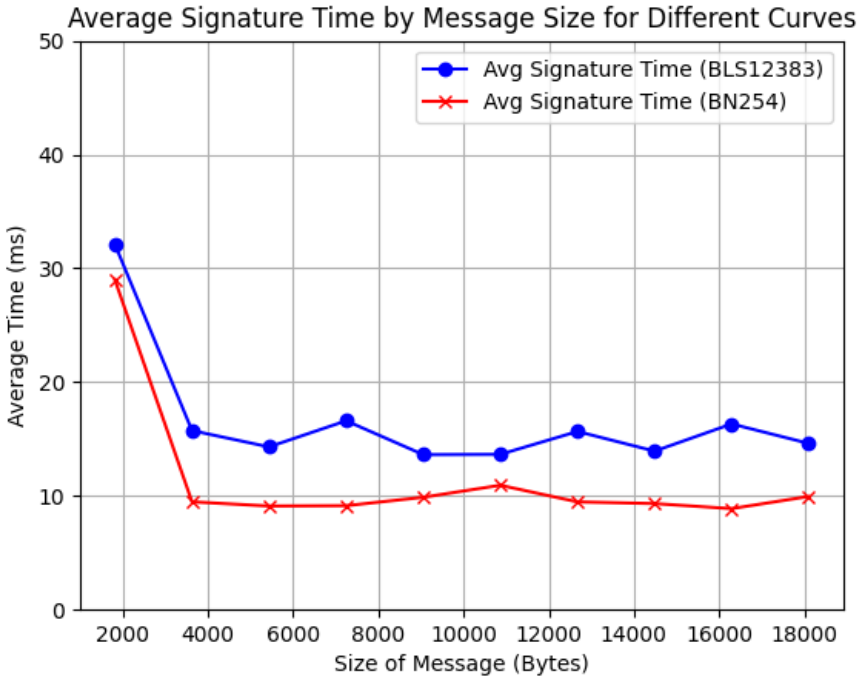

Figure 4.1: Average signature time by message size for different curves

4.2.2 Verification time analysis

Figure 4.2 illustrates the verification times for both curves. Verification times exhibited similar trends to signature times, with BN254 achieving more efficient verification than BLS12383. The pairing operations, which dominate verification, scale with the field size of the elliptic curve. The smaller BN254 field size translates to faster pairing computations. Additionally, resource contention when multiple processes compete for limited computing resources such as CPU, memory, or I/O bandwidth may have amplified delays during BLS12383 testing, given its higher computational demands.

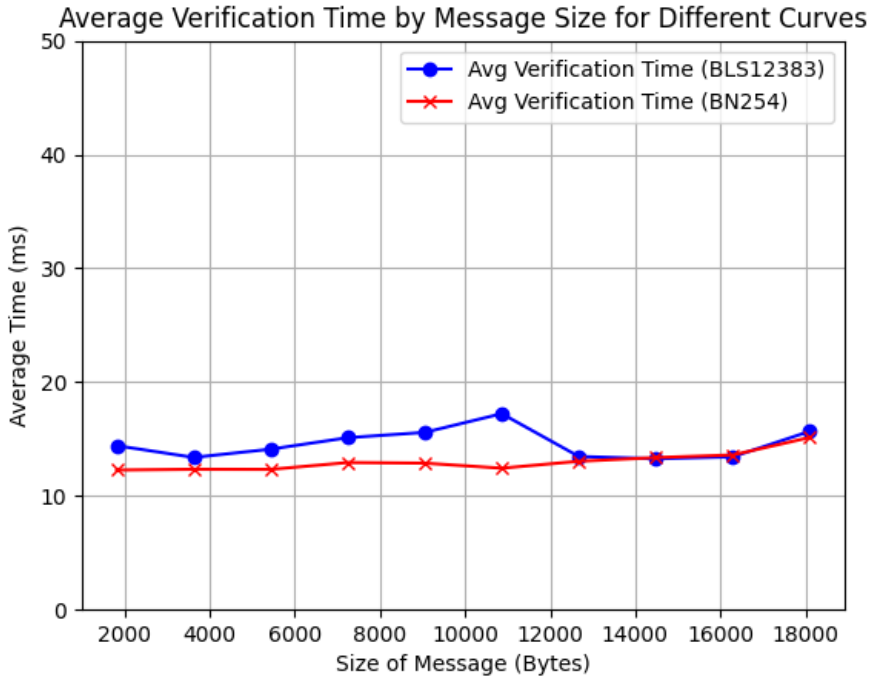

Figure 4.2: Average verification time by message size for different curves

- **Curve 1 (BLS12383)**
  - Verification times remain stable, averaging between 14-15 ms. A minor spike to 18 ms is observed at a message size of 11,000 bytes, caused by resource contention while running the client and server on the same device.
- **Curve 2 (BN254)**:
  - The BN254 curve exhibits greater consistency, with verification times averaging 12 ms across all message sizes, demonstrating its efficiency over BLS12383.

Figure 4.3 shows the average verification time for varying message sizes using the BLS12383 curve, where recourse constraints have been addressed by running the client and server on separate devices. The results show that verification times remain consistent across all message sizes, averaging around 8-9 ms.

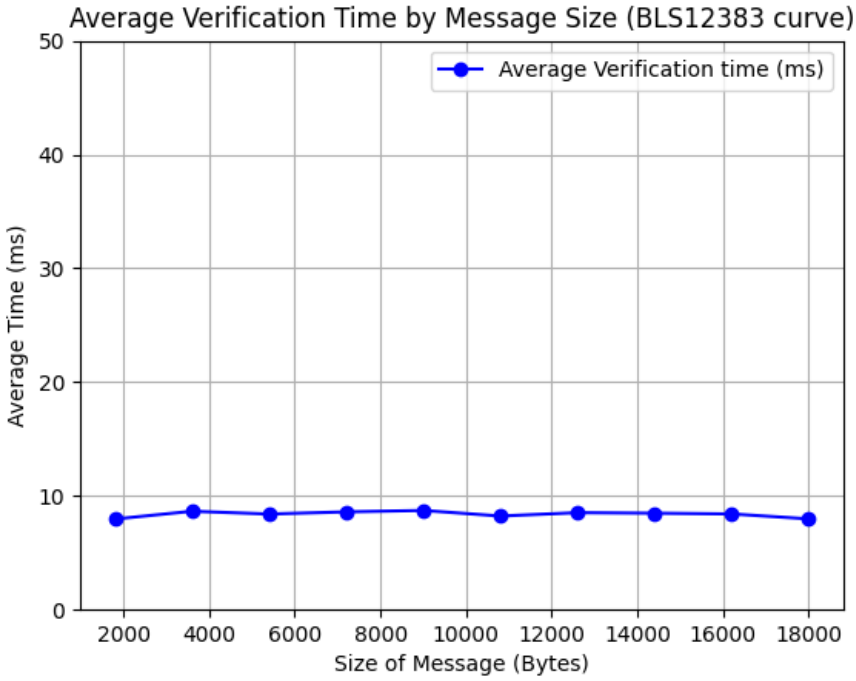

Figure 4.3: Average verification time by message size for BLS12383

4.3 Enhanced system analysis: BLS aggregated signature scheme

Building on the insights from the baseline system, the enhanced implementation incorporated the BLS aggregated signature scheme with batch verification. This shift addressed the scalability limitations observed in the baseline system, particularly the linear growth in verification time with an increasing number of clients.

4.3.1 Verification time efficiency

Figure 4.4 compares aggregated verification time against single-user verification multiplied by the number of clients. This highlights the efficiency gains offered by the aggregated signature scheme.

- **Aggregated verification time**:
  - The aggregated verification time grows almost linearly as the number of clients increases. This is expected because the aggregated signature

verification is performed in a batch, significantly reducing the computational overhead compared to verifying individual signatures.

  - For 200 clients, the aggregated verification time is substantially lower than the single-user time multiplied by the number of clients, underscoring the scalability of the scheme.

- **Single-user verification time**: In this approach, as the number of clients increases, the time required to verify all signatures individually scales linearly with the number of clients, resulting in high overhead for large-scale systems.

The significant reduction in verification time for the aggregated scheme is due to the single pairing computation required for the aggregated signature. In contrast, single-user verification requires one pairing computation per client, leading to a proportional increase in computational cost with the number of clients. This efficiency makes the aggregated signature scheme particularly suitable for Mobile Crowdsensing, where large numbers of clients may transmit data simultaneously.

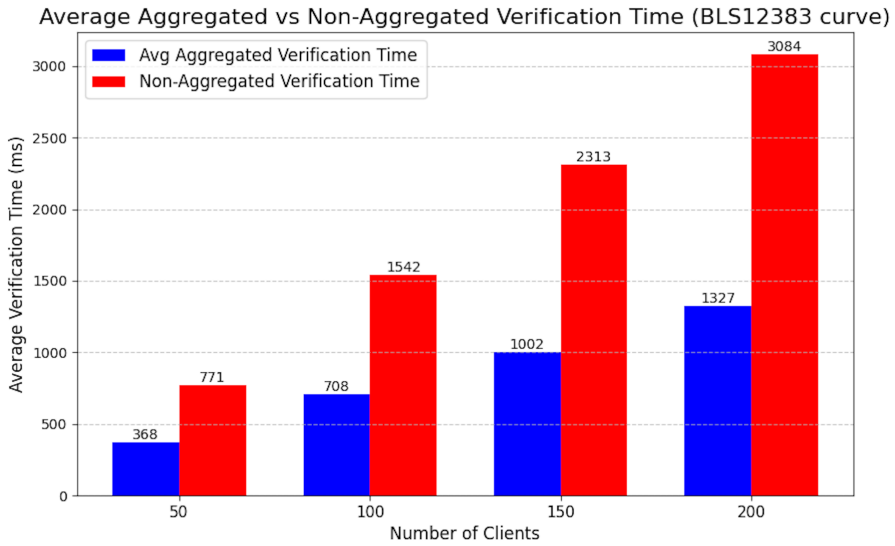

Figure 4.4: Aggregated vs. Non-Aggregated verification time across clients

4.3.2 Comparative analysis of BLS and ECDSA across message sizes

Figure 4.5 and 4.6 compares the signing and verification times for ECDSA and BLS schemes across varying message sizes.

- **ECDSA**:
  - **Signing Time**: remains consistent across all message sizes, averaging approximately 12-14 ms.
  - **Verification time:** also consistent, averaging between 7-8 ms.

- **BLS**:
  - **Signing time**: slightly lower than ECDSA, averaging around 10-12 ms across all message sizes.
  - **Verification time:** slightly higher than signing time, averaging 12-14 ms, with minor variations across message sizes.

BLS and ECDSA showed stable performance across varying message sizes, with BLS slightly outperforming ECDSA in signing times but incurring higher verification times. Signing in BLS involves a straightforward multiplication of a private key with a hashed message point. This operation is computationally efficient and benefits from the optimized pairing-friendly curves. Verification in BLS requires bilinear pairing, a more complex operation than the elliptic curve arithmetic used in ECDSA verification. The additional cost of pairings explains the marginally higher verification times for BLS.

The relatively stable signing and verification times for both schemes across all message sizes indicate that the computational cost of cryptographic operations in these schemes is not strongly influenced by message size. This is because the hashing step maps the message to a fixed-size element in the elliptic curve group, making subsequent operations independent of the original message size. The marginally better performance of BLS in signing times can be attributed to its more optimized design for such operations. However, its slightly higher verification times compared to ECDSA arise from the additional pairing operations required in the BLS scheme.

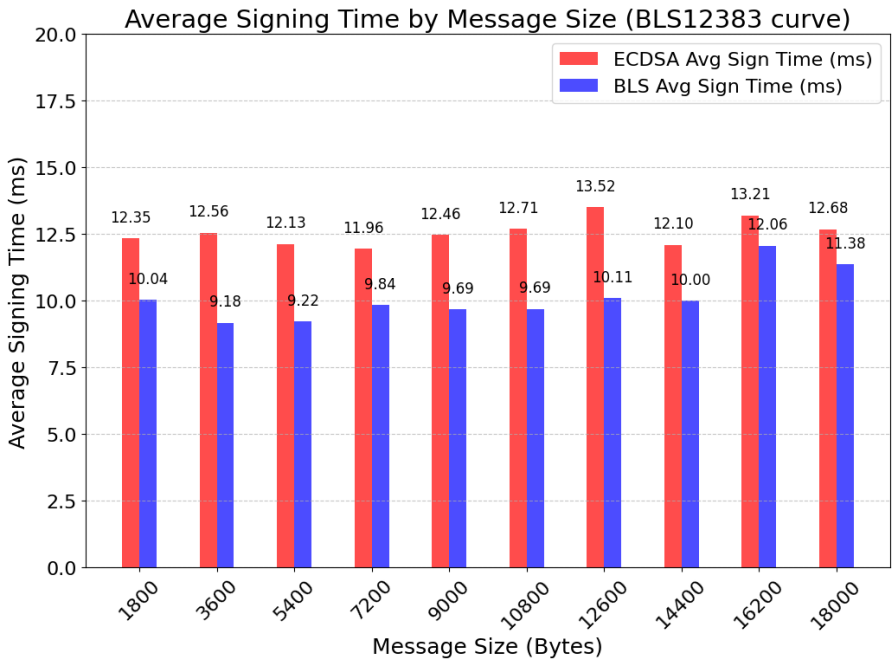

Figure 4.5: Comparison of ECDSA and BLS signature time by Message Size

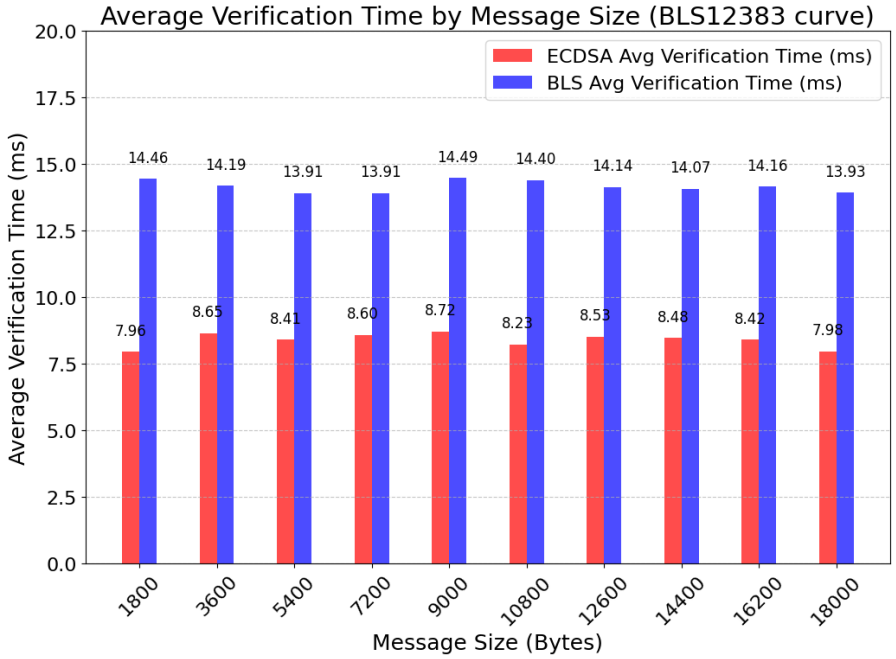

Figure 4.6: Comparison of ECDSA and BLS verification time by Message Size

As observed in Figure 4.5, the difference in signing time between ECDSA and BLS for the last three message sizes (14400, 16200, and 18000 bytes) was not as consistent as it was for the other message sizes. Specifically, the margin between the two schemes was noticeably smaller for these larger message sizes. To investigate this anomaly further, we conducted additional experiments under stricter and more controlled conditions. In this refined setup, the RA and DS were placed on separate physical machines, and the client device was replaced with a real Android device instead of an emulator. This change aimed to eliminate potential sources of measurement contamination associated with emulator overheads, which can introduce unpredict-able delays and impact the accuracy of timing measurements. Furthermore, we ensured no other processes or activities were running on any of the involved devices to minimize interference.

In addition to these adjustments, we increased the number of iterations from 200 to 400 to achieve a more robust and reliable average. The results of this new experiment are shown in Figure 4.7. As evident from the updated figure, the margin between ECDSA and BLS for the previously anomalous message sizes is now consistent with the margins observed for other message sizes. This confirms our hypothesis that the initial discrepancies were caused by measurement pollution in the earlier setup.These findings demonstrate that under controlled and realistic conditions,such as using separate servers and real hardware instead of emulators,the message signing time margin between ECDSA and BLS remains consistent, even for larger message sizes. This consistency further strengthens the reliability of our analysis and the observed behavior of the two schemes.

4.4 Comparison of BLS12383 and BN254

Figure 4.8 brings us back to performance comparison of the two elliptic curves, BLS12383 and BN254, specifically looking at the verification time of aggregated signatures. The comparison between BLS12383 and BN254 revealed significant differences in performance, with BN254 consistently achieving lower verification times. The larger field size of BLS12383 increases computational overhead for point multiplication and pairing operations. This difference highlights the trade-off between security and efficiency.

- **Curve 1 (BLS12383)**
  - The BLS12383 curve shows a linear growth in verification time as the number of clients increases, as observed in Figure 4.4. BLS is suitable for applications requiring higher cryptographic strength, such as critical infrastructure or sensitive data handling.

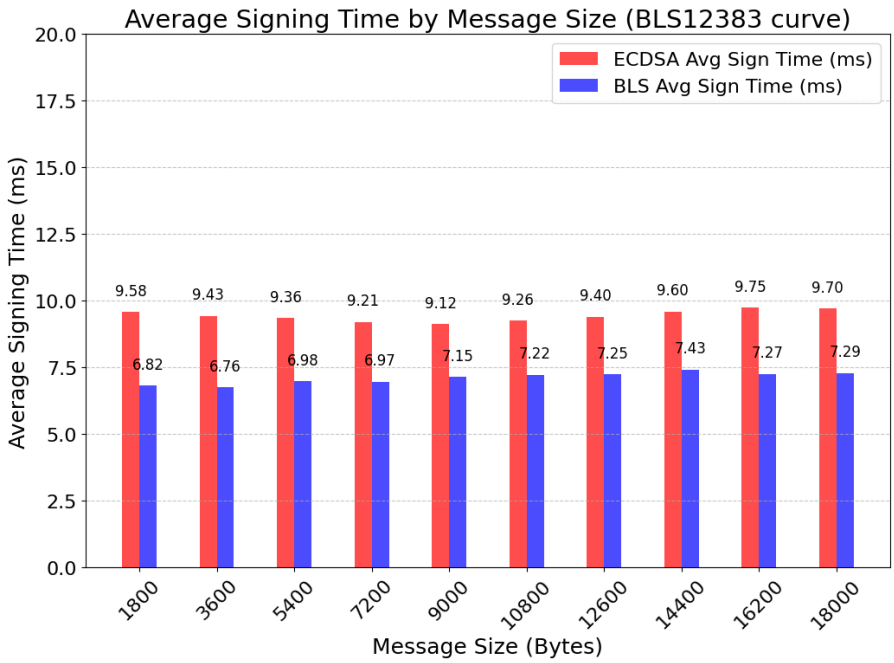

Figure 4.7: Comparison of ECDSA and BLS signature time by Message Size for 400 iterations

- **Curve 2 (BN254)**:

  - The BN254 curve follows a similar growth pattern to the BLS12383 curve as the number of clients increases. However, the verification time for each client count is roughly half that of the BLS12383 curve. This curve is optimal for less sensitive use cases where computational efficiency is paramount, such as large-scale MCS with non-critical data.

The observed difference in verification times between these two elliptic curves stems primarily from their field sizes, as indicated by the names of the curves. BLS12383 operates over a 383-bit field, compared to the 254 bits of the BN254 curve. This means that each operation in the verification process using BLS12383 involves larger numbers, resulting in higher overhead and slower verification times. However, the trade-off for using BN254 is its lower security. It provides only 100 bits of security, compared to the 128 bits offered by the BLS12383 curve.

Figure 4.9: Caption

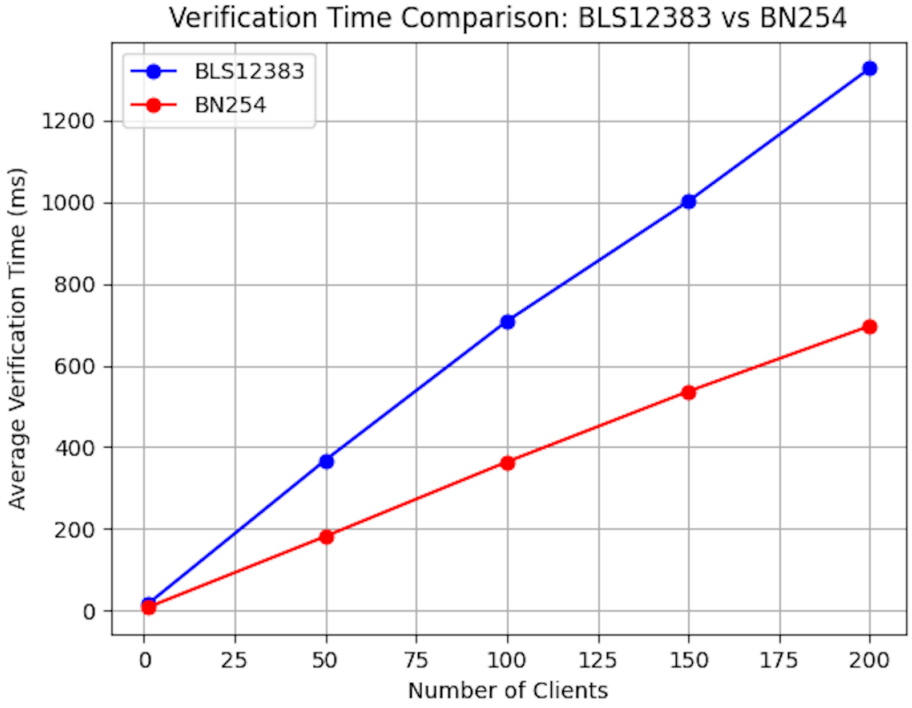

Figure 4.8: Comparison of aggregated verification time for BLS12383 and BN254

4.5 Breakdown of verification time

Table 4.3 shows a detailed breakdown of the cryptographic operations used in the batch verification process. The times shown in milliseconds is the averages calculated from 100 independent aggregated verifications using the BLS12383 curve. Showing the times for 2, 50, 100, and 200 signature aggregations and the ”per signature time”, which represents the average time spent on each cryptographic operation for a single signature within the aggregation.

Operation	2 Signatures		50 Signatures		100 Signatures		200 Signatures	
	Total time	**Per signature**	**Total time**	**Per signature**	**Total time**	**Per signature**	**Total time**	**Per signature**
Hash to point	5.61	2.81	128.94	2.58	375.00	3.75	496.94	2.48
Decode PK	0.86	0.43	23.01	0.46	67.53	0.68	87.46	0.44
Member Check	3.42	1.71	82.86	1.66	247.21	2.47	326.30	1.63
Pairing (PK, H(msg))	3.83	1.92	154.12	3.08	460.62	4.61	606.38	3.03
Aggregate Pairings	5.70	x	5.68	x	8.29	x	5.68	x
Final Exponent	5.64	x	5.63	x	8.41	x	5.78	x
Total	25.09	x	400.38	x	1167.91	x	1529.10	x

Table 4.3: Verification time broken down into cryptographic operations

In Table 4.3, the verification time for various cryptographic operations is broken down for different batch sizes (2, 50, 100, and 200 signatures). However, the case of 150 signatures is missing from the table. To ensure consistency with the metrics presented earlier, we decided to include and plot the data for 150 signatures as well.

The plot provided in figure 4.10 visualizes the average times for these operations when 150 clients are involved. As shown, the metrics are distributed across different cryptographic steps such as aggregation time, public key request time, total hashing time, decoding public keys, member checks, pairings, and final exponentiation. As observed, the verification time for 150 signatures falls between that of 100 and 200 signatures, affirming the consistency of the trend: as the number of signatures increases, the verification time also increases proportionally.

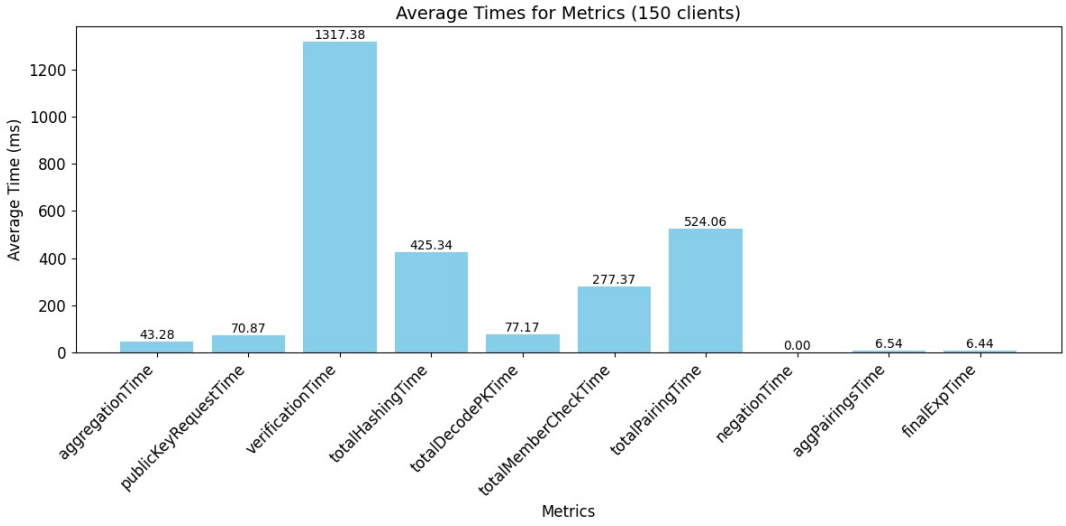

Figure 4.10: Verification time broken down into cryptographic operations for 150 signatures

Chapter 5

Conclusions

This chapter explains the conclusions obtained throughout the design, development and evaluation described in this thesis and proposes a number of improvements, extensions, or complements that may be of interest in order to improve and extend the work presented.

5.1 Conclusion

The development and evaluation of this Mobile Crowdsensing system highlighted significant progress in achieving secure, efficient, and scalable data management through cryptographic innovations. Starting with a baseline implementation that utilized the Elliptic Curve Digital Signature Algorithm , the project progressed to an enhanced architecture that incorporated Boneh-Lynn-Shacham aggregated signature schemes. This transition not only addressed the system's scalability challenges but also demonstrated the practicality of leveraging advanced cryptographic solutions in real-world scenarios.

The comparative analysis revealed that the BLS aggregated signature scheme offered substantial efficiency gains, particularly in signature verification. While the baseline system required individual verification for each client's message, the aggregated approach consolidated multiple signatures into a single verification step. This improvement reduced computational overhead and demonstrated near-linear scalability, making the system more suitable for large-scale MCS applications. Furthermore, across varying message sizes, both ECDSA and BLS schemes showed stable performance due to the inherent properties of hashing in elliptic curve cryptography. However, the BLS scheme proved slightly more efficient in signing operations, although it incurred marginally higher verification times due to pairing computations. These results validated the hypothesis that aggregated signature schemes enhance system performance

without compromising security.

Several challenges emerged during the implementation process. The integration of the MIRACL cryptographic library proved difficult due to limited documentation, especially for Java implementations, requiring extensive trial and error. Additionally, compatibility issues between the Python and Java versions of the library created obstacles, particularly in verifying messages signed in one environment within another. To address this, the server-side implementation was restructured entirely in Java, ensuring consistency with the Android-based client. Another significant challenge was the development and debugging of the Android client in Android Studio, which required substantial effort to resolve cryptographic integration issues while maintaining functional user interfaces.

The enhanced system demonstrated clear advantages in scalability and resource efficiency. It successfully showcased the potential of aggregated signature schemes to reduce computational and communication overhead while maintaining robust security guarantees. The findings also underscored the importance of systematic pre-initialization, as observed during the warm-up effect in the baseline system, to achieve accurate and reliable performance metrics.

In conclusion, this project demonstrated that advanced cryptographic techniques, particularly aggregated signature schemes, can effectively address the security and scalability challenges inherent in MCS systems.

5.2 Future work

Building upon the aggregated signature scheme implemented in this project, a compelling direction for future work would involve enhancing the privacy of users by incorporating mechanisms to ensure anonymity. A potential approach could involve using pseudonyms or anonymous credential systems, allowing users to participate in Mobile Crowdsensing tasks without revealing their true identities. This would add a layer of privacy protection, making the system more resilient to misuse or privacy breaches, which are critical in scenarios involving sensitive data such as location or behavioral patterns.

In addition, integrating advanced cryptographic techniques, such as group signatures or ring signatures, could further strengthen the system's anonymity guarantees. These techniques would allow data submissions to be unlinkable to individual users while still maintaining accountability. These enhancements would make the system suitable for broader applications, such as healthcare or urban sensing, where anonymity and privacy are of paramount importance.

5.3 Sustainable development

This project supports sustainable development by introducing efficiency improvements in Mobile Crowdsensing systems, particularly through the use of aggregated signature schemes. Traditional cryptographic methods, such as individual signing and verification, require significant computational resources and energy, especially as the number of participants increases. In contrast, the aggregated signature scheme allows multiple signatures to be combined into a single compact signature, enabling batch verification and drastically reducing computational overhead. This improvement directly translates to lower energy consumption on both client devices and servers, making the system more sustainable for large-scale deployments.

For example, consider a scenario in environmental monitoring where hundreds of devices collect and transmit data about air quality or noise pollution in a city. Without aggregated signatures, the server would need to process each signature individually, consuming more computational power and energy. Aggregated signatures streamline this process, reducing the number of cryptographic operations and saving energy. This efficiency is particularly important for battery-powered devices like smartphones or IoT sensors, extending their operational lifespan and reducing the frequency of recharging or battery replacement.

Moreover, the reduced computational demand at the server side means fewer resources are needed for server infrastructure, which can lower overall energy consumption and carbon footprint in data centers. This makes the system more suitable for real-world applications where energy efficiency and scalability are critical, such as in smart cities, healthcare monitoring, or disaster response.

By implementing such resource-efficient cryptographic techniques, the project not only enhances the scalability and performance of MCS systems but also contributes to reducing electronic waste, optimizing energy use, and promoting a more environmentally responsible approach to technology deployment. This ensures that the solutions are aligned with the principles of sustainability while addressing the growing demand for secure and privacy-preserving data collection systems.

5.4 Ethical considerations

Ethical considerations are central to this project, particularly in ensuring the privacy of users participating in Mobile Crowdsensing systems. MCS inherently relies on user-contributed data, which often includes sensitive informa-tion such as location, behavioral patterns, or health metrics. Protecting this data is essential to maintaining user trust and preventing misuse or exploitation of personal

information.

The system addresses these concerns by implementing privacy-preserving cryptographic mechanisms, such as aggregated signature schemes. These schemes ensure that user data can be authenticated and verified without exposing individual identities or sensitive information.

The project also aligns with principles of transparency and accountability by implementing robust cryptographic practices that users can rely on. It ensures that data collection and processing are secure, and access is limited to authorized parties, minimizing the risks of data breaches or unauthorized use. Overall, the emphasis on privacy in MCS systems highlights the ethical responsibility to protect users while enabling impactful applications like urban planning, environmental monitoring, and public health initiatives.

Bibliography

[1] E. Commission, "Digital economy and society index (desi) 2021," 2021, accessed: 6 May 2022. [Online]. Available: https://digital-strategy.ec.europa.eu/en/policies/desi

[2] J. Ekenblad and S. Garrido, "Security evaluation of ten swedish mobile applications," 2022.

[3] OWASP, "Owasp mobile security testing guide (mstg)," 2022, accessed: 4 May 2022. [Online]. Available: https://mobile-security.gitbook.io/mobile-security-testing-guide/

[4] P. Stirparo, "Mobileak: Security and privacy of personal data in mobile applications," 2015.

[5] OWASP Foundation, "Mobile security testing guide (mstg)," https://mobile-security.gitbook.io/mobile-security-testing-guide/, 2022, accessed: 2025-03-23.

[6] R. Nilsson, "Penetration testing of android applications," 2020.

[7] G. Stats, "Mobile operating system market share worldwide," 2022, accessed: 4 May 2022. [Online]. Available: https://gs.statcounter.com/os-market-share/mobile/worldwide