TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Villem Susi 242744IVCM

Secure Location Sharing on Android Devices Using Zero Trust Principles

Master's Thesis

Supervisor: Matthew James Sorell
PhD

Tallinn 2026

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Villem Susi 242744IVCM

Turvaline asukoha jagamine Android-seadmetes nullusalduse põhimõtetel

Magistritöö

Juhendaja: Matthew James Sorell
PhD

Tallinn 2026

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Villem Susi

20.05.2026

Acknowledgements

The author would like to express gratitude to the thesis supervisor, Matthew James Sorell, for his guidance, constructive feedback, and support throughout the development of this thesis. The author would also like to thank the academic staff of Tallinn University of Technology for providing the knowledge, resources, and learning environment that supported the development of this thesis. A special thank you goes to Jaan for his eternal patience in acting as a sounding board for the author's many half-formed ideas.

The preparation of this thesis involved the use of generative artificial intelligence (AI) tools in accordance with the CEUR-WS guidelines on the use of generative AI (https://ceur-ws.org/GenAI/Policy.html). Specifically, the author used ChatGPT (OpenAI, GPT-5.5) to assist with language refinement to improve grammar, clarity, wording, and readability of text written by the author. The author takes full responsibility for the accuracy, integrity and originality of the content.

Abstract

Secure Location Sharing on Android Devices Using Zero Trust Principles

The widespread adoption of mobile device location-sharing applications has introduced significant privacy and security challenges regarding the continuous collection, transmission and processing of sensitive data. Existing solutions often rely on perimeter-based security assumptions, which are unsuitable for environments where mobile devices often operate outside of controlled networks and are exposed to untrusted factors. Unauthorised access to location data can enable both digital and physical threats, including behavioural profiling, surveillance, and attacks.

The thesis analyses the security limitations of current solutions and proposes a Zero Trust architecture-based design for securing location data flows. The primary outcome is a software library that can be integrated into existing mobile applications to support the implementation of selected Zero Trust principles. The approach combines decision logic on endpoint devices and repeated trust evaluation with least-privilege and context-aware access control policies to reduce implicit trust relationships, particularly in scenarios where devices operate outside controlled networks.

The solution is evaluated through structured analysis and prototype-based testing, suggesting prototype-level technical feasibility and architectural plausibility for improving disclosure control in mobile location-sharing systems. By aligning mobile location sharing with Zero Trust principles, this thesis contributes to the development of more resilient and trustworthy mobile applications.

The thesis is in English and contains 52 pages of text, 7 chapters, 2 figures, and 9 tables.

Keywords:

Zero Trust Architecture, location sharing, mobile device security

CERCS: P175 Informatics, systems theory

Lühikokkuvõte

Turvaline asukoha jagamine Android-seadmetes nullusalduse (*zero trust*) põhimõtetel

Laialdane mobiilseadmete asukoha jagamise rakenduste kasutuselevõtt on tekitanud märkimisväärseid privaatsus- ja turvaprobleeme, seoses tundlike asukohaandmete pideva kogumise, edastamise ja töötlemisega. Olemasolevad lahendused tuginevad sageli perimeetripõhistele turvalisuse eeldustele, mis ei sobi keskkondadesse, kus mobiilseadmed asuvad väljaspool kontrollitud võrke ning on kättesaadavad mitteusaldusväärsetele osapooltele. Volitamata juurdepääs asukohaandmetele võib põhjustada nii digitaalseid kui ka füüsilisi ohte, sealhulgas käitumuslikku profileerimist, jälgimist ja rünnakuid.

Käesolev lõputöö analüüsib olemasolevate lahenduste puudujääke turvalisuses ning pakub välja nullusalduse arhitektuuril põhineva lahenduse asukoha andmete turvaliseks edastamiseks. Töö peamine tulemus on tarkvarateek, mida saab integreerida olemasolevatesse mobiilseadmete rakendustesse, et toetada nullusalduse põhimõtete rakendamist.

Lähenemine seob otsustusloogika lõppseadmetel ning korduva usalduskontrolli, kasutades vähima õiguse ning kontekstiteadlikkuse poliise, et vähendada vaikimisi usaldatavaid ühendusi, eriti olukordades kus seadmed töötavad ebausaldusväärsetes võrkudes.

Lahendust hinnatakse struktureeritud analüüsi ja prototüübi-põhise testimise abil, osutades tehnilisele teostatavusele ja arhitektuuri põhjendatusele, parandamaks andmete jagamise kontrolli mobiilseadmete asukohajagamiste süsteemides. Ühildades asukoha jagamise nullusalduse põhimõtetega, panustab käesolev töö vastupidavamate ja usaldusväärsemate mobiilirakenduste arendamisesse.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 52 leheküljel, 7 peatükki, 2 joonist ja 9 tabelit.

Võtmesõnad:

Nullusalduse arhitektuur, asukoha jagamine, mobiilseadme turvalisus

CERCS: P175 Informaatika, süsteemiteooria

List of abbreviations and terms

API	Application Programming Interface
BYOD	Bring Your Own Device
DoS	Denial-of-Service
HTTPS	Hypertext Transfer Protocol Secure
MFA	Multi-Factor Authentication
MitM	Man in the Middle
PDP	Policy Decision Point
PEP	Policy Enforcement Point
TLS	Transport Layer Security
UI	User Interface
ZTA	Zero Trust Architecture

Table of contents

1. Introduction

Location sharing on mobile devices has been widely popularised by major mobile device vendors. The widespread use of mobile device applications for personal location sharing that continuously collect and share users' locations has raised significant privacy and security risks. Unlike many other categories of personal data, location information can reveal where a person is, their routines, relationships, habits and other sensitive behavioural patterns. Unauthorised access to such data may therefore enable both digital and physical harms, including profiling, surveillance, and targeted attacks.

Mobile devices often operate outside trusted network boundaries and frequently connect through untrusted networks, which makes their operating environment difficult to control. As a result, traditional perimeter-based security models have become increasingly obsolete and inadequate for such use cases. This thesis analyses how Zero Trust principles can be implemented in mobile device location-sharing systems and presents the design of a supporting software library intended for integration into existing applications. The thesis contributes a concrete Android library architecture in which trust evaluation and disclosure control are enforced on endpoint devices rather than centralised backend services. This solution aims to reduce centralised trust assumptions and support more privacy-preserving disclosure of location data.

While Zero Trust Architecture has been extensively explored in enterprise and cloud environments, its application to consumer mobile endpoints remains comparatively underdeveloped. Existing research mainly focuses on individual security controls within mobile applications rather than on an integrated Zero Trust-based architecture for location sharing. This thesis addresses the following research question: How can Android-based location sharing be redesigned using Zero Trust principles to reduce reliance on centralised trust and support adaptive disclosure of location data? This thesis addresses the question by proposing a Zero Trust-based architecture for Android location sharing in which trust evaluation and policy enforcement are moved to endpoint devices, while the central server is reduced to a relay role. The design further links trust conditions to multiple disclosure levels, allowing location sharing to be adapted to the current trust state.

Chapter 2 introduces the technical background of mobile location data, mobile operating system security, and Zero Trust Architecture. Chapter 3 reviews the state of the art and identifies the research gap addressed by this work. Chapter 4 presents the contribution of the thesis, including the design requirements, trust model, architectural decisions, and software library structure. Chapter 5 evaluates the design through security analysis and prototype-based testing. Finally, Chapter 6 discusses the results, practical deployment implications, broader impacts, and directions for future work. Chapter 7 concludes the thesis by answering the research question and summarising the main findings.

2. Background

This chapter gives an overview of important location-sharing aspects in mobile devices. After this, the focus shifts to the standard security model in current popular mobile device operating systems. Lastly, the chapter opens the different aspects of Zero Trust Architecture.

2.1. Mobile device location collection

Location data can be categorised into a few distinct types based on its source, precision, and interpretability [1]. These types include GPS, network-based and semantic location data. The most common form is GPS data, which provides high-precision geographic coordinates from satellites expressed as latitude and longitude values. GPS-based data is typically accurate up to 10 meters under optimal conditions, but may degrade in indoor or obstructed environments [2]. Further, mobile devices also rely on network-based location data, derived from Wi-Fi access points and cellular tower triangulation. Although generally less precise than GPS, network-based positioning offers improved availability in indoor settings [3]. Alternatively to raw coordinates, location data can also be represented semantically, where geographic characteristics are translated to human-readable contexts such as “home”, “work”, or “favourite coffee shop”. Although semantic representations improve usability, they also expose behavioural patterns that are difficult to anonymise once aggregated over time [4].

2.1.1. Location data granularity and sensitivity

All of these forms of location data vary in granularity and usability. The granularity of location data refers to the level of spatial and temporal precision at which a device’s position is recorded or shared. Highly granular data could include real-time, exact geographic coordinates augmented by details like height above sea level or the floor number of a building. Lower-granularity data may represent approximate areas, such as a city district or a radius-based region (e.g. “within 1 kilometre”). GPS data from a smartphone can be highly variable, ranging from 3-meter precision with additional height data to having no satellite connection in remote or obstructed areas [5]. Network-based location data can only be collected when a suitable network node, such as a Wi-Fi access point or cellular tower, is available. Even then, location precision is generally

limited to the coverage area of the relevant node, with the exception of cellular triangulation in densely populated areas with many towers. In that case, it is still improbable to gain any further detail about the location like height data. GPS and network-based location data can be used together (Assisted GPS) to enhance the result [3]. Semantic location data is the most versatile in its usage possibilities. It can be as vague as only specifying the country or as specific as GPS coordinates. The main limitation is that the interpretability of semantic labels depends on the recipient's prior knowledge. For example, "home" is only understandable to those who know where that is for the person in question [5].

Hyper-precise location data therefore represents a trade-off where marginal usability gains often come at disproportionate privacy costs. Having the exact coordinates leak instead of a semantic keyword raises increased personal risk. Meanwhile, having the semantic keyword leak might not expose the exact location of the device, but instead exposes personal details about the behaviour of the victim. From a usability perspective, combining Assisted GPS with semantic labels may be advantageous, so both the exact location and nature of that location would be visible. However, if both of those pieces of data were to leak concurrently, the privacy loss implications would be direct (i.e. the exact location of the victim's home leaks).

2.1.2. Importance of time

Temporal granularity is as important as location granularity. Continuous real-time updates of location data expose more behavioural information than occasional or event-based requests. Higher location granularity improves observability and usability, but this improvement comes at a direct privacy cost. Continuous sharing enables precise tracking and behavioural inference, even when individual data points appear harmless in isolation. Even delayed, but still constant location sharing can become highly sensitive when aggregated over time, as many data points enable reconstruction of routines and personally meaningful places.

2.2. Mobile device operating system security model

Modern mobile operating systems provide a security model intended to protect both system resources and user data from unauthorised access. This model relies on several built-in mechanisms, including application isolation, permission controls, and protected storage and

communication channels. Understanding these mechanisms is important for this thesis because the proposed architecture builds on them while also addressing the trust and access-control limitations that remain at the application level.

2.2.1. Application sandboxing

Application sandboxing is a fundamental security mechanism in modern mobile operating systems that isolates applications from one another and from critical system resources. This model requires applications to run within a restricted environment with limited access to files, memory, and hardware interfaces [6]. This isolation prevents one application from directly accessing the data or functionality of another without explicit authorisation. On Android platforms, sandboxing is enforced by assigning each application a unique user identifier (UID) and applying process isolation at the kernel level, allowing system calls via a protected API [6]. While sandboxing reduces inter-application attacks, it does not prevent abuse of legitimately granted privileges [7]. Applications can still access sensitive resources like location services if granted the appropriate permissions by the user.

2.2.2. Permission systems

Mobile operating systems rely on permission systems to regulate application access to sensitive resources such as location data, camera input, storage, and network interfaces. In Android systems, applications must declare required permissions in a manifest file, and users must grant or deny these permissions at a granular level either during installation or during runtime [7]. However, a noticeable amount of users do not, for various reasons, pay enough attention to permissions, which indicates that Android permission alerts do not help users make correct security decisions [8].

2.2.3. Secure data storage and transfer

Securely storing and transferring data is essential to protect the integrity and confidentiality of that data. Modern mobile operating systems and development frameworks strongly encourage or require the use of encrypted communication protocols such as TLS to ensure confidentiality and integrity during data exchange [9]. TLS protects against eavesdropping, message tampering, and man-in-the-middle (MITM) attacks by establishing authenticated and encrypted channels

between endpoints. Android provides built-in libraries and network security configurations to enforce certificate validation, and secure transport by default [10]. Furthermore, mobile device operating systems also offer mechanisms for protecting stored data. Applications can store sensitive information within sandboxed internal storage that is inaccessible to other unauthorised applications and network calls [10]. Moreover, Android provides a hardware-backed key management system (Android Keystore) that enables secure generation, storage, and use of cryptographic keys [10].

2.2.4. Limitations of OS-level security

Although modern mobile operating systems provide strong built-in security mechanisms such as sandboxing, permission enforcement, and encrypted communication channels, these controls are not secure enough on their own [11]. Each application should implement its own additional separation controls for inter-application communication and data storage on top of the pre-existing OS level mechanisms [11]. This is because OS level protections primarily focus on isolating applications from system resources, but they do not evaluate the trustworthiness of users, devices or contextual factors. Once a user grants permissions to an application to access location data, the operating system does not continuously reassess whether that access remains necessary. As a result, OS-level mechanisms alone are insufficient for enforcing continuous trust reassessment once permissions are granted, leaving applications responsible for compensating controls [11].

2.3. Zero Trust Architecture (ZTA)

Zero Trust Architecture (ZTA) is a security concept to eliminate implicit trust and enforce continuous authentication, authorisation, and validation regardless of location or network. Instead of relying on implicit trust, every access request must be verified [12]. The concept relies on several principles, like “never trust, always verify”, least-privilege identity access policies, and strict continuous authentication. Users, devices, and applications need to be authenticated before each resource call, and trust decisions are based on multiple contextual factors like time of day and geolocation of the request [13]. In contrast to perimeter-based models, Zero Trust explicitly assumes that internal components may already be compromised and treats every

interaction as potentially hostile. This reduces implicit trust relationships and limits the potential impact of compromised devices or credentials [13].

2.3.1. NIST ZTA reference model

According to the NIST ZTA framework, the enforcement of Zero Trust policies relies on a set of interacting components responsible for evaluating and controlling access to protected resources (see Fig. 1) [12]. At the centre of this model is Policy Decision Point (PDP), which determines whether an access request should be allowed based on defined security policies and contextual information. The PDP consists of two logical elements. The Policy Engine, which evaluates trust context and applies organisational policies to reach an access decision, and the Policy Administrator, which translates this decision into specific actions and communicates them to enforcement mechanisms. These decisions are then enforced by Policy Enforcement Points (PEPs), which are positioned near the protected resources and are responsible for granting, denying, or terminating access requests [12]. The Policy Engine may consider various inputs when making decisions, including user identity, environmental context, and historical behaviour.

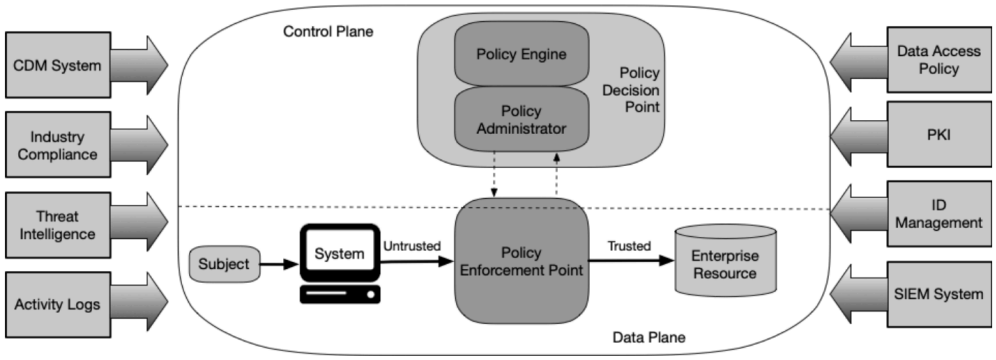

Figure 1. NIST ZTA logical components. [12]

2.3.2. Continuous authentication

Continuous authentication is an important key principle in ZTA, ensuring that trust is not established only at the initial login stage but is instead reassessed throughout the duration of a session. Traditional authentication models typically verify a user's identity once at the beginning of an interaction, after which the user is assumed trusted and allowed access to other resources [12]. In contrast, ZTA systems evaluate the authenticity from contextual information for every

resource request to verify the required authorisation to that resource [12]. If a risk is detected, access privileges may be reduced or revoked. Risk-based trust evaluation limits the duration and scope of access available to compromised devices. By requiring repeat verification rather than relying on a single authentication event, ZTA can reduce the duration during which an attacker could exploit unauthorised access.

2.3.3. Least-privilege and micro-segmentation

Another key principle of ZTA is the enforcement of least-privilege access, which ensures that users, applications, and devices are granted the minimum level of access required to perform their tasks. Instead of providing general access to all resources, access rights are restricted to specific resources and are often limited both in scope and duration [12]. Closely related to this is the concept of micro-segmentation, which divides system resources and network environments into smaller, isolated logical segments by implementing security policies closer to the resource being protected. Each segment is protected by its own access control policies and verification mechanisms, preventing unrestricted lateral movement within the system [14]. According to the NIST model, this can be achieved by bringing the PEP closer to the resource or the data. This allows only authorised entities to access that specific function or piece of data. By combining least-privilege access and micro-segmentation, it is possible to further enhance the confidentiality and integrity of sensitive resources and confine potential breaches to a much smaller subsystem. This layered access control is particularly important in mobile device ecosystems, where devices may frequently connect to and from untrusted networks.

2.3.4. Relevance of ZTA in mobile device environments

Mobile device environments introduce security challenges that differ from those found in traditional enterprise networks [14]. Smartphones and other mobile devices frequently operate across changing network conditions, including public Wi-Fi and cellular networks. Moreover, they interact with external services, devices, and applications. As a result, the assumption of a trusted internal network perimeter becomes difficult to maintain [12]. ZTA provides a security approach that is well suited for such environments because access decisions are based on continuously evaluated context points, such as application signature, geolocation, and identity management. This model allows systems to adapt security controls dynamically both when

developing the application and as conditions change while using it. This is particularly important for mobile device applications that process sensitive resources like location data. While ZTA enables finer-grained access decisions in mobile environments, its effectiveness depends heavily on the feasibility of continuous contextualisation under mobile resource constraints [14].

2.3.5. Limitations of ZTA

Although ZTA can significantly reduce risks when combined with strong identity management, monitoring, and security hygiene practices, it does not eliminate all threats. According to NIST, several risks remain when implementing ZTA, particularly because key decision components such as the Policy Engine and Policy Administrator control access to all resources [12]. If these components are misconfigured, compromised, or disrupted, they can lead to unauthorised access or operational failures. ZTA systems may also be vulnerable to denial-of-service (DoS) attacks that target policy enforcement or disrupt communication paths between components. Additionally, attackers can still exploit stolen credentials or devices, although ZTA can limit the damage. Other risks include limited visibility into encrypted network traffic, the sensitivity of stored monitoring data and policy configurations, and reliance on proprietary technologies that may create interoperability challenges [12]. Finally, the use of automated decision-making systems introduces potential risks related to incorrect policy decisions or the compromise of automated agents. While these issues are not unique to ZTA, they remain relevant regardless of whether ZTA is implemented, and highlight the importance of careful implementation, monitoring, and administration when deploying ZTA solutions. [14].

3. State of the art

Although direct research on this exact topic is limited, this thesis takes inspiration from and builds upon the works of similar papers. To find these papers, various methods were used. Primarily, papers were found with Google Scholar[1] searches for articles. For this, the following search queries were used: “Zero Trust AND mobile applications”, “Zero Trust AND location service”, “Zero Trust architecture”, “mobile applications AND location sharing”, “access control AND mobile applications”, “privacy AND location sharing”. This yielded 20 papers. Further exploration was conducted using backward snowballing, which yielded an additional 20 papers. The papers were selected according to the following inclusion criteria: topic related to Android security, context-based access control, endpoint trust evaluation, Zero Trust, and location sharing; sufficient technical detail; public accessibility; and preferably publication after 2020. The following exclusion criteria were applied: source with only an abstract available; highly specialised military or state-level applications; focus only on legal, ethical, and social implications; sources that propose blockchain related results. These papers were synthesised, and as a result, the list was narrowed to the 27 most relevant papers. In addition to that, two sources from European Law were added to support the legal implication analysis, totalling 29 references.

3.1. Security and privacy in mobile location sharing

Existing research on mobile location sharing security has primarily focused on mitigating privacy risks through data obfuscation, anonymisation, and encryption. Kachore et al. propose location obfuscation to reduce precision before data sharing, thereby limiting the ability of adversaries to infer sensitive user behaviour while preserving basic functionality [15]. Razaghpanah et al. and the official Android Developer platform mention encryption-based approaches, where location data is protected both at rest and in transit using a combination of symmetric and asymmetric cryptography, often combined with trusted third-party key management services [9][10]. Fine-grained sharing policies have also been explored, allowing users to define spatial, temporal, and social constructs to express location [5]. Several reviewed

[1] Google Scholar https://scholar.google.com/ Visited 20.01.2026

approaches rely on relatively static trust relationships, which limit their ability to react to device compromise or contextual changes after initial authorisation. While prior work has advanced privacy preservation, it has generally treated obfuscation, encryption, and access control as separate mechanisms rather than as part of an integrated Zero Trust decision model. Beyond protocol design, research has examined privacy threat models and human factors. Partovi et al. propose dynamic privacy metrics that quantify disclosure risk over time, analysing how privacy guarantees can be maintained in continuous services [16]. Safi et al. highlight that users' privacy decisions and perceptions play a significant role in location sharing practices, indicating that system design must account for both technical and behavioural aspects [17].

3.2. Access control models

Multiple papers have proposed solutions to securing location sharing using a decentralised system to eliminate a central data collection agent that has full control. This reduces the public attack surface by diversifying the data surface. Moreover, effective ZTA implementations rely on secure access control models. In traditional access models, access control lists (ACL), role-based access control (RBAC), context–aware access control (CAAC), and attribute-based access control (ABAC) play a crucial role [18]. However, these usually rely significantly on a central control node. The attribute-based access control model, however, allows for trusted attributes to be computed separately in each domain and common attributes to be shared to ensure trust [19]. ABAC also allows for more granular location data exposure. For example, exact coordinates versus semantic tags like "at work", depending on the risk evaluation based on attributes. The core problem with this is the secure exchange of these attributes without exposing too much data. Shahraki et al. [18] proposed a hierarchical group ABAC framework with the goal of simplifying the complexity of rulesets and administrative access by inheriting these rules top-down based on groups. This results in fewer rules and permissions to track and manage. They found that using Zero-Knowledge proofs to share and verify attributes comes with a decrease in efficiency but a significant increase in security, traceability and reduced central control. These aspects play a crucial role in handling sensitive data, like in their case healthcare, and in this case personal location. Smari et al. [19] proposed an extended ABAC model, which is based on the attributes of every user involved in the process. This considers individual attributes of people, their

systems, the context in which the action is performed, and the purpose with which the action was initiated. This principle has its roots closely tied to ZTA, with the core idea of verifying everything. Verifying every aspect of a user before trusting secures a decentralised system even further. Peebles [20] proposes an Ethereum blockchain-based decentralised system to infuse security with immutability in location data. However, Shafeeq et al. [21] argue that using blockchain for access control means increased transaction fees, delays, computational overhead, and scalability issues. These might not create problems for enterprises with access to a high volume of high-resource computing systems. But it certainly creates problems for resource-constrained mobile devices or IoT devices.

3.3. Zero Trust in mobile systems

Zero Trust Architecture is a security paradigm that operates on the principle of "never trust, always verify" [12]. It eliminates the assumption of implicit trust within the system perimeter and enforces continuous authentication, authorisation, and validation of every user and device [13]. Systems with an implemented ZTA approach focus on protecting assets and processes, not networks. They allow cohesion between enterprise level systems and remote users (BYOD) [12]. Due to the huge surface area for malicious cyber actors to attack various BYOD devices, ZTA is essential for such use cases [22]. The main ZTA implementation techniques include identity management, access control, analysis of network and application logs, internal and external threat intelligence, continuous diagnostics and mitigation, and compliance (Table 1) [14].

Table 1. Core Components of Zero Trust Implementation. [24]

Component	**Primary Function**	**Key Technologies**	**Impact**
Identity management	Authentication, authorisation	MFA, SSO, IAM	High
Network Segmentation	Access Control and Isolation	Micro-segmentation, VLAN	High
Data Security	Information Protection	Encryption, DLP, DRM	Critical
Monitoring	Threat Detection	SIEM, IDS/IPS	High

Tabalipa [23] proposes that ZTA for mobile device applications is divided into six pillars: Runtime protection, device trust, identity assurance, data protection, API security, and continuous monitoring. These approaches are very similar in nature with minute differences in detail.

3.4. Identified research gap

Current mobile device security technologies do not yet widely use ZTA. Techniques like mobile device application vetting, mobile device application management, data isolation, and secure containers allow for secure operations within the device [11]. Candia [11] states that these in-device mechanisms are a strong base for building a ZTA. It is possible to reduce the Android operating system's trusted computing base by decomposing trust into smaller components [25]. Fernandes et al. [25] have proposed a solution which deprivileges a portion of the kernel and divides it into a trusted host kernel (handles memory, UI management) and an untrusted container kernel (handles storage and network files). This segmentation method can effectively block privilege-escalation attacks from malicious applications by not allowing cross-requests to sensitive files. This is important because it reduces the likelihood that the ZTA application itself will be compromised. However, in addition to securing the device's internal functionality, outside access must be considered as well. Qazi [26] proposes that most application programming interface (API) issues in ZTA systems stem from a lack of awareness and are therefore avoidable. The majority of the problems can be resolved by using API gateways [26]. Further, by integrating permission analysis and real-time behaviour monitoring into mobile device environments, it is possible to significantly enhance protection against external malware compared to traditional antivirus or static analysis tools [27]. In practice, implementation of any of these techniques depends highly on the specific mobile device model and version in question [14]. Thus, existing work addresses several parts of the problem, including obfuscation, encryption, access control, decentralisation, and Zero Trust. However, these mechanisms are usually treated separately. The gap addressed by this thesis is the lack of an integrated Android-oriented architecture that combines endpoint-based trust evaluation, adaptive location granularity, secure attribute exchange, and reduced reliance on a central decision-making service.

4. Contribution

This section specifies the limitations and requirements applied in this study. Secondly, the basis for validation is established by specifying the security requirements and different constraints that the proposed prototype needs to adhere to. Further, the architectural and functionality design decisions are explained and analysed. Finally, this section gives an overview of how these design decisions implement ZTA principles by linking PoC elements to specific concepts.

4.1. Design objectives and requirements

This thesis focuses on mobile device-based location sharing systems built for Android devices. It targets the Android 16 operating system, as this is currently the latest version. The research considers ZTA principles as defined by NIST[2] and other academic works; identity, attribute and role-based access control models; trust verification mechanisms. The research excludes large-scale enterprise solutions; highly specialised use-cases, e.g. the military; security of geolocation technologies (GPS, 5G signal triangulation). The PoC location-sharing and trust verification functionality was built as a library that can be used by another application; the rest of the application's development is considered out-of-scope. The research is limited by the following assumptions. The evaluation of the application was executed using controlled test cases, not through the deployment of a real production system with multiple users. The study assumes that baseline mobile device OS security mechanisms are secure regarding application file access and memory access isolation, as well as pre-quantum encryption complexity. Threat modelling focuses on unauthorised access, compromised devices and applications, and weak trust relationships, not state-level hardware attacks.

4.1.1. Security and privacy requirements

A Zero Trust-based location sharing application should be designed so that no user, device, or session is trusted by default. Access to location data must be granted only after explicit

[2] https://www.nist.gov/

verification and should remain subject to continuous reassessment throughout the interaction. Based on this principle, the system should satisfy the following security requirements.

- SR1: Users must be able to control the level of specificity of their outgoing location data, depending on the trustworthiness of the receiving party's device.
- SR2: The application must verify the state of security of the requesting device before granting access to location-sharing functionality.
- SR3: Users and devices must only be granted the minimum level of access necessary for any started task.
- SR4: All authentication and evaluation are performed on-device without reliance on a central decision service, which is only used as a data sharing medium.
- SR5: The application must authenticate every user with the device password before granting access to location-sharing functionality.
- SR6: Any user must be able to control the frequency of authentication needed for accessing their location.
- SR7: All location data and related authentication data must be protected with end-to-end encryption during transmission using modern cryptographic protocols.
- SR8: All location data, authentication data, and policy-related metadata must be encrypted at rest using modern cryptographic protocols.
- SR9: The application must reduce user access to core functionality if suspicious behaviour is detected by policies.
- SR10: A user must be able to block access for any device and remove that device from the registered devices list.
- SR11: The system must maintain secure logs of authentication attempts, policy decisions, and location requests such that confidentiality and integrity are preserved, and only the minimum amount of necessary information is retained.

4.1.2. Performance constraints

Performance is a separate but closely related aspect of usability. An application must remain responsive and must not excessively drain battery or generate unnecessary network traffic. These metrics can highly depend on the device this application is used on, which means they cannot be defined with measurable specifics. Therefore, the following performance constraints are set.

- PR1: The system must introduce minimal latency to any functionality, ensuring near real-time data availability.
- PR2: Continuous authentication and policy evaluation must not significantly reduce application responsiveness.
- PR3: The system must independently scale to support multiple concurrent requests without significant performance reduction.
- PR4: Background operations must run only when needed, not consistently, to reduce unnecessary battery consumption.
- PR5: Network communication must be done at minimal frequency to tolerate irregular network connectivity.
- PR6: Cryptographic operations must be optimised to avoid excessive computational cost.

4.2. Architectural design

This section presents the architectural design of the proposed solution and explains how Zero Trust principles are applied to Android-based location sharing. It defines the trust model, the verification workflow, the policy decision logic, and the secure exchange of trust-related data between devices. Together, these design elements form the basis for reducing centralised trust and enabling adaptive control over disclosed location data.

4.2.1. Trust model

The trust model defines how trust is established, evaluated, and enforced between users, devices, and system components. In this model, access to location data is determined through continuous

assessment of several trust inputs, including user identity, device state, contextual conditions, and behavioural patterns (see Fig. 2).

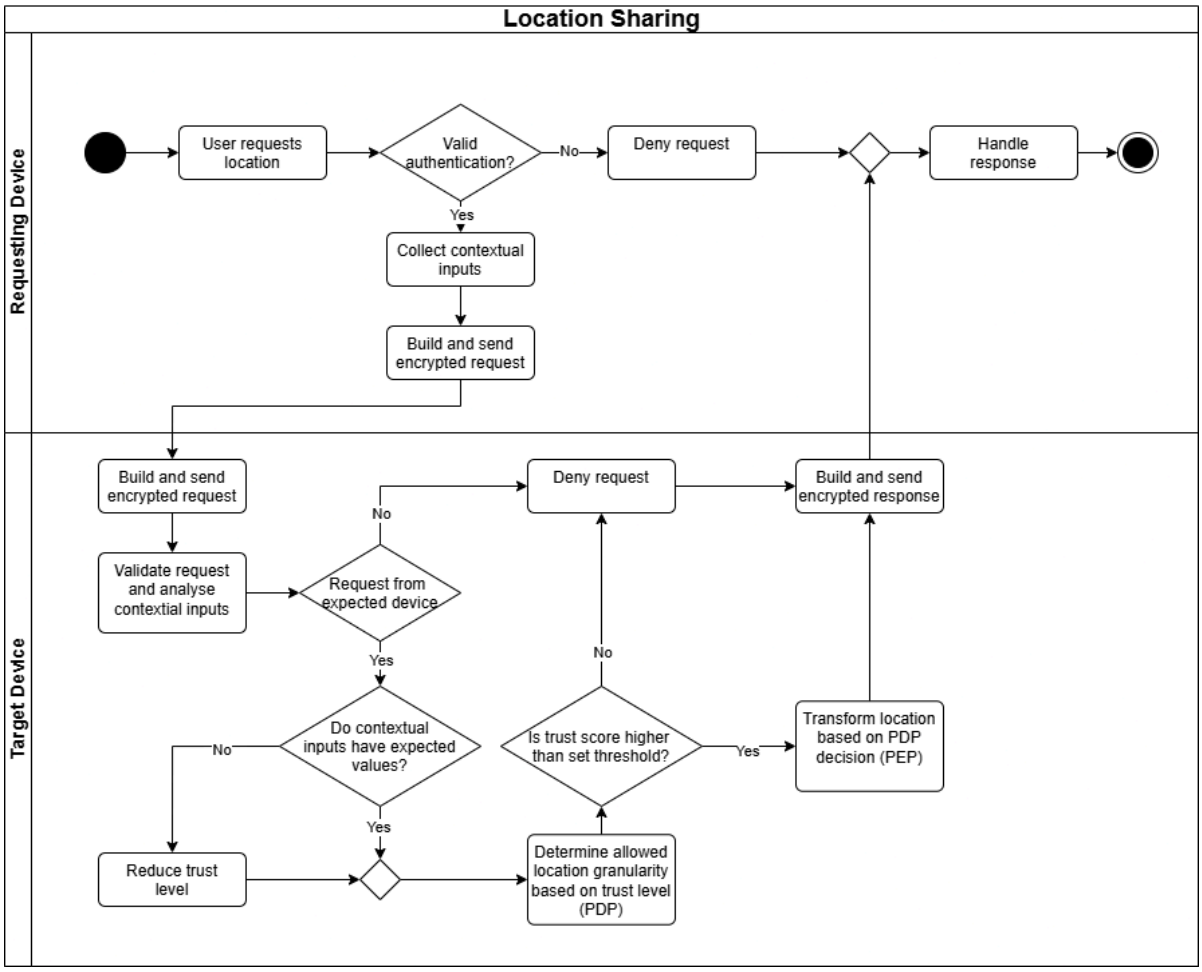

Figure 2. High-level architecture of the trust model. Developed by the author of the thesis.

User identity is verified through device internal authentication, while device trust is evaluated based on security configuration and unique identifiers like public key fingerprinting. Contextual information - such as time, network environment, and geolocation - further influences access decisions. These inputs are processed by the PDP, which determines an appropriate trust level for each request. Based on the evaluated trust level, the system enforces fine-grained access control over location data. Rather than producing only binary allow-or-deny decisions, the model supports multiple levels of data disclosure, based on a predefined user-modifiable set of rules. For example, a fully trusted request may result in sharing precise location coordinates, whereas failing only the time-related policy condition while satisfying the remaining conditions may

result in a semantic location like “at home”. Trust is continuously re-evaluated throughout the session, allowing the system to adapt to each request and, where necessary, revoke or reduce access if risk level increases.

4.2.2. Identity and device verification workflow

The identity and device verification workflow defines how the system authenticates users and evaluates device trust before granting access to location data. In alignment with Zero Trust principles, both user identity and device state are treated as independent factors that must be verified for each access request (see Fig. 2). The process begins with user authentication, in which the requesting party must prove their identity by providing credentials that are checked in-application. This removes the need for central server-side account management and therefore increases the privacy of user identities. Authentication alone is not sufficient to establish trust. Therefore, the target device additionally evaluates the requesting device to determine whether it is present in the trusted devices list, thereby reducing the risk of unauthorised key use, and whether it meets the predefined security requirements. This involves assessing attributes such as application signing certificate hash, operating system version, allowed permissions, and the presence of potential risk indicators such as a rooted operating system. These inputs are collected by default from the requesting device with every request, not actively requested by the target, to reduce the potential depth of data loss from supply-chain attacks. Following successful identity and device verification, the results are forwarded to the PDP, where they are combined with contextual and behavioural data to determine an appropriate level of access. Importantly, this workflow is not limited to the initial access request. In the case of continuous location monitoring, repeated verification is necessary throughout the session.

4.2.3. Policy decision and enforcement

The PDP component is responsible for translating evaluated trust signals into concrete access control actions within the system. In the proposed architecture, this process is driven by a trust score model, which aggregates multiple inputs into a single value representing the overall trustworthiness on a scale of 0 to 100. These inputs are evaluated by the PDP in a manner consistent with the role of the Policy Engine in the NIST Zero Trust Architecture model [12], where access decisions are based on multiple policy-relevant signals rather than a single

authentication event. The score is calculated as a weighted sum of three components: device security state, contextual indicators, and behavioural indicators (see Table 2). The weighting scheme is intended as a modifiable prototype model based on the relative security importance of each category. The potential point values for each trust factor are kept identical so that they sum up to 100 because the relevance of each separate factor is highly dependent on each distinct user's preference. Therefore, the resulting library allows the user to modify these values as long as they still add up to 100, and each major component has at least 10 points achievable. A core distinction is made for rooted OS, mismatched signing certificate hash, and unregistered device. These attributes indicate high risk when present, and the requests including these are therefore automatically denied.

Table 2. PDP main trust score factors. Developed by the author of the thesis.

Component	**Factors**
Device security state	Registered device OS security state Application signing certificate hash Allowed permissions
Contextual indicators	Time Geolocation Network
Behavioural indicators	Frequency of requests Frequency of context change Trust recency

The final trust score is obtained by aggregating these values, allowing the system to represent trust as a continuous and comparable metric. This score then serves as the primary input for policy decision logic, which maps predefined score ranges to access outcomes. Requests with scores below 70, or with fewer than 10 points in any category, are denied access. Scores between 70 and 79 permit access only to semantic location, scores between 80 and 89 permit approximate location, and scores between 90 and 100 permit precise coordinates. These thresholds are intended as conservative policy boundaries between disclosure levels. If the thresholds were lower, too many high-risk missing factors could still result in a precise location policy. If the thresholds were harsher, it would be too difficult for a real trusted user to reliably achieve a precise location policy outcome due to variation in non-controllable parameters, like a rare need

to request location during nighttime. Enforcement is then performed directly in the data handling process through the PEP component, which transforms the location data according to the assigned policy before it is returned to the requester.

4.2.4. Secure attribute exchange

Secure attribute exchange defines how trust-related data is collected, transmitted, and validated between participating devices. In the proposed architecture, authentication is performed locally on the requesting device, without the need for a central authentication service. This reduces reliance on centrally managed user accounts and limits unnecessary exposure of user identity information. Password validation is enforced on the OS level using the native device password or biometrics. All subsequent validation and policy enforcement is done on the target device that owns the location data. A central service is used solely as a relay for data exchange and is not intended to observe or meaningfully modify plaintext data in transit. As a result, all attributes used for trust score calculation must be securely transferred between devices and must be decryptable by a shared key. Key sharing is done during initial device registration, which has to be completed in-person. To ensure integrity and confidentiality, attributes generated on the requesting device are cryptographically protected before transmission. This includes signing them with the sender's private key and encrypting the result with the shared key. Additionally, attributes are bound to specific requests and include validity timestamps to prevent replay attacks. Upon receiving a request, the target device decrypts the message, validates the signature, and only then starts the trust score attribute validation. Only minimal metadata, such as timestamp, origin alias, and result, is retained on the target device. After a decision is made in the PDP, the PEP signs and encrypts the result as described above and then sends it back to the requesting device via the central service.

4.3. Software library design

This section describes how the proposed architecture is realised as an Android software library. It explains the main functional components of the library, the intended integration model for host applications, and the handling of location data and policy evaluation within the implementation.

The purpose of this design is to translate the architectural principles described earlier into a reusable and practical software component.

4.3.1. Core components

The PoC is implemented as a software library consisting of several core components. The first component is the authentication module, which is used mainly when opening the application. This module is responsible for verifying user identity and thereby denying or allowing access to the application. This module is invoked every time a user opens the application or tries accessing the application functionalities when the application has been open for a prolonged time. The authentication module prompts the user for the device's native password, which is validated by the OS. The second module is the client interface module, which enables the application to both initiate a request workflow and also receive the response. Thirdly, the attribute collection module gathers relevant trust inputs from the device and packages them as a signed message before relaying that message to the data transfer module. The fourth module, the data transfer module, is responsible for sending data through a central service to the target device. This module performs the final end-to-end encryption of the signed message and exchanges HTTPS requests with the central service. The remaining modules are primarily used on the target device during a standard request workflow. The fifth module is the validation module. It is responsible for decrypting the message and verifying the authenticity, integrity, and freshness of all incoming attributes. This ensures that only correctly sourced and up-to-date data is used in further processing. The validated attributes are then passed to the PDP module, which computes the trust score based on predefined weights. This component functions as the primary decision-making component for the library. Finally, the PEP module receives the trust score from PDP and applies the corresponding policy by collecting location data from the device using the attribute collection module and transforming it into a suitable format. This package is then signed and sent to the data transfer component, where it is encrypted and sent back to the requesting device. There it is decrypted by the data transfer module and finally processed by the client interface module to display the result. By separating functionality into these distinct components, the library ensures clarity of responsibility, modularity, and easier integration into applications.

4.3.2. Integration model for existing applications

The proposed software library was designed for straightforward integration into existing mobile applications without requiring significant architectural changes. Integration is achieved through an API that abstracts the underlying security mechanisms and exposes simple methods for requesting and receiving location data, and modifying the trust score settings. On the requesting side, applications invoke the library to initiate a location request, while the library handles authentication, attribute collection, and secure attribute exchange transparently. On the target side, the library integrates with existing location services and processes incoming requests, ensuring that all access decisions and policy enforcement are applied without the need for user interaction. The library is thus designed to operate independently of application-specific logic, allowing developers to incorporate it without substantially modifying existing application logic. This also helps reduce the risk of developers inadvertently making insecure security implementations. Additionally, this ensures that applications only receive validated and policy-compliant location data, which further simplifies integration by reducing the need for application side data parsing. To support the whole range of Android devices and the latest OS versions, the library relies on standard mobile platform features and avoids specialised dependencies. This allows it to be deployed across different environments with minimal adaptation.

4.3.3. Location data handling

On the target device, raw location data is obtained from the device's native location services and maintained in its most precise available form internally. When a response is generated, the system applies transformation functions to adjust the level of detail according to the assigned policy. These transformations may include coordinate rounding to reduce precision, spatial generalisation to a predefined area, for example, a city, or mapping coordinates to semantic labels such as "at home" or "at work". By performing these operations locally, the library ensures that sensitive raw data is never unnecessarily exposed or transmitted. Additionally, consistency is maintained between different levels of representation by ensuring that all derived location outputs originate from the same underlying data source. This prevents discrepancies between precise and generalised data and supports predictable outcomes. Location data is also

handled in a stateless manner for each request, avoiding long-term storage of historical location information within the device unless explicitly done so by the surrounding application.

4.3.4. Policy evaluation

Policy evaluation, as a part of the PDP module, defines how access control policies are structured, interpreted, and applied to incoming requests. Policies are defined as a set of conditional rules that describe how different combinations of identity, device, and contextual attributes influence the resulting access level. These rules are evaluated in a deterministic manner to ensure consistent behaviour across all requests. In the proposed architecture, policy evaluation operates by matching incoming requests against a predefined set of policy conditions. Each policy specifies constraints related to factors such as user relationships, permitted access contexts, or acceptable device conditions. Once a request satisfies the conditions of a given policy, the corresponding amount of points is added to the trust score of that request. This allows the system to support multiple policy profiles, enabling different behaviours depending on the context, such as stricter rules for more distant relationships and more permissive rules for closely trusted contacts such as family members.

4.3.5. Cryptographic protocol design

Each device maintains two long-term asymmetric key pairs in the Android Keystore. The first key pair is used for digital signatures, while the second key pair is used for encryption-related key agreement. Both key pairs use the P-256 elliptic curve, also known as secp256r1. The signing key is used with ECDSA and SHA-256 to authenticate request and response claims. The encryption key is used with Elliptic Curve Diffie-Hellman to establish a shared secret for each encrypted message. Where supported by the device, these keys are hardware-backed, and the availability of hardware-backed key storage is also included as one of the evaluated trust inputs.

For message confidentiality, the library uses a hybrid encryption construction based on P-256 ECDH, HKDF-SHA256, and AES-256-GCM. For every outgoing encrypted payload, the sender generates a fresh ephemeral P-256 key pair and performs ECDH with the recipient device’s long-term encryption public key. The resulting shared secret is passed through HKDF-SHA256 with a random salt and contextual information, binding the derived key to the sender and

recipient device identifiers. The derived 256-bit symmetric key is then used with AES-GCM to encrypt the signed payload. AES-GCM provides both confidentiality and integrity protection for the encrypted message body.

The encrypted envelope contains the algorithm identifier, ephemeral public key, salt, IV, ciphertext, sender device identifier, and recipient device identifier. The recipient uses its Android Keystore-protected private encryption key and the included ephemeral public key to derive the same AES key, decrypts the payload, and then validates the embedded digital signature. This means that encrypted transport protection and explicit sender authentication are both applied before any trust attributes are accepted for policy evaluation.

Location requests are bound to a unique session identifier and timestamp. The timestamp is used to reject stale requests, while the session identifier is stored locally after processing to prevent replay of the same request. Responses follow the same protection pattern: the target device signs the policy result and transformed location payload, encrypts the response to the requester's public encryption key, and returns the encrypted envelope through the relay service. The requesting device then decrypts the response and verifies the target device's signature before exposing the result to the surrounding application.

The same cryptographic mechanisms are also used for non-standard application payloads, allowing the library to protect arbitrary messages between paired devices. This keeps the integration model consistent: application developers do not need to implement separate encryption or signing logic for custom payloads.

Device pairing is secured through public key fingerprint verification. During pairing, the library computes a SHA-256 fingerprint over the signing and encryption public keys of the device. This fingerprint can be exchanged through an out-of-band channel, such as a QR code workflow, allowing users or the surrounding application to verify that the public keys being registered belong to the intended device. This reduces the risk of registering attacker-controlled keys during initial trust establishment.

The library also includes key lifecycle management for long-term device keys. Each device tracks a key version, original key creation time, last rotation time, and optional rotation reason as part of its local encrypted state. When key rotation is triggered, the SDK deletes and regenerates both the signing key pair and the encryption key pair in the Android Keystore, producing new

P-256 public keys and a new pairing fingerprint. The updated registration material is returned to the surrounding application so that previously paired devices can re-register or verify the new keys through the pairing workflow. Since rotated keys replace the previous private key material, devices that still hold the old public keys can no longer validate signatures or encrypt new messages to the rotated device until the pairing record is updated. Key lifecycle events, including rotation, paired-device key updates, revocation, and local key material clearing, are recorded in the secure audit log, providing traceability for trust-establishment changes.

Data stored locally by the library is also encrypted at rest. User metadata, paired device information, semantic location labels, session records, replay markers, and audit logs are stored using AES-GCM encryption with a symmetric key held in the Android Keystore. Each stored value is encrypted with a fresh IV. This protects policy-related metadata and local trust state from direct inspection if the application storage is accessed outside the normal application boundary.

The library additionally maintains a secure audit log for important security events, including user setup, authentication attempts, request creation, policy decisions, response creation, and response processing. Audit entries are encrypted at rest, hash-chained using SHA-256, and signed using the device signing key. This provides tamper evidence for local security records: modifying an earlier entry invalidates the hash chain and signature relationship of the log. The audit log is not intended to prevent deletion of all local data, but it improves integrity protection for retained records.

5. Validation

This chapter evaluates the proposed Zero Trust-based location-sharing architecture in order to assess its effectiveness, feasibility, and alignment with the defined requirements. The validation focuses on analysing how the system behaves under different access scenarios, both default and malicious use cases. This section begins by describing the threat model and defining the evaluation criteria used. This is followed by a security analysis to evaluate the system's resilience against potential attack patterns. After that, the practical testing methods used to evaluate the proof-of-concept are described. Finally, the chapter discusses the limitations of the validation approach, highlighting assumptions and constraints that may affect the results.

5.1. Validation strategy

This section defines the validation approach used to evaluate the proposed architecture. It introduces the criteria, assumptions, and threat model that form the basis for assessing whether the design satisfies its stated security and functional objectives.

5.1.1. Evaluation criteria

The evaluation of the proposed architecture is based on a set of criteria that reflect its security, functionality, and performance objectives. These criteria are used to assess whether the library operates as designed and, more importantly, whether it effectively addresses the challenges associated with secure mobile location sharing. First, the correctness of access control is evaluated by verifying that the system consistently applies the defined required policies based on the calculated trust score. This includes confirming that different trust levels result in the appropriate level of location data exposure. Closely related to this is the granularity of data control, which assesses the system's ability to adjust location precision according to policy requirements without producing inconsistent or incorrect results. Secondly, evaluation considers adaptability to changing conditions, focusing on the system's ability to update access decisions in response to changes in device state, context, and user behaviour. This criterion is particularly important for evaluating the effectiveness of continuous authentication and dynamic trust evaluation mechanisms. Thirdly, security effectiveness is evaluated by examining whether the

system prevents unauthorised access, limits excessive data exposure, and ensures that only validated and trustworthy inputs influence decision-making. This includes assessing the robustness of attribute validation, resistance to tampering, and enforcement of least-privilege access. Finally, performance impact is considered to determine whether the introduced security mechanisms remain suitable for mobile environments. This includes observing any noticeable delays in request processing, computational overhead on devices, and responsiveness of location-sharing operations. Together, these criteria provide a structured basis for evaluating both the functional behaviour and security properties of the proposed system, ensuring that the validation covers the key objectives defined in the design section.

5.1.2. Threat model

The threat model defines the potential adversaries, attack surfaces, and security assumptions relevant to the proposed architecture. Since the proposed system operates mainly on end devices, with data being communicated through a central server, the threat model focuses on both facets. The primary adversaries considered in this model include the following: Malicious MitM actors who try to intercept or manipulate communication in transit, malicious or compromised users attempting to access data beyond their privileges, compromised devices that may provide falsified trust attributes, and stolen devices that try to use valid credentials from a registered device. Thus, the primary attacks considered are communication interception, unprivileged authentication, replay attacks, and trust attribute injection/modification. The relay server is treated as an untrusted component, meaning it is not assumed to enforce security policies or maintain data confidentiality beyond basic transport guarantees. Consequently, all data that leaves towards the central server is assumed breached and is therefore cryptographically protected before transmission. Similarly, all data that is sent to a device from the central server is assumed to come from a breached state and therefore needs to be validated before any operations. The model assumes that mobile devices and concurrently their operating systems provide baseline security mechanisms, such as secure key storage, application sandboxing, and secure computation. If any of these fail, the proposed architecture cannot work as intended. For example, mobile operating systems enforce application sandboxing to prevent unauthorised memory access, yet this protection can be weakened on rooted or otherwise compromised devices. In such cases, an attacker may be able to extract sensitive data directly from application

memory or intercept it through runtime manipulation. Since data must eventually be exposed in plaintext to the receiving application, that rogue application can steal it. Furthermore, it is assumed that cryptographic functions are securely implemented and cannot therefore be used as attack vectors. Physical attacks on devices, such as hardware tampering, are considered out of scope.

The threat model is designed around the STRIDE[3] framework. STRIDE was selected as the threat modelling framework because the proposed solution is primarily a software architecture with identifiable components, data flows, and trust boundaries. The system includes requesting devices, target devices, an untrusted relay server, encrypted request and response messages, local storage, policy decision logic, and policy enforcement logic. STRIDE is suitable for this type of analysis because it provides a structured method for identifying threats across six broad categories: spoofing, tampering, repudiation, information disclosure, denial of service, and elevation of privilege. These categories correspond closely to the security concerns of the proposed architecture: authentication and device impersonation, modification of trust attributes or responses, accountability for requests and policy decisions, confidentiality of location data, availability of the relay channel, and attempts to obtain more precise location data than permitted. STRIDE was also chosen because the validation is performed at the architectural and prototype level rather than through a production deployment. More quantitative or process-heavy threat modelling methods would require operational data, attacker probability estimates, or organisational context that is outside the scope of this thesis. STRIDE therefore provides an appropriate balance between structure and practicality. It enables the thesis to reason systematically about relevant attack classes while remaining compatible with the prototype-oriented validation strategy.

Spoofing threats primarily target the authentication process on the requesting device. An attacker may attempt to impersonate a legitimate user by using stolen credentials. Additionally, an attacker could attempt to impersonate a trusted device by submitting forged device attributes. To mitigate this, the system relies on strong OS-native authentication mechanisms and cryptographically verifiable tokens that bind identity and device information to each request. Tampering threats arise in the communication between the requesting device and the target

[3] https://owasp.org/www-community/Threat_Modeling_Process#stride

device via the relay server. Since the server is not trusted, an attacker controlling or intercepting traffic could attempt to modify transmitted attributes or alter location data responses. For example, manipulated device posture data could artificially increase the trust score. To prevent this, all exchanged attributes and responses are cryptographically protected, allowing the target device to detect any unauthorised modifications before performing policy evaluation. Repudiation threats occur when a user denies having made a location request or shared data. This makes repudiation relevant for both the requesting and target devices. To address this, the library maintains local signed logs of authentication events, requests, and policy decisions, enabling verification of actions while ensuring its integrity. Information disclosure threats are most prevalent in this subject area, as location data is highly sensitive. Attackers may attempt to intercept communication (e.g. MitM attacks) or exploit the relay server to access data in transit. Additionally, as previously identified, a compromised requesting device may expose location data after decryption through unauthorised memory access by malicious processes. The library reduces these risks by encrypting all communication, minimising data depth based on policies, and restricting access to data altogether for untrusted entities. Denial of service threats affect the availability of data requests and responses, particularly through the disruption of the central server or communication channels. An attacker may block or delay requests, preventing basic functionality. While this impacts usability, it does not compromise data confidentiality or integrity. This can only be mitigated through alternate data channels and fallback servers in separate physical locations. However, infrastructure setup is outside of the scope of the development for this PoC library. Elevation of privilege threats occur when an attacker exploits authentication to gain higher access than permitted. In the proposed architecture, this can be done by providing misleading attributes to increase the granted trust score or by evading the authentication of a stolen device (for example, guessing the password). This risk is reduced by strict validation of all checkable input attributes. It is important to distinguish that not all aspects of the attributes are validatable. For example, the cryptographic validity of the signature on the attributes can be checked, but location spoofing is hard to detect.

The use of STRIDE also introduces limitations. STRIDE helps identify broad classes of threats, but it does not by itself quantify likelihood, prove the absence of vulnerabilities, or fully capture privacy inference risks caused by repeated authorised access. Therefore, the STRIDE analysis is used as a structured security analysis tool rather than as evidence of complete security. Residual

risks such as endpoint compromise, misleading device attributes, indirect behavioural inference, and relay availability remain relevant and require additional evaluation in production-oriented future work.

5.2. Security analysis

This section examines the security properties of the proposed architecture by analysing how it responds to representative threats and attack scenarios. The purpose is to determine whether the design meaningfully reduces important risks and improves the placement of trust-sensitive functions compared with more centralised approaches.

5.2.1. Attack scenario evaluation

To evaluate the security of the proposed architecture, several attack scenarios are analysed to assess how the architecture responds to potential threats. These scenarios focus on attempts to bypass trust evaluation, access unauthorised data, or manipulate general system behaviour. Each scenario is evaluated in terms of attack feasibility, system response, and resulting impact, using a coarse 1–3 scale (1 - low, 2 - medium, 3 - high) for impact and feasibility (see Table 3). The risk value for each scenario is then calculated as impact multiplied by feasibility.

Table 3. Risk levels of considered attack methods.

Attack type	**Impact**	**Feasibility**	**Risk (impact * feasibility)**
Authentication with stolen credentials	2	2	4
MitM data breach	2	2	4
Attribute manipulation	3	2	6
Endpoint compromise	2	1	3
DoS	1	2	2

The first scenario considers an attacker using stolen credentials to impersonate a legitimate user on that user’s stolen device. In the proposed architecture, authentication is performed locally on the requesting device and is bound to that specific device. As a result, stolen credentials alone are not sufficient to authenticate from another device. However, this attack remains feasible if the

attacker gains access to the legitimate user's device, for example, through theft or remote compromise. In that case, the attacker is able to initiate valid requests. This risk is difficult to mitigate using device posture evaluation. However, each user is enabled to remove any device from their registered devices list, for example, if the owner of the stolen device reports it to that user. This attack is plausible if users do not securely manage their credentials and devices. Furthermore, the proposed system cannot adequately address this situation in any reasonable way before first harm is done. However, the impact is bound to only those users that have allowed that device into their registered devices list, no other users are impacted.

The second scenario considers a MitM attack. This scenario has two cases - whether or not the attacker has gained access to any private keys used in a particular data exchange. If the attacker does not have access to any private keys, any attempts to read data in transit are ineffective, since all data is cryptographically secured before transit. Therefore, confidentiality of the data remains intact. Moreover, if the attacker tries tampering with the data, since the data is signed and encrypted, any changes are immediately obvious to the target device after initial cryptographic validation. In the case that the attacker has access to a user's private key, both the confidentiality and integrity are affected for the user whose private key has leaked. That could mean exposed location data for the target user or exposed device attributes for the requesting user. In both cases, availability can be impacted by modifying the data randomly, making it unreadable, or by not allowing requests or responses to reach their respective destinations. This attack is feasible, if the central relay server has internal weaknesses and can therefore be maliciously taken over. Mitigating this is outside of the scope of this work.

The third scenario considers attribute manipulation, where an attacker attempts to increase their trust score by submitting falsified device or contextual attributes. For example, a compromised device may report itself as not rooted or located in a secure network. This is mitigated through a combination of consistency checks. This means comparing previous request attributes to the current ones and checking if any remarkable differences exist. As previously mentioned, not all attributes can be verified to an equal degree, but risk is lowered by reducing the trust score, if any of the attributes show inconsistencies with the logged history. The system assumes that a fully compromised device cannot be trusted, and therefore limits the potential impact of such compromise rather than attempting to fully prevent it. These attacks are plausible when the

attacker operates from a rooted or otherwise compromised device. The impact is severe, since the attacker can spoof trusted attributes to any target they have access to and receive their location.

The fourth scenario considers endpoint compromise on the requesting device, where a malicious process attempts to access location data after it has been decrypted. While sandboxing in mobile devices usually prevents such access, this protection may be bypassed on rooted devices. The proposed architecture cannot prevent this type of attack, as data must be available in plaintext to the requesting application. However, the impact is reduced by restricting data exposure based on trust level, ensuring that compromised devices are less likely to receive highly precise responses.

Finally, a denial of service scenario is considered. In the case of disruption or prevention of data transfer on the relay server or between devices, the attacker may prevent location requests or responses from being delivered. This is plausible in the case of either a larger network attack or the takeover of the relay server. While this impacts availability, it is not compromising confidentiality nor integrity, as sensitive operations are performed on endpoint devices and no trust is placed in the relay server.

5.2.2. Attack surface reduction analysis

The proposed architecture reduces the attack surface by limiting both the number of trusted components and the amount of data exposed at different stages of the process. In conventional location-sharing systems, central backend services often handle user authentication, session management, authorisation logic, and storage of shared location data. This creates a broad and concentrated attack surface, since compromise of a single backend component may allow an attacker to access multiple users' locations, manipulate access control, or tamper with legitimate responses to send out false data. In such architectures, the server is not only a communication relay but also a trusted decision-making component, which makes it a high-value target. In the proposed architecture, this centralised attack surface is reduced at the architectural level by limiting the server's role to message relay only. The server does not authenticate users, calculate trust scores, enforce access policies, or evaluate attributes. Most importantly, it does not require access to raw location data before policy decisions are made. This means that even if the relay server is compromised, the attacker does not automatically gain access to any of the user's data. The architecture also reduces the exposure of location data during communication. Instead of

always sharing exact coordinates, the target device evaluates the request locally and only then generates an output that matches the permitted level of exposure. This means that the most specific and thus most sensitive representation of data remains on the target device and is not unnecessarily transmitted through the network or processed by other devices. From an attack surface perspective, this is important because it reduces the value of intercepted traffic and limits what an attacker can obtain in the worst-case event of traffic interception. Further, in the proposed design, each location request is evaluated independently using changing trust-related inputs, which reduces the attack surface associated with long-lived permissions, reused sessions, or overprivileged access. A compromised session is less useful if access is continuously re-evaluated and if each request produces only a limited output based on current trust conditions. Similarly, the design is intended to make replay attacks less effective by binding requests to freshness data.

However, the proposed architecture does not eliminate all attack surfaces. By moving validation, trust evaluation, and authentication to endpoint devices, the security of those devices becomes more critical. A compromised requesting device may expose location data after it has been decrypted for real use, and a compromised target device may manipulate responses or interfere with enforcement logic. These risks cannot be fully removed because the system ultimately depends on endpoint devices to process and display the data. However, such a compromise is more likely to affect an individual interaction or device rather than the entire user base. For this reason, the proposed architecture is designed for reducing centralised and high-impact attack surfaces while accepting that endpoint-level attack surfaces remain a residual risk. Further, controlling direct access to location data does not fully prevent secondary inference by the receiving application. Even when the shared output is restricted to approximate or semantic information, an application may still infer movement, direction, or changes in user context from auxiliary signals available on the device. Such signals may include repeated location responses over time, motion-related sensor data, device orientation, battery consumption, or other observable resource patterns. However, solving this is considered outside of the scope of this thesis. Overall, this is a meaningful architectural improvement within the stated threat model, because the system avoids implicit trust in shared infrastructure and limits the attack scope wherever possible.

5.3. Prototype testing design

This section describes how the proof-of-concept implementation was evaluated in practice. It presents the prototype setup and the test scenarios used to assess whether the implemented workflow, access-control logic, and trust evaluation behaviour operate in accordance with the proposed design. The results of these validation tests are reported and analysed in Section 6.1, including the trust-score test results and single-factor degradation results shown in Tables 5 and 6.

5.3.1. Prototype setup

To examine the prototype-level implementability of the proposed architecture, a prototype implementation was developed and tested in a controlled development environment. The prototype was implemented in Kotlin[4] using Android Studio Panda 3[5], which provides a suitable platform for building and evaluating mobile application libraries in an Android-based context. OpenAI Codex[6] (GPT-5.5) was used during prototype development as an AI-assisted coding tool to support implementation tasks. Codex was used only as a development aid; the architectural decisions, security requirements, policy logic, validation criteria, and final review of the implemented code remained the responsibility of the author. All AI-assisted code was manually inspected, adapted where necessary, and tested within the controlled prototype environment before being used in the validation activities described in the following sections. The core functionality of the proposed solution was developed as a separate Android Library project[7] rather than a standalone application. This design choice supports modularity and reflects the intended real-world use case, as the contribution of this thesis is not a complete application, but a reusable software component that can be integrated into existing applications. To evaluate the prototype, a separate Android Studio project was used that links the library using Maven, allowing the library to be imported as a dependency in a reproducible way, and therefore enables the simulation of real user behaviour to make function calls to the library. This again serves as an

[4] https://kotlinlang.org/ (Viewed 12.03.2026)

[5] https://developer.android.com/studio (Viewed 12.03.2026)

[6] https://openai.com/codex/ (Viewed 20.05.2026)

[7] https://github.com/villemsusi/ztaloc

intended use case where a developer integrates the library into their application. This setup made it possible to evaluate both the internal functionality of the library as well as its suitability for integration into existing applications. However, the following sections focus on functionality testing, since this is the core objective of this thesis.

5.3.2. Workflow validation

Workflow validation was used to verify that the implemented prototype behaves according to the intended system design and that its components interact correctly with each other. The purpose of this stage was not to evaluate large-scale performance or real-world deployment, but to confirm that the library correctly implements the architectural logic proposed in this thesis. In particular, the validation focused on whether requests are created and transmitted correctly, whether trust-related inputs are collected and delivered in the expected format, whether the target device can process the request and evaluate these inputs, and whether the resulting location data is returned to the requesting application correctly. The validation was performed by executing representative request-response scenarios. A successful result was defined as a complete and correct flow from the initiation of the request to the display of the resulting location. This includes verifying that authentication on the requesting device produces the request artifacts, that the necessary trust inputs are collected, that all of the data is signed and encrypted before transit, and that the target device performs validation, trust evaluation, policy enforcement, signing, and encryption before returning a response. In addition, the requesting device must correctly receive, validate, and expose the returned location data to the integrating application. For this, a collection of manual and automatic tests was designed and executed to cover the entire functionality range.

5.3.3. Access control validation

Access control validation was used to verify that the prototype enforces the intended security policies correctly under different trust conditions. While workflow validation shows that the overall request-response flow operates as expected, this stage focused specifically on whether the system produces the correct authorisation outcome when trust-related inputs vary. The objective was to determine whether the implemented trust score model and policy mapping logic behave consistently with the previously defined design. These tests were performed by constructing a set

of trust attribute collections, in which the values for all attributes are varied in a controlled manner. For each scenario, the resulting trust score was observed and compared against the expected outcome. For example, a request originating from a properly authenticated user on a registered and uncompromised device and under normal contextual conditions was expected to produce a high trust score and allow access to precise location data. In contrast, requests involving untrusted device conditions were expected to result in lower trust scores and correspondingly more restrictive outputs. This also includes testing use cases where a very low trust score denies access to location data altogether.

5.3.4. Continuous trust re-evaluation

Continuous trust re-evaluation was used to verify that the prototype can adapt access decisions when trust conditions change quickly. The purpose of this validation stage was to confirm that the implementation does not treat trust statically, but instead updates access outcomes continually. The tests in this stage were based on scenarios where a request initially satisfies the conditions for a given level of access, but one or more trust inputs change shortly after. Such changes may include any of the previously specified trust conditions. For each scenario, the system was observed to determine whether the updated inputs are required, validated, and processed for each new request. Further, it was tested whether a new trust score is produced and a corresponding new policy applied. The expected behaviour was that a decrease in trust leads to a corresponding reduction in the level of permitted data exposure and vice versa.

5.4. Requirements completion verification

This section verifies whether the requirements defined in Section 4.1 were satisfied by the proposed architecture and prototype (see Table 4 and Table 5). The purpose of this verification is to connect the design objectives to concrete validation evidence. A requirement is considered satisfied when it is supported by the architecture and verified through prototype testing, workflow validation, access-control testing, cryptographic design analysis, or structured security analysis. A requirement is considered partially satisfied when the proposed design supports it, but the prototype validation does not provide sufficient evidence for full confirmation.

Requirements that are not directly tested are marked as not validated and are discussed further in the limitations section.

Table 4. Security and privacy requirements.

ID	Requirement	Mechanism in the proposed design	Verification	Status
SR1	Users must be able to control the level of specificity of their outgoing location data, depending on the trustworthiness of the receiving party's device.	PDP trust score maps requests to deny, semantic, approximate, or precise disclosure; PEP transforms location data before response.	Access-control validation using different trust-score scenarios	Satisfied
SR2	The application must verify the state of security of the requesting device before granting access to location-sharing functionality.	Device attributes include registration status, OS security state, signing certificate hash, permissions, and rooting indicators.	Workflow validation and attack-scenario analysis	Satisfied
SR3	Users and devices must only be granted the minimum level of access necessary for any started task.	Trust-score thresholds restrict location precision according to current trust level.	Access-control validation	Satisfied
SR4	All authentication and evaluation are performed on-device without reliance on a central decision service, which is only used as a data sharing medium.	Requesting device handles user authentication; target device performs validation, PDP, and PEP enforcement; relay only transfers encrypted messages.	Architecture review and workflow validation	Satisfied
SR5	The application must authenticate every user with the device password before granting access to location-sharing functionality.	OS-native device authentication is invoked before protected functionality.	Workflow validation	Satisfied
SR6	Any user must be able to control the frequency of authentication needed for accessing their location.	The authentication module supports repeated authentication after configured time or protected action.	Design review	Partially satisfied
SR7	All location data and related authentication data must be protected with end-to-end encryption during transmission using modern cryptographic protocols.	ECDH, HKDF-SHA256, AES-256-GCM, ECDSA signatures, HTTPS relay	Cryptographic protocol analysis and tamper/replay reasoning	Satisfied

.

Table 4. Security and privacy requirements

SR8	All location data, authentication data, and policy-related metadata must be encrypted at rest using modern cryptographic protocols.	AES-GCM local storage with Keystore-protected key	Design review	Satisfied
SR9	The application must reduce user access to core functionality if suspicious behaviour is detected by policies.	Behavioural indicators reduce trust score; low score causes deny or reduced disclosure	Repeated trust re-evaluation tests	Partially satisfied
SR10	A user must be able to block access for any device and remove that device from the registered devices list.	Registered device list and revocation mechanism	Revocation test	Satisfied
SR11	The system must maintain secure logs of authentication attempts, policy decisions, and location requests such that confidentiality and integrity are preserved, and only the minimum amount of necessary information is retained.	Encrypted, hash-chained, signed audit log	Design review and integrity analysis	Partially satisfied

Table 5. Performance requirements

ID	**Requirement**	**Verification**	**Status**
PR1	The system must introduce minimal latency to any functionality, ensuring near real-time data availability.	Prototype timing measurements	Partially satisfied
PR2	Continuous authentication and policy evaluation must not significantly reduce application responsiveness.	Measured request processing time	Partially satisfied
PR3	The system must independently scale to support multiple concurrent requests without significant performance reduction.	Not directly tested	Not validated

Table 5. Performance requirements

PR4	Background operations must run only when needed, not consistently, to reduce unnecessary battery consumption.	Design review	Partially satisfied
PR5	Network communication must be done at minimal frequency to tolerate irregular network connectivity.	Design review	Partially satisfied
PR6	Cryptographic operations must be optimised to avoid excessive computational cost.	Timing measurement of request/response operations	Partially satisfied

The verification indicates that the core architectural requirements were satisfied at the prototype level, particularly those related to endpoint-based trust evaluation, reduced reliance on a central decision service, adaptive disclosure, and cryptographic protection of exchanged data. However, some requirements are only partially satisfied because they were validated through design analysis rather than repeated empirical testing. In particular, concurrency, battery impact, long-term background behaviour, and robustness across different Android devices require additional evaluation.

5.5. Limitations

Although the proposed architecture provides a structured approach for applying Zero Trust principles to location sharing, the validation presented in this thesis is subject to several limitations. First, the evaluation was based on a PoC implementation rather than deployment in a real-world production environment. As a result, the findings show technical feasibility and architectural correctness, but they do not fully capture how the system would behave under large-scale usage, concurrent requests, diverse network conditions, or long-term maintainability. Secondly, the prototype was developed and tested in a limited Android-based environment. This means that the results may not generalise directly to other platforms, device models, or operating system versions with different security functions and hardware capabilities. In particular, device integrity checks, local authentication behaviour, and secure storage mechanisms may vary significantly across implementations on different platforms. Thirdly, the trust score model in this thesis is based on reasoned design choices rather than empirical calibration. The selected default

weights, thresholds, and scoring logic are intended to provide a practical and intuitive basis for policy decisions, but they have not been optimised through rigorous measurements or case studies. Consequently, while the scoring model is suitable for demonstrating the architecture, the exact point values should not be interpreted as universally optimal. Finally, the validation does not include a formal usability study or dedicated user testing. Although usability concerns are considered in the design, the thesis does not empirically measure how end users would perceive the functionality during normal use. These limitations, however, do not invalidate the contribution of the thesis, but they define its scope. The results of validation should therefore be understood as evidence that the proposed architecture is technically plausible, rather than proof of complete real-world effectiveness across all deployment conditions.

6. Discussion

This chapter interprets the results of the validation and considers their broader meaning in relation to the research question and thesis contribution. It discusses what the proposed architecture improves, what limitations remain, how it compares with the state of the art, and what future work is needed to further develop the approach.

6.1. Result analysis

The main research question of this thesis was how Android-based location sharing can be redesigned using Zero Trust principles to reduce reliance on centralised trust and support adaptive disclosure of location data. The results indicate that this objective was addressed at the level of architectural design and prototype-oriented validation. More specifically, the thesis shows that Zero Trust principles can be applied to this use case by relocating authentication, validation, trust evaluation, and policy enforcement to endpoint devices, while reducing the central server to a relay role and linking trust conditions to multiple disclosure outcomes. In this sense, the primary result of the thesis is a technically plausible and internally coherent architecture that answers the research question in a concrete and implementable way.

A first important result is that the proposed design reduces centralised trust assumptions in the proposed architecture when compared with conventional location-sharing architectures. In many existing systems, central backend infrastructure is trusted not only to relay messages but also to authenticate users, enforce authorisation logic, manage sessions, and process or store sensitive location data. In the proposed architecture, these trust-sensitive functions are redistributed. Authentication is performed locally on the requesting device, trust evaluation and policy enforcement are performed on the target device, and the central server is limited to encrypted message relay. This is a significant architectural change because it reduces the number of centrally trusted functions from several security-critical roles to a single communication role. The practical implication of this result is that compromise of the relay server no longer automatically implies compromise of policy decisions or plaintext location data. The design therefore, supports one of its core aims. The measured prototype timings suggest medium

processing overhead in the tested environment, although broader performance evaluation is required (see Table 6).

Table 6. Timing of on-device operations.

Device	**Operation**	**Mean time (ms)**	**Max time (ms)**
Samsung Galaxy S21 Android version 15	Create location request	982	1463
	Process request	463	714
	Process response	279	344
	Full request + response flow	1725	2259
OnePlus 6 Android version 15	Create location request	1704	2421
	Process request	942	1232
	Process response	450	540
	Full request + response flow	3097	3759
Nothing Phone 3 Android version 16	Create location request	1266	1840
	Process request	600	694
	Process response	301	341
	Full request + response flow	2168	2822

A second result is that the architecture supports more granular and adaptive access control than static or binary sharing models. Instead of treating access as a one-time allow-or-deny decision, the proposed design uses a trust score in the range of 0 to 100 and maps that score to four explicit disclosure outcomes: deny, semantic, approximate, and precise disclosure. This means that the design does not merely decide whether a requester may access location data, but also determines how much detail is justified under the current trust conditions. From the perspective of the research question, this architecture shows that location disclosure can be made adaptive rather than static, and that Zero Trust principles can be used not only as access verification, but also as controlled reduction of data sensitivity.

The results of the validation strategy further support the correctness of the proposed workflow. Workflow validation was intended to determine whether the prototype behaved according to the designed architecture, including the creation of requests, collection of trust-related inputs, protection of those inputs during transmission, validation on the target device, trust evaluation, policy enforcement, and return of the processed response. Within the scope of the prototype, the workflow logic showed that the architecture can be realised as an Android library and that its major components interact in the intended order. This is an important result because the contribution of the thesis is not merely conceptual. The work set out to define a reusable software library design for integration into existing Android applications, and the prototype setup shows that such integration is technically feasible in principle.

Access control validation was especially important for answering whether the thesis contribution works as intended. The proposed trust score model was designed to map different combinations of trust inputs to corresponding disclosure outcomes. In the validation design, representative trust scenarios were used to determine whether high-trust conditions produce precise disclosure, intermediate trust conditions reduce disclosure, and insufficient trust produces denial. Even though the thesis does not present a large empirical dataset, the resulting logic is coherent with the intended design. Requests from properly authenticated users on registered and uncompromised devices under plausible contextual conditions were expected to receive the least restrictive outcome, while requests involving suspicious conditions were expected to receive more restrictive outputs. This shows that the implemented scoring logic follows the defined policy mapping, although the policy thresholds themselves require further security justification. In other words, the results (see Table 7) indicate that the architecture is capable of enforcing proportional disclosure based on evaluated trust, which is one of the key contributions claimed in the Introduction and Contribution chapters.

Table 7. Trust Score Test Results.

Test case	**Bad Inputs**	**Expected score**	**Actual score**	**Expected policy**	**Actual policy**
Baseline	-	100	100	Precise	Precise
Rooted device	Rooted OS	90-100	93	Deny	Deny
Unknown requester	Unregistered device	90-100	93	Deny	Deny

Table 7. Trust Score Test Results.

Suspicious requests	Request frequency, timestamp	80-100	86	Deny	Deny
Signing certificate hash mismatch	Application signing certificate hash	90-100	94	Deny	Deny
Device was previously registered but later removed	Unregistered device	90-100	93	Deny	Deny
Replay attack	Timestamp	90-100	93	Deny	Deny
Weak device security	OS version, permissions	80-90	80	Approximate	Approximate
Late evening request	Timestamp	90-100	93	Precise	Precise
Nighttime and public Wi-Fi	Timestamp, network	80-90	86	Approximate	Approximate
Impossible travel	Geolocation, Frequency of context change	80-90	80	Deny	Deny
Request from nearby but unusual country	Geolocation	90-100	93	Precise	Precise
Various inputs 1	OS version, permissions, network	70-80	74	Semantic	Semantic
Everything wrong	All inputs	0	0	Deny	Deny

In hard-fail cases, such as rooted OS or denial-triggering behavioural anomalies, denial overrides the numeric trust-score result.

Another important result concerns continuous re-evaluation. A defining feature of Zero Trust is that trust should not be established once and then assumed to remain valid. The prototype testing design therefore examined scenarios in which trust-relevant conditions change over time and new requests must be re-evaluated. The expected behaviour was that the system would reduce or

revoke access when trust decreases (see Table 8), and relax restrictions only when sufficient trust conditions are satisfied. Within the scope of the proposed architecture, this behaviour is supported by the fact that each request is evaluated independently rather than inheriting trust from previous sessions. Although this work did not implement persistent sessions or cross-session comparison, this is still a meaningful result because it shows that the thesis incorporates one of the core principles of Zero Trust into the workflow.

Table 8. Single-Factor Degradation Test Results.

Changed input	**Score difference**	**Expected result**	**Actual result**
OS version	-9	Score downgrade	Score downgrade
OS rooted	-7	Deny	Deny
App version	-7	Score downgrade	Score downgrade
Time	-6	Score downgrade	Score downgrade
Geolocation	-8	Score downgrade or deny	Score downgrade
Network	-5	Score downgrade	Score downgrade
Request frequency	-10	Score downgrade or deny	Deny
Trust recency	-7	Score downgrade	Score downgrade
Permissions	-8	Score downgrade	Score downgrade

In cases where the after-score remains within the same disclosure range, the result is reported as a score downgrade rather than a policy downgrade.

At the same time, the thesis does not show that the proposed architecture is universally secure, nor does it empirically prove that the selected trust weights and thresholds are optimal. The validation is explicitly limited to a proof-of-concept implementation in a controlled Android environment, and the trust score model is based on reasoned design choices rather than calibration through large-scale empirical observations. This means that the results validate technical plausibility and architectural coherence, but they do not yet establish production readiness or generalisable quantitative performance claims. Similarly, the absence of a formal usability study means that the thesis cannot fully assess how end users would perceive the

security mechanisms in daily use, even though usability was considered at the level of design constraints. These limitations are important because they demonstrate that the proposed approach is feasible and aligned with its objectives, but it does not yet prove that it would outperform all alternative implementations.

From a critical perspective, the strongest result of this thesis is the change in trust placement. The biggest fundamental contribution lies in the architectural redesign itself. Even if future work changed the exact point values, thresholds, or attribute weighting logic, the main result would remain that location sharing can be organised so that authentication, validation, and data transformation occur at the endpoint level rather than centrally. Overall, the results show that the contribution is internally coherent and technically plausible within the defined prototype scope, as shown in Table 9. The proposed architecture satisfies the central design objective of reducing reliance on central trust, supports adaptive disclosure of location data through a policy-driven trust score model, and can be realised as an Android library.

Table 9. Validation Summary.

Objective	**Validation basis**	**Result**
Reduced reliance on central trust	Attack surface reduction analysis	The server is reduced to a relay role.
Zero Trust logic at endpoints	Workflow validation	All core logic functions are performed at endpoint-level.
Adaptive disclosure of data	Access control validation	The model supports four levels of data disclosure.
Trust as a continuous variable	Policy evaluation	Trust used as new input for policy decisions for each request.
Protected transmission and storage of data	Threat model	Attributes and responses are signed, encrypted, timestamped, and validated before use.
Implementable Android library	Codebase	Prototype Android library implementing the proposed workflow.
Reduced damage from interception/server compromise	Threat model	Confidentiality and integrity are protected against relay/server compromise under the stated cryptographic assumptions; availability remains affected.

6.2. Practical deployment considerations

Although the proposed architecture is technically feasible, its real-world deployment requires consideration of several practical constraints. One important factor is device diversity, as Android devices differ significantly in operating system version, hardware-backed security support, and background execution policies. These differences affect how consistently the proposed mechanisms, such as secure key storage and device integrity checks, can be implemented across devices. As a result, a production deployment would need fallback logic for devices that do not support the same security features at the same level. Another important consideration is the application integration effort. While the proposed solution is designed as a reusable Android library, existing mobile applications may differ in architecture, networking, and authentication logic. Integrating the library into production systems would therefore require careful adaptation to the host application's structure, especially in relation to permission handling and device registration. For example, if device registration is performed online, either the library design or the surrounding application logic needs to be adapted in order for key exchange to be secure. A further consideration is usability. Security mechanisms such as device registration and trust-based access restrictions improve the security posture, but they can also cause frustration for end users. For example, initial device pairing, when done offline, must be completed twice - once per device - to allow for secure key exchange. However, this takes twice as long and can therefore be a nuisance. Further, requests may be denied or downgraded under changing conditions that users do not immediately understand, which could reduce trust in the application. Therefore, the best practice would be for this to be implemented as transparently as possible. The proposed architecture also introduces operational dependencies. Even though the central server is not trusted for policy evaluation or plaintext data access, it still remains necessary for routing requests and responses between devices. This means that service availability, message delivery reliability, and device reachability are still important concerns. Mobile devices may be offline, background-restricted, or unreachable due to network conditions, which can affect on-time request and response delivery. In a production environment, this would therefore require mechanisms such as message queueing.

6.3. Comparison with the state of the art

Compared with the state of the art presented in Chapter 2, the proposed contribution of this thesis is the design of a concrete Zero Trust-based architecture specifically for consumer-oriented mobile location sharing. Existing work has primarily focused on separate mechanisms such as location obfuscation, encryption, fine-grained sharing policies, or decentralised access control models. In contrast, the proposed design brings these strands together into a single architecture in which every privacy and security preserving functionality is performed on the endpoint devices.

Additionally, this thesis improves on the way in which location disclosure is tied to trust evaluation. Prior work has proposed approximate, semantic, or policy-based location sharing, but these techniques have been treated as standalone privacy features. In the proposed architecture, the level of disclosure is not determined only by a predefined sharing preference, but by a continuously evaluated trust score that combines multiple indicators. This creates a more explicit link between current risk and the amount of location data revealed, allowing the same system to produce precise, approximate, semantic, or denied responses depending on trust conditions. In this sense, the thesis extends prior work by showing how granular location exposure can be integrated into a Zero Trust decision model rather than treated as a separate privacy mechanism.

The thesis also differs from prior work in its implementation perspective. Several related studies remain conceptual or focused on specific isolated controls, whereas this thesis frames its contribution as a reusable Android library intended for integrating into existing applications. This makes the work more directly oriented toward practical adoption in consumer mobile device software. The prototype design, module breakdown, and validation strategy therefore contribute not only a conceptual architecture but also a path for implementing that architecture in a realistic development setting. From this perspective, the work provides an implementation-oriented contribution relative to the reviewed literature by translating high-level Zero Trust principles into a mobile application integration model that is concrete enough to be built and tested.

At the same time, the proposed design does not improve every aspect of the topic. It does not fully solve endpoint compromise. If the requesting device is compromised, location data may still be exposed after decryption, and if the target device is compromised, local trust inputs or

enforcement logic may be manipulated. Similarly, the architecture does not eliminate indirect inference through side information, since an authorised application may still derive behavioural information from repeated requests or auxiliary device signals. In addition, the trust score model used in the thesis is a practical prototype mechanism based on reasoned design choices rather than empirical calibration. For these reasons, the contribution of this thesis should be understood as an architectural and implementation-oriented advance over the state of the art, rather than a complete solution to all remaining privacy and security challenges.

6.4. Broader implications

The proposed architecture has broader implications beyond its technical contribution, particularly in the context of consumer mobile applications, privacy regulations, and legal accountability. For consumer applications, the adoption of Zero Trust-based location sharing could reduce the reliance on broad and persistent trust assumptions that are common in current mobile ecosystems. In practical terms, this means that applications aimed at everyday users could expose less precise information by default and make location disclosure more adaptive to device and contextual risk. This may improve user trust in such applications, especially in communities where users are increasingly concerned about continuous tracking and centralised data collection.

From a privacy and regulatory perspective, the proposed design is consistent with technical design goals associated with data minimisation and confidentiality, although legal compliance would require separate assessment. Under the GDPR, location data qualifies as personal data because it can identify or help identify a person, directly or indirectly, and processing must therefore comply with principles such as lawfulness, data minimisation, purpose limitation, and integrity and confidentiality [28]. The design also reflects the principle of data protection by design and by default, since precise location data is retained on the target device unless a policy decision explicitly permits broader disclosure [28]. Additionally, EU guidance has treated geolocation data as requiring particular care, especially where it is processed through communications services or can reveal highly sensitive behavioural patterns [29].

The legal implications are also significant. A system that limits centralised access to plaintext location data may reduce the consequences of backend compromise, but it does not remove the legal responsibilities associated with processing location data. The operator of such a system would still need a valid legal basis for processing that data and the safeguards that protect it [28]. Moreover, reducing direct disclosure does not eliminate downstream legal risk if authorised applications or users derive additional information from repeated requests or side information. For this reason, the broader implication of the proposed architecture is not that compliance becomes automatic, but that technical design choices can support privacy-preserving and more accountable design practices. In that sense, the architecture shows how Zero Trust principles can contribute not only to stronger security but also to more accountable consumer-facing handling of location data.

6.5. Future work

Several directions for future work follow naturally from the limitations and design decisions of this thesis. First, the proposed architecture should be extended from prototype-level validation to a more complete implementation and evaluated in a realistic multi-user deployment. Such a study would make it possible to assess long-term maintainability, behaviour under varying network conditions, and the scalability of trust evaluation when multiple devices and users interact concurrently. It would also allow the trust score model to be calibrated more empirically rather than relying only on reasoned design choices. In particular, future work should examine in more detail which components of the trust score model should contribute how much to the overall trust score.

Second, the improvement of endpoint trust assessment should be considered. As discussed in this thesis, the proposed design still depends heavily on the integrity of the participating devices. Future work could therefore focus on stronger device attestation mechanisms, improved detection of rooted or otherwise compromised environments, and more robust validation of difficult attributes such as spoofed location. Related to this, the current approach to continuous trust re-evaluation could be expanded with richer behavioural analysis, allowing the system to detect suspicious request patterns or environmental inconsistencies more accurately. This would

also strengthen the empirical basis for deciding how much behavioural and contextual evidence should influence the final access decision.

Another relevant direction concerns cryptographic key management and trust establishment between devices. In the proposed design, secure device registration and key exchange are simplified to keep the system implementable within the scope of the thesis. A production-ready system should require more complete lifecycle support, including key rotation, device revocation, and recovery after device loss. Future work could explore how secure and usable trust establishment can be achieved at scale without reintroducing reliance on centralised infrastructure.

A further research direction concerns the trade-off between availability and privacy when devices are offline. In the current design, keeping location data primarily on the target device improves confidentiality and reduces centralised trust, but it also creates a practical dilemma: if the target device is offline, the requester may receive no location data at all. Future work could explore solutions where location data is obtainable from a short-term cache of stale data, without exposing that data on the central server.

Finally, the implementation scope could be broadened beyond the current Android-focused prototype. Cross-platform support, particularly for other major mobile operating systems like iOS, would help determine which aspects of the architecture are generally portable and which depend strongly on platform-specific security primitives. In parallel, future work could evaluate how the proposed library model integrates with existing commercial mobile applications, whether developers can adopt it with minimal effort, and how users perceive the resulting balance between privacy, transparency, and usability.

7. Conclusion

The objective of this thesis was to examine how Zero Trust principles can be applied to Android-based location sharing and to design a reusable software library that supports this approach in practice. The research question was how Android-based location sharing can be redesigned using Zero Trust principles to reduce reliance on centralised trust and support adaptive disclosure of location data.

The thesis showed that this can be approached at the prototype level by relocating authentication, validation, trust evaluation, and policy enforcement to endpoint devices, while limiting the central server to the role of an encrypted message relay. In the proposed design, access to location data is evaluated separately for each request on the basis of device-related, contextual, and behavioural inputs. The resulting trust score is then used to determine the level of disclosure, ranging from denial to semantic, approximate, or precise location sharing.

The main result of the thesis is that the proposed architecture reduces reliance on centralised trust compared with conventional server-centric location-sharing models. The server is no longer responsible for trust-sensitive decision-making and does not need access to plaintext location data before policy enforcement. In addition, the architecture can be realised as an Android library, which supports its intended use as an integratable software component rather than only as a conceptual model. The proof-of-concept validation showed that the proposed workflow can be implemented as designed and that the trust-based disclosure model is technically plausible and architecturally coherent within the scope of the prototype.

At the same time, the thesis has clear limitations. The evaluation was conducted at a proof-of-concept level rather than through real-world deployment, the trust score model is based on reasoned prototype defaults rather than empirical calibration, and no formal usability study was carried out. Endpoint compromise also remains a residual risk that cannot be fully eliminated by the proposed design. Despite these limitations, the thesis answers the research question positively and contributes a Zero Trust-inspired architecture, a reusable Android library prototype, and a validation of its feasibility.

References

[1] Liu J., Wolfson O., Yin H. Extracting Semantic Location from Outdoor Positioning Systems. 2006. 7th International Conference on Mobile Data Management, p. 73. https://doi.org/10.1109/MDM.2006.87

[2] Merry K., Bettinger P. Smartphone GPS Accuracy Study in an Urban Environment. 2019. PLOS ONE, vol. 14, no. 7. https://doi.org/10.1371/journal.pone.0219890

[3] Barnes S. Location-Based Services: The State of the Art. 2003. e-Service Journal, vol. 2, no. 3, pp. 59-70. https://doi.org/10.1353/esj.2004.0001

[4] Ilarri S., Illarramendi A., Mena E., Sheth A. Semantics in Location-Based Services. 2011. IEEE Internet Computing, vol. 15, no. 6, pp. 10-14. https://doi.ieeecomputersociety.org/10.1109/MIC.2011.156

[5] Richter D., Winter S., Richter K.-F., Stirling L. Granularity of Locations Referred to by Place Descriptions. 2013. Computers, Environments and Urban Systems, vol.41, pp. 88-99. https://doi.org/10.1016/j.compenvurbsys.2013.03.005

[6] Kawabata H., Isohara T., Takemori K., Kubota A. SanAdBox: Sandboxing third party advertising libraries in a mobile application. 2013. 2013 IEEE International Conference on Communications, pp. 2150-2154. https://doi.org/10.1109/ICC.2013.6654845

[7] Bugiel S., Davi L., Dmitrienko A., Fischer T., Sadeghi A.-R., Shastry B. Towards Taming Privilege-Escalation Attacks on Android. 2012. NDSS, vol. 17, pp. 19. https://download.hrz.tu-darmstadt.de/media/FB20/Dekanat/Publikationen/TRUST/NDSS_2012_Towards_Taming_Privilege-Escalation_Attacks_on_Android.pdf

[8] Felt A. P., Ha E., Egelman S., Haney A., Chin E., Wagner D. Android Permissions: User Attention, Comprehension, and Behaviour. 2012. Proceedings of the Eighth Symposium on Usable Privacy and Security, pp. 1-14. https://doi.org/10.1145/2335356.233536

[9] Razaghpanah A., Nikai A. A., Vallina-Rodriguez N., Sundaresan S., Amann J., Gill P. Studying TLS Usage in Android Apps. 2017. Proceedings of the 13th International Conference

on Emerging Networking Experiments and Technologies, pp. 350-362. https://doi.org/10.1145/3143361.3143400

[10] Android Developers. Network Security Configuration. 2025. https://developer.android.com/privacy-and-security/security-config. Accessed: 12.03.2026.

[11] Candia T. Best Practices for Deploying Zero Trust in Your Mobile Environment. 2023. BizTech Magazine. https://biztechmagazine.com/article/2023/02/best-practices-deploying-zero-trust-your-mobile-environment

[12] Rose S., Borchert O., Mitchell S., Connelly S. Zero Trust Architecture. 2020. NIST Special Publication 800-207. https://doi.org/10.6028/NIST.SP.800-207

[13] Yeoh W., Liu M., Shore M., Jiang F. Zero trust cybersecurity: Critical success factors and A maturity assessment framework. 2023. Computers and Security, vol. 133. https://doi.org/10.1016/j.cose.2023.103412

[14] Syed N. F., Shah S. W., Shaghaghi A., Anwar A., Baig Z., Doss R. Zero Trust Architecture (ZTA): A Comprehensive Survey. 2022. Computer Standards & Interfaces, vol. 89. https://doi.org/10.1109/ACCESS.2022.3174679

[15] Kachore V. A., Lakshmi J., Nandy S. K. Location Obfuscation for Location Data Privacy. 2015. 2015 IEEE World Congress on Services, pp. 213-220. https://doi.org/10.1109/SERVICES.2015.39

[16] Partovi A., Zheng W., Jung T., Lin H. Ensuring Privacy in Location-Based Services: A Model-based Approach. 2020. https://doi.org/10.48550/arXiv.2002.10055

[17] Safi M. I., Patil S., Page X. Predicting smartphone location-sharing decisions through self-reflection on past privacy behavior. 2020. Journal of Cybersecurity, vol. 6, issue 1. https://doi.org/10.1093/cybsec/tyaa014

[18] Shahraki A. S., Rudolph C., Alavizadeh H., Kayes A.S.M., Rahayu W., Tari Z. Securing cross-domain data access with decentralised attribute-based access control. 2025. Ad Hoc Networks, vol. 173. https://doi.org/10.1016/j.adhoc.2025.103807

[19] Smari W. W., Clemente P., Lalande J.-F. An extended attribute based access control model with trust and privacy: Application to a collaborative crisis management system. 2014. Future Generation Computer Systems, vol. 31. https://doi.org/10.1016/j.future.2013.05.010

[20] Peebles R. decentralised Location Sharing Using Blockchain. 2025. University of Victoria. https://hdl.handle.net/1828/22558

[21] Shafeeq S., Alam M., Khan A. Privacy Aware decentralised Access Control System. 2019. Future Generation Computer Systems, vol. 101. https://doi.org/10.1016/j.future.2019.06.025

[22] Adahman Z., Malik A. W., Anwar Z. An Analysis of Zero-Trust Architecture and its Cost-Effectiveness for Organizational Security. 2022. Computers & Security, vol. 122. https://doi.org/10.1016/j.cose.2022.102911

[23] Tabalipa A. Bridging the Mobile Trust Gap: A Zero Trust Framework for Consumer-Facing Applications. 2025. Cornell University. https://doi.org/10.48550/arXiv.2508.16662

[24] Adapa V. R. K. Zero Trust Architecture Implementation in Critical Infrastructure: A Framework for Resilient Enterprise Security. 2024. International Journal of Advanced Research in Engineering and Technology, vol. 15-6, pp. 76-89. https://doi.org/10.34218/IJARET_15_06_006

[25] Fernandes E., Aluri A., Crowell A., Prakash A. Decomposable Trust for Android Applications. 2015. 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, pp. 343-354. https://doi.org/10.1109/DSN.2015.15

[26] Qazi F. A. Insecure Application Programming Interfaces (APIs) in Zero-Trust Networks. 2021. Capitol Technology University. https://www.proquest.com/openview/93d809dffbd1a584b18ee126cd81d4b9/1.pdf?pq-origsite=gscholar&cbl=18750&diss=y

[27] Nazir A., Iqbal Z., Muhammad Z. ZTA: A Novel Zero Trust Framework for Detection and Prevention of Malicious Android Applications. 2025. Wireless Networks, vol. 31, pp. 3187-3203. https://doi.org/10.21203/rs.3.rs-4464369/v1

[28] European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal

data and on the free movement of such data (General Data Protection Regulation). 2016. Official Journal of the European Union, L 119, pp. 1–88. https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679

[29] European Data Protection Board. Guidelines 04/2020 on the use of location data and contact tracing tools in the context of the COVID-19 outbreak. 2020. https://www.edpb.europa.eu/sites/default/files/files/file1/edpb_guidelines_20200420_contact_tracing_covid_with_annex_en.pdf

Appendix 1 - Non-Exclusive License for Reproduction and Publication of a Graduation Thesis[8]

I, Villem Susi

1. grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis "Secure Location Sharing on Android Devices Using Zero Trust Principles", supervised by Matthew James Sorell, PhD

    1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;

    1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.

2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.

3. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

[8] The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.