ICMS: A Flexible Location-Based Access Control System for Mobile Devices

Ahmed Khalil Abdulla, Spiridon Bakiras, and James She

***Abstract*—This paper introduces *I Control My Space* (ICMS), a novel, fine-grained, and flexible system that allows space owners to define and enforce access restriction and service policies on mobile devices at their premises. This is achieved by enforcing restrictions on the use of applications and even specific permissions on the mobile device (e.g., camera, microphone, etc.). The system is built around a centralized database, where space owners can securely define their restrictions or even delegate control of their space to another owner. ICMS is designed with abstraction in mind, so that it can be customized on existing access technologies and platforms—like Android-based mobile devices—for various levels of security and privacy. A proof-of-concept prototype of ICMS is developed under the Android operating system, and promising results of its performance and effectiveness are demonstrated.**

***Index Terms*—Location-based access control, space-based access control, access control, location privacy**

I. Introduction

MOBILE devices are pervasive in our daily activities today. This is, in part, due to the vast number of available mobile apps that add extra functionality to mobile devices, ranging from entertainment-focused to business-oriented applications [1]. The inherent mobility of smart devices facilitates their uninterrupted usage, even as users roam inside different physical spaces (e.g., buildings, museums, etc.). Almost all smart devices today are equipped with various built-in modules (e.g., camera, microphone), however, the use of such modules or specific applications is not always welcomed or allowed by the physical space owner. Therefore, controlling the access of these devices to services and applications by the space owner has become a necessity. For instance, a government entity might want to disable the camera module for all mobile devices, in order to prevent photos from being taken in highly sensitive offices within its building.

Due to the demand for controlling access based on user location, we find many contributions in the literature that address location-aware access control models and systems. However, most of these efforts offer the access control enforcement privileges to the service providers (e.g., application developers) rather than the physical space owners. Some approaches do provide access control decisions to physical space owners [2]–[4], but they fail to propose a comprehensive solution that addresses the technical challenges associated with space owners being able to configure access restrictions within their own spaces.

In this paper, we introduce *I Control My Space* (ICMS), a flexible and scalable system that enables physical space owners to enforce fine-grained access restrictions on mobile devices that navigate within their spaces. ICMS employs a central server that acts as a root space authority for a large area (e.g., a country) and stores all the details related to physical spaces and access policy configurations. The client-side of ICMS is running on the end-user's mobile device. It is responsible for obtaining space and restrictions details from the central server and enforcing them on the mobile device's applications and services, according to the end-user's current location.

An important feature of our system is that it offers the ability to delegate control of a set of spaces to another authority. This level of flexibility enables the enforcement of different sets of restrictions on different spaces, if required, and reduces significantly the overhead of a space authority that manages a large number of physical spaces. Moreover, ICMS supports the definition of granular access policies that restrict specific permissions on specific applications. In addition, ICMS operates in a seamless and transparent way, because it does not require any particular interaction with the end-user. Finally, our system preserves the privacy of the user's location information, by making such data visible only at the client-side. The contributions of this paper can be summarized as follows:

- We propose a novel location-based access control system—I Control My Space (ICMS)—that allows physical space owners to setup access restriction policies on mobile devices attending their premises.
- We implement a proof-of-concept prototype using a cloud platform and customized Android-based smartphones.
- We conduct a comprehensive set of empirical evaluations that demonstrate ICMS's feasibility and its negligible impact on the device's performance.

The rest of the paper is organized as follows. Section II discusses the related work and Section III details the proposed system design. Section IV introduces our proof-of-concept implementation and Section V reports the results of our experimental evaluations for various performance metrics. Section VI discusses various aspects of ICMS and highlights future research directions. Section VII concludes our work.

II. Related Work

T. Chowdary et al. [2] proposed a policy-based mobile application that enforces security and restricts access to applications and data on Android devices, based on the user's location and profile. The policy decision enforcement is achieved by affecting several core services components within the Android

middleware. The access profiles for users are predefined by the system administrator. The system supports two different categories of users: regular users and specified users. The desired outcome of their work is to limit the students' use of the campus WiFi to educational only apps, while having another profile for faculty within a university campus. However, the proposed solution can only enforce access restrictions when the user is connected to a WiFi infrastructure controlled by the space owner.

R. Franziska et al. [3] proposed an extensible world-driven access control framework that aims to tackle issues that confront device users and bystanders in the emerging continuous sensing applications. Such applications require continuous monitoring of multiple data sensors, such as video and audio, in order to deliver their functionality. The framework design allows a real-world object owner to define access policies on objects, in order to prevent untrusted applications from gaining access to such data (e.g., access to camera stream while in a locker room). Still, it requires tagging real-world objects with policy definitions (using QR codes), which is not a practical and scalable concept.

E. Carlos et al. [4] proposed a theoretical space-sensitive access control model that restricts the space-sensitive functionality of mobile applications, based on (i) the user's location; and (ii) an access control policy enforced by either the physical space owners or the application developers. Their proposed model necessitates the application developer to include the required logic that queries the user's location and sends an authorization request to a central server where all policies are stored. Nevertheless, the authors did not focus on addressing other challenges, such as space ownership registration, revocation, delegation, and protection of user location information.

Several efforts have focused on location-based access control (LBAC) models and systems that mainly utilize a location-based constraint to allow service providers and end-users to control access to certain resources they own [2]–[16].

Some efforts presented their models as extensions to existing role-based access control (RBAC) models. This was achieved with the addition of location-based attributes, used to enable certain roles in specific locations [9], [15], [16]. In addition, other researchers have presented the use of spatial attributes to tackle a specific case of access control, such as the user authentication process [11], [17].

A lot of work has explicitly focused on the introduction of proximity-based constraints in LBAC models and systems, which takes into consideration the locations of other users in the system relative to the location of the access requester [12], [13], [16].

Moreover, some researchers focused on addressing the challenge of protecting the privacy of user location information in the context of LBAC [17]–[20], while others focused on addressing accuracy limitations in localization technologies and verifying the integrity of the reported location information [10], [15].

Finally, some efforts have addressed different aspects within the context of LBAC, such as: the ability of an adversary to fake their reported location information [21]; proposed evaluation criteria to evaluate LBAC models [22]; the study of further requirements of LBAC models [23]; and suggesting future research directions in the field of LBAC [24].

Table I depicts a comparison of ICMS against similar LBAC systems. The superiority of ICMS is evident in terms of application logic modifications requirement, user location privacy, and limitations related to physical spaces.

III. The Proposed System — I Control My Space (ICMS)

A. Design Goals

ICMS is designed with the following goals in mind:

- **Priority-based access control enforcement:** Designing a model that offers prioritized enforcement of access control decisions is an essential goal of our system. Multiple levels of access enforcement are provided to all involved parties: space owner, end-user, and application developer. At the highest level, a space owner is offered with the ultimate decision to restrict any application or feature on any user smartphone that operates at their own locations. The next enforcement level is assigned to the end-users, so that they can decide what applications and features are usable on their mobile devices. The lowest level is assigned to the application developers, who should not be able to bypass restrictions applied by space owners or end-users on their applications.
- **Scalable hierarchical structure of space authorities:** The proposed system should support a hierarchical organization of space authorities, in order to improve the system's scalability. Under a hierarchical structure, a space authority (e.g., a governmental entity) is able to delegate sub-spaces under its ownership to other space authorities (e.g., organizations, companies, individuals). Such delegation enables the delegated authority to manage the access restrictions associated with the delegated sub-space(s).
- **Space ownership management:** Due to the planned hierarchical design structure of the space authorities, our system will support flexible space ownership management operations that include ownership registration, delegation, and revocation.
- **Fine-grained access control:** Space authorities should not be limited in the way they define the access restriction policies within their controlled areas. Rather, the proposed model should allow for deactivating specific permissions for specific applications (e.g., restrict camera access on the WhatsApp application). This facilitates the definition of more granular access restriction policies.
- **Transparency of decisions:** We believe that the end-user should have full transparent knowledge of the access restriction policies that are currently being applied on their mobile device by a certain space authority, while navigating around different space boundaries.
- **Seamless operations:** Fully automate the system's operation from the end-user's perspective, by eliminating the need for any kind of interaction with the end-user. This is to ensure that the space access policies are being enforced

TABLE I: Comparison of ICMS against existing LBAC systems

System	Requires applications modification	Requires OS modification	Preserves user location privacy	Limitations related to physical spaces	Security bypass possibilities
T. Chowdary et al. [2]	No	Yes	No	Restrictions only enforced while the user is connected to a specific WiFi infrastructure.	(1) Device rooting (*difficult*); (2) User connects to cellular network rather than WiFi (*easy*).
R. Franziska et al. [3]	No	Yes	Yes	Real-world objects must be tagged (e.g., using QR codes).	(1) Device rooting (*difficult*); (2) Misplacement of real-world object tags (*easy*).
E. Carlos et al. [4]	Yes	No	No	No limitations	(1) Uncooperative developer who did not include the required logic into the application source code (*easy*); (2) Location spoofing (*difficult*).
ICMS	No	Yes	Yes	No limitations	(1) Device rooting (*difficult*); (2) Location spoofing (*difficult*).

on the end-user's device, without affecting its usability or introducing any change to the normal user experience.
- **Location privacy:** The system should correctly enforce the location-based access control policies without sharing or disclosing the user's location to any other party.

B. Threat Model

The threat model defines the adversaries and the possible attacks that can be performed to compromise the system's access control policies. Our goal is to prevent the manipulation of the defined access control policies and ensure their enforcement on end-user mobile devices. We present the following threat actors:

- **Malicious end-user:** A malicious end-user may try to circumvent the applied restrictions in order to activate the use of certain applications and permissions that are not allowed by the underlying space authority. In addition, the end-user may try to deceive the system by faking their current location information in order to obtain unauthorized access to restricted applications and permissions.
- **Curious/Malicious application developer:** A curious application developer may want to monitor or collect user location information (e.g., for analysis and marketing purposes). In addition, a malicious developer may try to circumvent the restrictions applied by either a space authority or an end-user, by having the application forcefully activating itself on the end-user's device or granting itself restricted permissions.
- **Curious space authority:** A curious space authority may try to collect information about user locations and/or build user-specific profiles in terms of app usage. Such attempts might compromise end-user privacy.

C. System Architecture

ICMS's architecture consists of three major components that interact with each other. We describe each component below.

1) **Trusted software module:** ICMS contains a software extension to the operating system of the mobile device that runs and executes within the kernel-level as a privileged service (a.k.a. system service). We call this portion of the software the *Trusted Software Module* (*TSM*). The *TSM* runs on the end-user's mobile device and acts as the trusted part of the client-side portion of the system. The two main functions of the *TSM* are: (i) to obtain the end-user's current location; and (ii) to enforce the required access restrictions on the device's applications and permissions.
2) **Space manager server:** The *Space Manager Server* (*SMS*) is a central server managed by a root space authority (e.g., a governmental entity that owns a large physical space). The server holds a database with details about all registered space authorities, the exact spaces they control, and the space-related delegation and restriction information. Furthermore, it offers a set of application programming interfaces (APIs) that allow space authorities to interact with their associated data stored at the central database (e.g., to update delegation and restriction details). Additionally, the *TSM* retrieves database records and updates by communicating directly with the *SMS*.
3) **Space authority:** A *space authority* (*SA*) is essentially a space owner of one or more physical spaces. Note that, ownership of a particular space can be delegated to a space authority by its original space owner, or by another intermediate delegated authority. The space authority can be an organizational entity, a company, or even a home owner.

In addition to the aforementioned components, our system leverages the services of two external parties: (i) a certificate authority (*CA*); and (ii) a location service provider (*LSP*). For authentication purposes, ICMS relies on the existing public key infrastructure (PKI), so there is a requirement to communicate with certificate authorities. Furthermore, our system needs to communicate with a location service provider to obtain the user's location information, regardless of the locating technologies and services adopted by the *LSP* (e.g., GPS, cellular, or WiFi infrastructure). Fig. 1 illustrates ICMS's components, their operations, and interactions with each other.

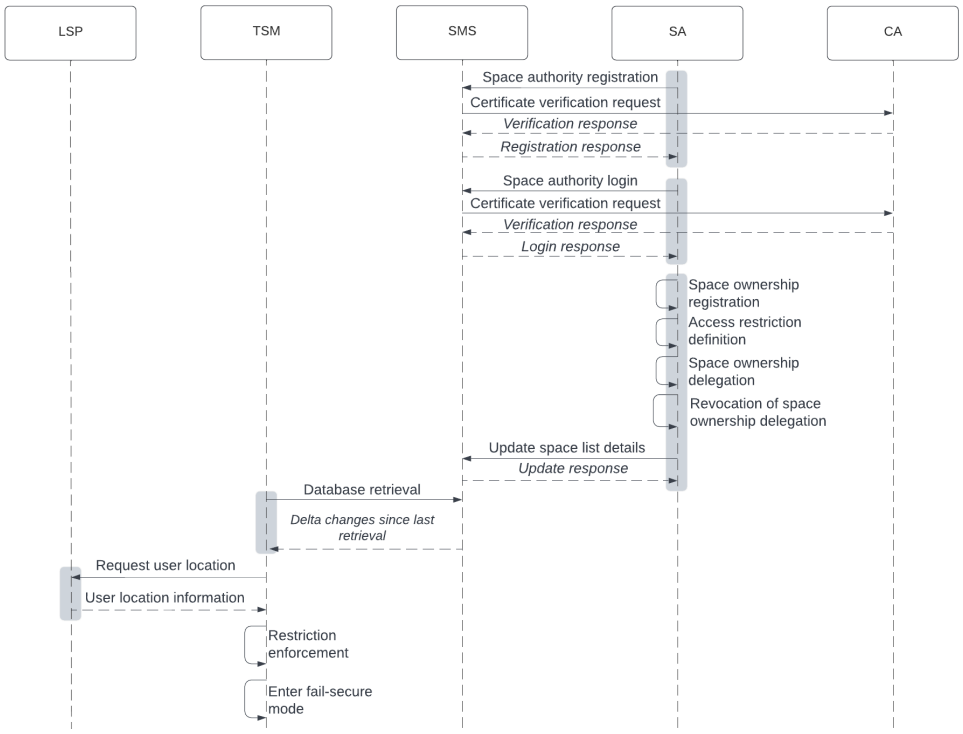


Fig. 1: ICMS system's components and their interactions

D. System Communication Paths

In this section, we highlight the communication paths present among our system components and their characteristics. The major communication paths are (i) the path between the *TSM* (client-side) and the *SMS* (server-side); and (ii) the path between the space authority and the *SMS*. Communications in both channels are performed in a simple request–response fashion.

Verifiable Requests. Any request sent by a space authority to the central server (*SMS*) is digitally signed by the requester. This allows the *SMS* to verify the request before performing any kind of processing associated with that request.

Update Time-stamping. When the *SMS* receives a request from a space authority that requires an update to the central database record associated with the requester space authority, the *SMS* attaches a *timestamp* to that record which marks its latest modification time. Using this timestamp, the *TSM* can further optimize the performance (time and power) when retrieving the database updates, by only requesting records that have been updated after the last data retrieval process.

For completeness, we want to stress that all the interactions between the end-user, the mobile device, and the *TSM* take place within the device itself. On the other hand, the communication between the *TSM*/*SMS* and the *CA*, as well as the communication between the space authorities and the *SMS* are all encrypted in transit through a secure TLS channel.

E. Operations

ICMS supports numerous operations that involve interactions between its components, as highlighted in Fig. 1. We divide them into different categories and discuss them in separate sections below. For the sake of clarity, we summarize all notations used in Table II.

1) Space authority registration and login: In this section, we discuss the registration and login operations that require interaction between a space authority (*SA*) and the space manager server (*SMS*).

Space authority registration: The main purpose of this operation is to register and create a record for a new *SA* in the central database. It is performed once by a space authority when it first registers itself with the *SMS*. The operation requires that the space authority holds a valid public key certificate issued by a trusted *CA*. The actions performed during the registration process are listed below:

1) The *SA* sends a registration request message $m = \{\text{“}register\text{”}, SA_{ID}, t\}$ to the *SMS*, where t is a timestamp that corresponds to the current time. It also sends the digital signature $s = Sign(SA_{SK}, m)$ of message m, signed with *SA*'s private key, as well as its public key certificate ($SA_{\mathcal{C}}$).
2) Once the *SMS* receives the request, it verifies the validity of the public key certificate $SA_{\mathcal{C}}$. If validated successfully, the *SMS* uses key SA_{PK} from $SA_{\mathcal{C}}$ to verify the digital signature associated with the request, i.e., $Verify(SA_{PK}, m, s)$.
3) After the successful verification of the signature, the *SMS* creates a record in its central database for *SA* and responds back with a *success* message.

Space authority login: This operation enables a space authority to login to the central server (*SMS*) and retrieve its associated data from the database. The operation involves the

TABLE II: System Design Notations

Term	Definition
APP	A mobile application with a unique identifier APP_{ID}.
SA	Space authority *SA* is a space owner of one or more physical space(s). It has a unique identifier, SA_{ID}.
SA_{PK}	Space authority *SA*'s public key.
SA_{SK}	Space authority *SA*'s private key.
$SA_{\mathcal{C}}$	Space authority *SA*'s public key certificate.
SP	Representation of a physical area. It has a unique identifier, SP_{ID}, and a defined boundary, $SP_{\mathcal{B}}$.
$SP_{\subset}$	A subset space of space *SP* ($SP_{\subset} \subset SP$).
$SP_{\mathcal{B}}$	A boundary of space *SP* is represented as a polygon shape. The polygon can have an arbitrary number of edges to allow a flexible representation of any physical space on a map.
SP_{SA}	Space ownership representation, where a space *SP* is either owned or delegated by other authorities to space authority *SA*.
SL_{SA}	A list of spaces that that are either owned or delegated by other authorities to space authority *SA*.
$\mathcal{P}$	Permission $\mathcal{P}$ that is given to an *APP* to allow it to access certain device services and features.
$\mathcal{R}$	A restriction policy record ($\mathcal{R} = \{\mathcal{P}, APP_{ID}\}$) that represents a restriction on an application with identifier APP_{ID} from being granted permission $\mathcal{P}$.
$SL_{SA}[SP].\mathcal{R}$	A list of restriction policy records that are enforced by space authority *SA* on a space *SP*.
$SP_{SA \rightarrow DA}$	A delegation record representation where space authority *SA* is delegating space *SP* to space authority *DA*.
$SL_{SA}[SP].\mathcal{D}$	A list of delegation records that correspond to space *SP*, which are delegated by space authority *SA*.
$Sign(k, m)$	A digital signature of message m signed by a private key k.
$Verify(k, m, s)$	A verification of signature s on message m, using public key k.
L_u	Representation of the current location information of user u. Location information includes latitude (lat) and longitude (lon) values ($L_u = \{lat, lon\}$).

following steps:

1) The *SA* sends a login request message $m = \{\text{“}login\text{”}, SA_{ID}, t\}$ to the *SMS*, where t is the current time. Similar to the registration operation, the *SA* also sends the digital signature $s = Sign(SA_{SK}, m)$ of m, and its public key certificate $SA_{\mathcal{C}}$.
2) The *SMS* first validates $SA_{\mathcal{C}}$ and, if successful, it verifies signature s, i.e., $Verify(SA_{PK}, m, s)$.
3) If the signature verification is successful (which guarantees freshness, due to timestamp t), the *SMS* sends back the data associated with the *SA* that are stored in the central database.

By offering this operation in our system, we allow the space authority to work in a stateless fashion, where it is not required to hold any kind of data locally, other than its own private key. In the event where a space authority needs to view its own data, register a space, delegate/revoke space ownership, or add/update access restrictions, it can perform the login operation to obtain the up-to-date information. It can then modify the underlying records accordingly, before submitting the updated data back to the *SMS*, as detailed in the following section.

2) Space authority data update: In this section, we discuss the operations that involve updating a space authority's data which are stored at the *SMS*.

Update space list details: This is the main operation available to a space authority, in order to update its space list details at the central server. The following actions are performed during this operation:

1) The *SA* digitally signs its space list details with its private key (SA_{SK}): $Sign(SA_{SK}, SL_{SA})$.
2) The *SA* sends the updated SL_{SA}, the associated digital signature, and its public key certificate to the *SMS*.
3) The *SMS* validates the received public key certificate before performing the signature verification. If the signature is verified successfully, the *SMS* modifies the *SA*'s database record with the newly updated space list details.

The space list contains most of the details related to the space authority, such as geographic locations, restriction policy records, and delegation records. Therefore, this operation is invoked after any operation or action that manipulates the space list data. Such operations include *space ownership registration*, *access restriction definition*, *space ownership delegation*, and *revocation of space ownership delegation*. All of them are discussed in the following paragraphs.

Space ownership registration: This operation assigns ownership of a space (*SP*) to a space authority. It involves the following actions:

1) The *SA* defines the outer boundary ($SP_{\mathcal{B}}$) of space *SP*. The boundary consists of a list of connected coordinates that forms a polygon shape on a map, and serves as a representation of the real-world boundary of the underlying space. Each coordinate in the list has the *longitude* and *latitude* values of a geographic location on the map. Moreover, the *SA* chooses an identifier SP_{ID} (e.g., a space name) to uniquely identify the space.
2) The *SA* adds the *SP*'s details into its space list SL_{SA}.
3) Finally, the *SA* performs the **update space list details** operation discussed earlier.

In a nutshell, the space ownership registration operation ensures that physical space ownership is correctly registered under the supervision of the *SMS*. Once the *SMS* receives the updated details that include new spaces to be registered, it performs a validation process (e.g., automated and/or manual processes) to ensure that the requesting space authority is allowed to own the claimed physical space.

Access restriction definition: This is a key operation that allows a space authority to define a restriction policy $\mathcal{R}$ that is enforced on users within a specific physical space *SP*. The operation consists of the following actions:

1) The *SA* constructs a restriction policy record $\mathcal{R}$, by choosing the application's unique identifier APP_{ID} and the permission $\mathcal{P}$ that should be restricted.
2) The constructed restriction record $\mathcal{R}$ is added to the list of restrictions associated with space *SP* that is under *SA*'s control ($SL_{SA}[SP].\mathcal{R}$).
3) Finally, the *SA* performs the **update space list details** operation discussed earlier.

Using this operation, a space authority can define fine-grained access restriction policies for different spaces under its control.

For instance, a space authority can decide to disable a certain device feature (e.g., camera) from being used by an *APP*, by utilizing permission $\mathcal{P} = CAMERA$ to form the following restriction record: $\mathcal{R} = \{CAMERA, APP_{ID}\}$. In order to assure greater flexibility in defining restrictions, we introduce the use of a special character " $*$ " as a value to either $\mathcal{P}$ or APP_{ID}. For example, a space authority can disable a specific application (*APP*) by setting the permission value to " $*$ " to form the following restriction record: $\mathcal{R} = \{*, APP_{ID}\}$. On the other hand, an authority can disable a specific permission (e.g., access to camera) from being granted to any application, by setting the APP_{ID} value to " $*$ " to form the following restriction record: $\mathcal{R} = \{CAMERA, *\}$.

Space ownership delegation: With this operation, a space authority *SA* can delegate control of an owned space *SP* to another space authority *DA*. It is worth noting that the delegated space can be either the entire space *SP* of an *SA*'s record in the database, or a sub-space $SP_{\subset}$ of it. The operation involves the following actions:

1) The *SA* defines the boundary of the space that needs to be delegated ($SP_{\mathcal{B}}$) and assigns a unique identifier for this space (SP_{ID}).
2) Once the space and its boundary are defined, the space details and the unique identifier of the *DA* are put together to construct a delegation record $SP_{SA \rightarrow DA}$.
3) After constructing the delegation record, the *SA* adds it to the list of delegation records associated with space *SP*: $SL_{SA}[SP].\mathcal{D}$.
4) The *SA* performs the **update space list details** operation discussed earlier.

After a space *SP* has been delegated to a new space authority *DA*, that authority can define new access restriction policies for *SP*. Furthermore, a space authority can divide a larger controlled space into smaller sub-spaces and delegate them individually to different authorities.

Revocation of space ownership delegation: By utilizing this operation, a space authority *SA* is able to revoke a delegation relationship ($SP_{SA \rightarrow DA}$) at any time. The operation performs the following straightforward actions:

1) The *SA* removes the delegation record $SP_{SA \rightarrow DA}$ from the delegation records list $SL_{SA}[SP].\mathcal{D}$.
2) The *SA* performs the **update space list details** operation discussed earlier.

This is a simple operation where a space authority can, at any time, revoke an existing delegation relationship. This is achieved by removing the associated record from its delegation list and then communicate the updated data to the *SMS*, in order to update its associated record in the central database.

3) Client-side operations: In this section, we discuss operations performed at the client-side (*TSM*) within the end-user's mobile device.

Database retrieval: This operation is invoked by the *TSM* periodically (based on a predefined interval), in order to retrieve all the updated records from the *SMS*'s database. It involves the following actions:

1) The *TSM* sends a request to the *SMS* along with a *timestamp* value that represents the largest timestamp among the previously retrieved records.
2) The *SMS* responds back with all records that have been updated since the supplied *timestamp*.
3) The *TSM* updates its local copy of the database with the new records. It also remembers the largest *timestamp* associated with these records for use in the next database retrieval process.

In summary, this operation ensures that the *TSM* stores the most up-to-date space restriction records, by periodically pulling the updates from the *SMS*. This is assured by the use of the *timestamp* value that allows the *TSM* to request the updates that are not reflected in its local database. For the initial database retrieval request, where the *TSM*'s database copy is empty, the *TSM* uses a *timestamp* value of 0 (zero) to retrieve all the records in the central database.

Our decision to maintain local copies of the centralized database at the mobile devices is based on the numerous advantages that it offers. First, if the *TSM* only requests the records around its current location, it discloses frequent location measurements to the *SMS*, which violates the user's privacy. On the other hand, a full database retrieval offers perfect location privacy. Second, it is possible that the mobile device does not have internet connectivity at certain areas. In this case, the *TSM* can use the existing (even if obsolete) database records to enforce the location-based restrictions. Finally, we do not expect that the centralized database would be updated very frequently, after the initial download. As such, the overhead on the mobile devices is minimal. Note that, if the size of the entire database is very large, the database retrieval operation can be implemented in a less aggressive way, where the *TSM* requests database records from a specific geographic area (e.g., 100km around the user's current location, or within a specific city or state). This can be determined based on the trade-off between user privacy and storage space.

Restriction enforcement: This operation is performed periodically by the *TSM*, using a fixed interval. It requires communication with a location service provider (*LSP*) to obtain the current location L_u of end-user u. The aim of this operation is to ensure that the correct location-based access restrictions are applied on u's mobile device, while present within a controlled space *SP*. The following actions are performed:

1) The *TSM* sends a request to the *LSP* to obtain u's current location L_u.
2) Once L_u is known, the *TSM* processes it against the local copy of the restrictions database, in order to obtain the record that corresponds to space authority *SA*, where $L_u \subset SP_{\mathcal{B}}$ and $SP \in SL_{SA}$.
3) From the retrieved record, the *TSM* extracts all the restriction policies $SL_{SA}[SP].\mathcal{R}$. Furthermore, if *SP* is delegated to another authority ($SP_{SA \rightarrow DA}$), the *TSM* obtains the record associated with the delegated authority *DA* and performs this step again to retrieve *DA*'s restriction policies.
4) The *TSM* stores the current state of granted permissions in the user's mobile device, so that it is able to revert

permissions back to the previous state, once u moves out of *SP* (e.g., $Lu \not\subset SP_{\mathcal{B}}$).

5) The *TSM* enforces the collected restrictions, by disabling the intended permissions and applications on u's mobile device.

In this operation, the *TSM* ensures that all the location-based restrictions defined by the responsible space authorities are being enforced on u's mobile device applications and services. In Step 1, the *LSP* may use any available localization technology (e.g., GPS, WiFi) to estimate the user's current location. Moreover, as mentioned previously, Step 2 preserves u's location privacy, by not sharing any location information with the central server. Finally, Step 3 allows the system to define an arbitrary chain of delegations for a specific space *SP*, where each space authority on the hierarchy can define its own restriction policies.

Enter fail-secure mode: This is an exceptional operation that is executed when ICMS is unable to operate as expected (e.g., not meeting the required preconditions). When the fail-secure mode is entered, the following actions are performed:

1) The *TSM* tries to forcefully meet the required preconditions (e.g., if GPS is a required service, the *TSM* will try to force-enable the GPS service on the device).
2) If the required preconditions are met, the *TSM* exits the fail-secure mode and continues its normal operation.
3) If any of the required preconditions cannot be met, the *TSM* performs the configured fail-secure mode behavior (e.g., disable access to all applications). This behavior can be configured and customized to match the required purpose during implementation.
4) The *TSM* continuously performs the same steps until the required preconditions are met.

If the system cannot function as expected (e.g., due to the user disabling some of the required services), the *TSM* enters a fail-secure mode that prevents security bypass attempts. For example, ICMS requires the following services: (i) localization service to geolocate the user; and (ii) internet connectivity to communicate with the *SMS*. In our implementation, we customized the fail-secure mode to ensure that the user is not allowed to obtain unauthorized access to applications and services, by disabling access to all applications. However, to offer a layer of abstraction between the system design and its implementation, we kept Step (3) very generic. Thus, this operation can be tailored during the implementation phase to achieve the required security assurance level, which is best suited for the intended environment.

IV. Implementation

A. Implementation Overview

We built a fully functional implementation of our proposed system. The implementation consists of three separate modules: (i) the *TSM* (client-side); (ii) the *SMS* (server-side); and (iii) the user interface for space authorities to communicate with the *SMS*. The source code from all modules is publicly available on GitHub [25].

We implemented the *TSM* as a system service within the Android operating system for the following reasons: (i) the Android operating system is open-source and its source code is publicly available for mobile manufacturers and developers to modify and customize; and (ii) it is one of the most popular operating systems for mobile devices. We introduce most of our customization on the Application Framework layer of the Android architecture [26]. The entire source code for the *TSM* is written in Java [27], the supported language for Android Application Framework development.

We implemented the *SMS* server with an asynchronous event-driven JavaScript runtime, Node.js [28]. Node.js offers a non-blocking behavior, which allows scalability when dealing with large numbers of client connections. This is essential in our case, because all clients must communicate with the central server at regular intervals, in order to download the latest database updates. We hosted the server on Heroku [29], a platform-as-a-service provider. Furthermore, we implemented the *SMS*'s database with MongoDB [30], a cross-platform document-oriented database (a.k.a. NoSQL). Our decision to employ a NoSQL database is due to the nature of the stored data structure, where we represent the space authority data with a single database document. This approach facilitates easy update and retrieval operations. Finally, we hosted the database on *MongoDB Atlas*, a cloud-database service provider [31].

To allow the space authorities to interact with the *SMS*, we designed a web interface using React.js [32], a JavaScript library for building user interfaces. This module is hosted within a local machine with internet connectivity, and allows the space authority to interact with the *SMS*.

B. Operations Implementation

Space authority registration and login. ICMS's web interface module enables a space authority to communicate and interact with the central server (*SMS*). The web interface can be accessed from any machine connected to the internet. The main screen of the web interface displays two options that are available for the space authority: *registration* and *login*. The space authority enters its *unique identifier*, *passphrase*, and its *private key* in the registration form. The passphrase is used as a decryption key to decrypt the private key, which is AES-encrypted and stored on the local machine of the space authority. In addition, the private key can be selected from the file system by clicking on the browse private key button. Once the register button is clicked, the page leverages the decrypted private key to digitally sign the registration request, and subsequently sends it to the central server (as discussed in Section III-E1). After the *SMS* processes the request, it will respond back with the result (e.g., success/failure of registration). The login page is practically identical to the registration page, and it is invoked by the space authority to login and retrieve its data from the *SMS*. Once the login request is verified by the *SMS*, it responds back with all the details stored in the central database and associated with the logged in space authority.

Updating space authority details. Once the space authority is logged in, it can display all its data (space list data), as illustrated in Fig. 2. By updating such data, the space authority can define access restrictions on spaces under its control and

perform delegation/revocation of space ownership. Listing 1 shows the structure of the space list data.

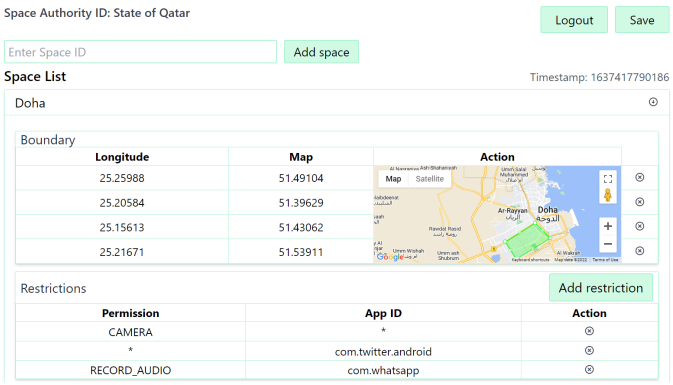


Fig. 2: Space authority details screen

Listing 1: Space list data structure (in JSON format)

```
{spaceList: [
    space: {
        id: String,
        boundary: [
            latitude: Double,
            longitude: Double
        ]
    },
    restrictionRecords: [{
        permission: String,
        appId: String
    }],
    delegationRecords: [{
        space: {
            id: String,
            boundary: [
                latitude: Double,
                longitude: Double
            ]
        },
        delegatorId: String
    }]
}
```

Client-side implementation. As highlighted earlier, we implemented the *TSM* as a system service within the Android operating system. We named the service *space manager*. The service periodically executes two main tasks: *database update* and *access control*.

In the database update task, the *TSM* performs the following checks:

- **Location service status:** Ensures that location service is enabled on the mobile device, in order to obtain the user's location.
- **Internet connectivity status:** Ensures that either WiFi or cellular data service is enabled, so that it can communicate with the *SMS* to ensure that the local copy of the database is recent.

If any of the above-mentioned checks fail, the *TSM* switches to the fail-secure mode. On the other hand, if all checks are successful, the *TSM* performs the **database retrieval** operation (discussed in Section III-E3) to obtain the latest updates from the *SMS*, and applies them to the locally stored copy. Nevertheless, ICMS can survive internet service unavailability for short time intervals. This is achieved with a pre-configured *time threshold*, i.e., if the *TSM* is not able to communicate with the *SMS* for a period of time less than the set *threshold*, the *TSM* will consider the local copy as fresh and valid.

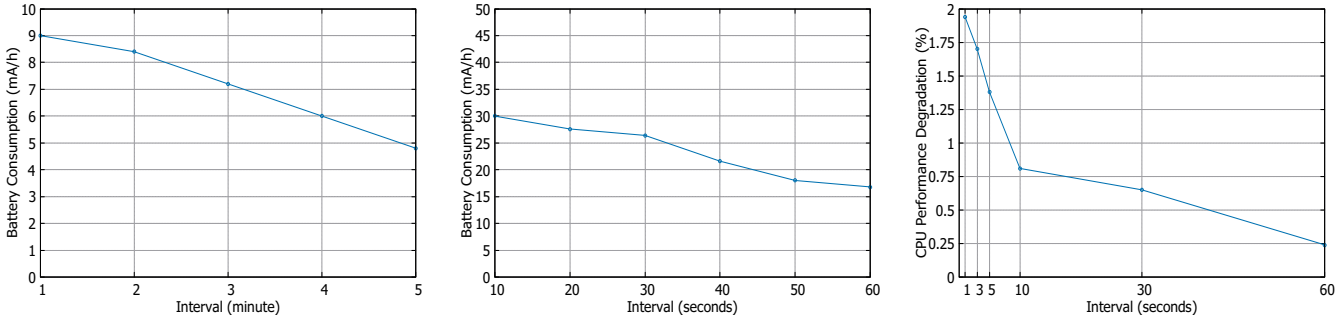
During the access control task, the *TSM* continuously performs the **restriction enforcement** operation (also discussed in Section III-E3) that reads the user's current location and applies the associated restrictions. We achieved restriction enforcement by explicitly calling the *revokeRuntimePermission()* method to revoke permissions, and the *setApplicationEnabledSetting()* method to disable applications from the *PackageManager* object, an object used in the Android environment to access information about application packages. Moreover, to prevent the end-user from re-granting restricted permissions or re-enabling restricted applications, we hooked our custom code to check for enforced restrictions on both the *grantRuntimePermission()* and *setApplicationEnabledSetting()* methods in the *PermissionManagerService* object. This is to prevent any restriction bypassing attempts by the end-user.

We implemented an aggressive fail-secure mode, where we disable all applications from the user's mobile device. In other words, our priority is to prevent end-users from bypassing location-based restrictions, while ignoring user experience and device usability. At the same time, the *TSM* will try to enable the required services (location service and internet connectivity) to ensure that ICMS can function properly, by being able to obtain the user's current location from the *LSP* and to communicate with the *SMS* to pull a recent copy of the database. We should emphasize again that the implementation of the fail-secure mode is decided by the system developers, and less aggressive approaches may be employed instead.

V. Evaluation

This section presents the results of our experimental evaluation that measures various performance metrics of our implementation. First, we highlighted the cost of the cryptographic primitives involved in ICMS's operations. In all measurements, we used a 4096-bit RSA key-pair along with the SHA-512 hash algorithm for the digital signature. Furthermore, we focused on evaluating the performance of the client-side operations, since they are the most frequently executed operations in our system. In addition, we intentionally left out the calculation of the network connection overhead from the reported results, because this cost depends on many variables, including internet speed, network latency, caching, and others. Finally, we used the Battery Historian tool [33], developed by Google, to inspect battery related information and measure power consumption on the mobile device, and also used the Geekbench 5 benchmark [34] to measure CPU performance.

The space manager server (*SMS*) is hosted within a free package in Heroku, which comes with 512MB of RAM and a virtual shared processor [35]. The central database is hosted on MongoDB Atlas free subscription that comes with shared RAM and a virtual processor [36]. We ran all the space authority's operations (performed via the web interface) on a laptop equipped with a 1.80GHz Intel Core i7 processor and 8GB of RAM. The mobile device used in the evaluation was



(a) Battery consumption of database update retrieval operation (b) Battery consumption of GPS location updates (c) CPU performance degradation compared to the baseline measurement

Fig. 3: ICMS's impact on the mobile device's performance

a Samsung Galaxy J7 model with 2 GB of RAM and a 1.6 GHz Octa-core Exynos 7870 processor.

Space authority registration and login: In both the space authority registration and login operations, the space authority performs a digital signature generation while the space manager server performs digital signature verification. After measuring their overheads, the signature generation takes an average of 7 ms while the verification process takes an average of 0.3 ms.

Updating space authority details: All the operations that involve updating the space authority's details (space ownership registration, access restriction definition, space ownership delegation, space ownership delegation revocation) only require the space authority representative to use the web interface to perform updates and modifications to the authority's data, as shown in Fig. 2; thus, there is no computational overhead at the *SMS* for performing such operations. Nevertheless, all these operations end by invoking **Update space list details**, which involves digital signature generation and verification processes. Consequently, the cost at the *SMS* is quite low, and consists of one signature verification (0.3 ms) for each update.

Given the high execution frequency of the client-side operations and the limited resources of mobile devices, we next focus of assessing the performance of the client-side operations.

Retrieval of database updates: We measured the average battery consumption (mA/h) of the database retrieval operation when executed at different frequencies. In particular, Fig. 3a presents the additional battery consumption when compared to a reference measurement, where the database retrieval operation is disabled. The figure shows a battery consumption of 4.8 mA/h to 9 mA/h, which is rather negligible. A typical smartphone battery today has a capacity of 3,000 mAh which means that, even when the update operation is performed every minute, the system consumes just 0.3% of the battery capacity per hour. And, in reality, the update interval would typically be larger, because we do not expect space authorities to issue frequent updates on their data.

User location updates: We also measured the battery consumption when varying the frequency of obtaining fresh GPS measurements. Fig. 3b illustrates the the additional battery consumption when compared to a reference measurement, where GPS location updates are disabled. Clearly, the battery consumption increases as the location update request interval shortens. In particular, the hourly battery consumption for location updates ranges from 0.56% of the battery capacity (16.8 mA/h) for 60-second intervals, to 1% (30 mA/h) for 10-second intervals. Note that, location updates also incur an additional computational cost, because the system has to map the user's location inside a controlled space (if any). The complexity of this operation is linear in the number of spaces in the database. In practice, the CPU time for this operation is very low, requiring less than 50 μsec for 10 spaces, each with 10 sides.

Enforcing restrictions: The restriction enforcement routine takes an average of 85 ms. This time includes reading the retrieved restriction policies, remembering user permissions configuration, enabling/disabling applications, and granting/revoking permissions based on restriction policies. We also measured the performance of the CPU when executing the operation with varying frequency. Fig. 3c depicts the change in the multi-core CPU performance score (as a percentage), compared to the baseline score. (The baseline score is 412 when ICMS is disabled.) As expected, the degradation in performance increases as the interval of running the operation decreases. However, the overhead is rather minor, with the most frequent execution of the operation (every 1 second) resulting in less than 2% performance degradation compared to the baseline measurement.

Entering fail-secure mode: Based on our fail-secure mode implementation, the measured time for entering a mobile device into a fail-secure mode is 17 ms. This is a relatively inexpensive operation, as we are disabling all applications on the mobile device without performing any type of checks and validations.

A. Security Analysis

Before concluding our evaluation, we should discuss the security properties of ICMS and identify viable attacks against our system. In a nutshell, we consider ICMS as compromised,

when the location-based restriction policies set by any space authority are not enforced. An adversary may achieve this goal by attacking the various entities involved. First, if the adversary compromises the *SA* (including its private signing key), they can modify or eliminate the restrictions under the physical space(s) it controls. It is, thus, imperative for the *SA*s to secure their own systems against such attacks. Second, the adversary may compromise the *SMS* and attempt to modify the restrictions database. However, this attack is not feasible, because the restrictions (including empty lists) are digitally signed by the corresponding *SA*s.

At the client-side, we assume that the *TSM* is a trusted module, pre-installed in the device's operating system. As such, users do not have access to it, and all operations performed at the client-side (e.g., restriction enforcement) are guaranteed to be executed as expected. Additionally, application developers cannot manipulate ICMS's operations, because they are executed in kernel-level. Moreover, all operations are running transparently, and developers are not even required to introduce any changes to their application logic. Thus, application developers are unable to manipulate or circumvent the access restrictions. However, there are two feasible attacks to circumvent the security of ICMS: device rooting and GPS location spoofing. (Note that both can be considered as side-channel attacks and not direct attacks against the ICMS system.) First, a rooted device allows the user to remove or deactivate any part of the OS, including ICMS. To detect such attacks, a space owner may perform a physical inspection of all devices that enter their premises (via widely available apps that detect rooted devices). Alternatively, the adversary may utilize various GPS spoofing techniques to fake their location [37] and circumvent location-based restrictions. To eliminate this inherited weakness, in our future work, we plan to rely on Bluetooth Low Energy (BLE) beacons to perform device localization.

VI. Discussion

In this section, we shed light on some aspects of ICMS, highlight its limitations, and discuss some research directions that we plan to investigate in the future.

A. Flexible Space Division

Our system design allows a space authority to sub-divide an owned space to an arbitrary number of sub-spaces. This important property enables two key features: (i) it allows a single space authority to apply different access restrictions in each of the sub-spaces; and (ii) it allows an individual space authority to delegate/revoke the control of sub-spaces to/from different space authorities.

B. Flexible Restriction Policies

ICMS's flexibility stems from (i) the ease in specifying physical spaces; and (ii) the level of granularity offered when applying specific restriction policies into diverse locations and under various conditions. Table III illustrates the flexibility of ICMS, by applying fine-grained restriction policies to achieve a desired outcome in different environments.

C. Chain of Delegations

Due to the flexible design of our space list data structure (presented in Listing 1), ICMS enables the construction of a chain of delegations, where a specific space (or sub-space) is delegated from the original owner to the delegated owner via a chain of other intermediate delegated authorities. Furthermore, the system allows any of the intermediaries to revoke its delegation. As a result, the chain will be broken and delegations will stop when reaching the authority revoking the delegation.

D. Services Requirements

One of ICMS's limitations is the requirement of having some system services enabled at all times, in order for the system to function correctly. These services include location service and Internet connectivity. First, location service is necessary for the *TSM* to obtain the user's current location and, in our implementation, we opt to enter the fail-secure mode once the location service is disabled from the mobile device for any reason. This is to ensure that the end-user does not bypass any access restrictions (unauthorized access). Similarly, Internet connectivity is essential for the *TSM* to retrieve database updates from the *SMS*. However, to accommodate unexpected network connection interruptions, we configure a certain threshold time (in seconds), where we consider the current local copy of the database as fresh and valid. If that threshold time is exceeded without any successful communication with the *SMS*, the *TSM* forces the mobile device to enter the fail-secure mode. The threshold time can be fine-tuned during implementation to best suit the environment conditions and the system's purpose of use. However, as future work, we plan to propose a system design that does not require such services to be available, with an aim to increase the system's resiliency and maximize its operational efficiency.

E. Accuracy of User Location

Another inherent limitation, which is not related to ICMS's design, is the accuracy of the measured location information. This limitation is tied to the existing localization technologies and relates to the hardware modules available in today's mobile devices. Such an inherent limitation diminishes the level of granularity for space identification in the system, especially when dividing and configuring indoor areas (e.g., due to the low accuracy of GPS technology in indoor environments). Still, we expect ICMS to perform very well when configured in outdoor or larger spaces. Nevertheless, in our future work, we plan to investigate the applicability of more accurate localization technologies (such as BLE beacons) that will allow ICMS to work seamlessly with small and/or indoor spaces.

F. Support for 3-D Spaces

ICMS's current design and implementation do not support the definition of three-dimensional spaces (e.g., to enforce access restrictions in a specific story of a multi-story building). This is because we rely only on the latitude and longitude coordinates to define space boundaries. However, the system's

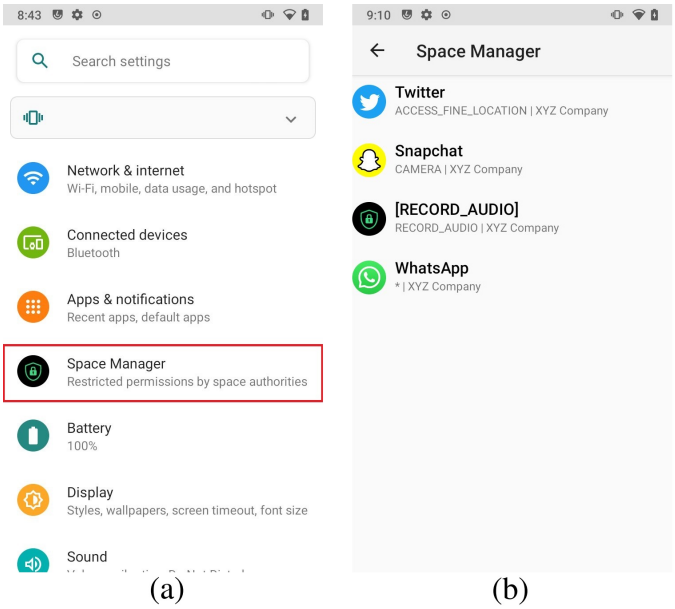
TABLE III: Empirical evaluation to illustrate ICMS's flexibility under diverse location types

				
Location	Military Base	Classroom (exam mode)	Classroom (lecture mode)	Shopping Mall
Desired outcome	Avoid taking photos/videos, recording sensitive discussions, or exposing the building's location.	Prevent students from taking photos of the exam paper or using WhatsApp to communicate with outside parties.	Restrict the use of social media apps while in classroom.	Users have complete control over their mobile devices.
Desired restrictions	Disable usage of camera, microphone and location sharing capabilities.	Disable usage of camera and WhatsApp.	Disable most common social media apps (e.g., Facebook, Snapchat, Instagram).	No restrictions.
Restriction policies as configured in the proposed system	$\{CAMERA, *\}$, $\{MICROPHONE, *\}$, $\{ACCESS\_COARSE\_LOCATION, *\}$	$\{CAMERA, *\}$, $\{*, WHATSAPP\}$	$\{*, FACEBOOK\}$, $\{*, SNAPCHAT\}$, $\{*, INSTAGRAM\}$	*N/A* (no restrictions)

design is ready to incorporate the *altitude* dimension to the location information. As a result, the space authority can then define its space boundary in the 3-D space, using latitude, longitude, and altitude information. We plan to introduce this enhancement to our system implementation, given that today's mobile devices have built-in sensors and modules that measure the device's altitude from a specific plane (e.g., sea level). Moreover, we plan to investigate its accuracy and performance.

G. Transparency of Decisions

ICMS is very flexible, in that it allows a chain of delegations for a specific physical space, where access restriction policies may be configured by multiple authorities per space. Therefore, for transparency, the system provides the end-user with a full list of restrictions currently applied on their mobile device's applications and services. Additionally, the list shows the space authority that is enforcing each of the restriction policies. In our implementation, we customized the default Settings app to include an item in the menu titled *Space Manager*. Once clicked, the end-user is presented with the full list of currently applied restrictions. Fig. 4 best illustrates the customization introduced to the default Settings app.



(a) (b)

Fig. 4: Customization introduced on the built-in Settings App. (a) Settings App main menu showing *Space Manager* item. (b) *Space Manager*'s settings screen showing currently applied restrictions.

VII. Conclusions

This paper introduced a novel space-based access control system, ICMS, that offers the advantages and flexibility not possible by existing works. ICMS enables a scalable, hierarchical configuration of physical spaces, where the space owner can define fine-grained access restrictions and enforce them on the services run by mobile devices inside its controlled space. Moreover, all the system operations are seamless to the end-user and do not require any interaction. In addition, ICMS promotes transparency by revealing the space authorities enforcing the current restrictions to the end-user. More importantly, our system preserves the privacy of the user's location information by not disclosing their location to any party. A proof-of-concept prototype of ICMS using Android-based mobile devices along with a cloud platform is implemented, in order to prove its feasibility in real-life applications. Several experiments with the prototype are conducted to demonstrate the promising results and novel flexibility for the required access policies in diverse location types.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable and constructive feedback that allowed us to improve significantly the quality of this work.

References

[1] A. Jay. Number of apps in apple app store in 2020/2021: Demographics, statistics, and predictions. https://www.statista.com/statistics/276623/

number-of-apps-available-in-leading-app-stores/#:%E2%88%BC:text=The+Apple+App+Store+was,million+available+apps+for+iOS.
[2] T. N. Chowdary, P. P. Raj, C. Anupama, and D. Haritha, "Location & user profile based mobile application access," in *International Conference on Informatics and Analytics*. Association for Computing Machinery, 2016.
[3] F. Roesner, D. Molnar, A. Moshchuk, T. Kohno, and H. J. Wang, "World-driven access control for continuous sensing," in *ACM Conference on Computer and Communications Security*. Association for Computing Machinery, 2014, p. 1169–1181.
[4] C. E. Rubio-Medrano, S. Jogani, M. Leitner, Z. Zhao, and G.-J. Ahn, "Effectively enforcing authorization constraints for emerging space-sensitive technologies," in *ACM Symposium on Access Control Models and Technologies*. Association for Computing Machinery, 2019, p. 195–206.
[5] Y. Borse, D. Patole, and P. Ahirao, "Geo-encryption: A location based encryption technique for data security," in *International Conference On Computing, Communication, Control And Automation*, 2019, pp. 1–4.
[6] S. Choi, C. Gutierrez, H.-S. Lim, S. Bagchi, and E. Bertino, "Secure and resilient proximity-based access control," in *International Workshop on Data Management & Analytics for Healthcare*. Association for Computing Machinery, 2013, p. 15–20.
[7] J. Li, A. Squicciarini, D. Lin, S. Liang, and C. Jia, "Secloc: Securing location-sensitive storage in the cloud," in *ACM Symposium on Access Control Models and Technologies*, 06 2015, pp. 51–61.
[8] T. Wang, Y. Liu, Q. Pei, and T. Hou, "Location-restricted services access control leveraging pinpoint waveforming," in *ACM Conference on Computer and Communications Security*. Association for Computing Machinery, 2015, p. 292–303.
[9] I. F. Cruz, R. Gjomemo, B. Lin, and M. Orsini, "A location aware role and attribute based access control system," in *International Symposium on Advances in Geographic Information Systems*. Association for Computing Machinery, 2008.
[10] C. A. Ardagna, M. Cremonini, E. Damiani, S. D. C. di Vimercati, and P. Samarati, "Supporting location-based conditions in access control policies," in *ACM Symposium on Information, Computer and Communications Security*. Association for Computing Machinery, 2006, p. 212–222.
[11] A. C. Hsu and I. Ray, "Specification and enforcement of location-aware attribute-based access control for online social networks," in *ACM International Workshop on Attribute Based Access Control*. Association for Computing Machinery, 2016, p. 25–34.
[12] O. Oluwatimi, D. Midi, and E. Bertino, "A context-aware system to secure enterprise content," in *ACM Symposium on Access Control Models and Technologies*. Association for Computing Machinery, 2016, p. 63–72.
[13] P. D. Martin, M. Rushanan, T. Tantillo, C. U. Lehmann, and A. D. Rubin, "Applications of secure location sensing in healthcare," in *ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*. Association for Computing Machinery, 2016, p. 58–67.
[14] B. Shebaro, O. Oluwatimi, and E. Bertino, "Context-based access control systems for mobile devices," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 2, pp. 150–163, 2015.
[15] M. S. Kirkpatrick and E. Bertino, "Enforcing spatial constraints for mobile rbac systems," in *ACM Symposium on Access Control Models and Technologies*. Association for Computing Machinery, 2010, p. 99–108.
[16] M. S. Kirkpatrick, M. L. Damiani, and E. Bertino, "Prox-rbac: A proximity-based spatially aware rbac," in *ACM International Symposium on Advances in Geographic Information Systems*. Association for Computing Machinery, 2011, p. 339–348.
[17] Y. Zhu, D. Ma, D. Huang, and C. Hu, "Enabling secure location-based services in mobile cloud computing," in *ACM SIGCOMM Workshop on Mobile Cloud Computing*. Association for Computing Machinery, 2013, p. 27–32.
[18] U. Hengartner and P. Steenkiste, "Implementing access control to people location information," in *ACM Symposium on Access Control Models and Technologies*. Association for Computing Machinery, 2004, p. 11–20.
[19] L. Karimi, B. Palanisamy, and J. Joshi, "A dynamic privacy aware access control model for location based services," in *IEEE International Conference on Collaboration and Internet Computing*, 2016, pp. 554–557.
[20] P. Mainali, C. Shepherd, and F. A. Petitcolas, "Privacy-enhancing context authentication from location-sensitive data," in *International Conference on Availability, Reliability and Security*. Association for Computing Machinery, 8 2019.
[21] A. Abdou, A. Matrawy, and P. C. van Oorschot, "Accurate manipulation of delay-based internet geolocation," in *ACM on Asia Conference on Computer and Communications Security*. Association for Computing Machinery, 2017, p. 887–898.
[22] A. v. Cleeff, W. Pieters, and R. Wieringa, "Benefits of location-based access control: A literature study," in *IEEE/ACM International Conference on Green Computing and Communications International Conference on Cyber, Physical and Social Computing*, 2010, pp. 739–746.
[23] M. Decker, "Requirements for a location-based access control model," in *ACM International Conference on Advances in Mobile Computing and Multimedia*. Association for Computing Machinery, 2008, p. 346–349.
[24] E. Bertino and M. S. Kirkpatrick, "Location-based access control systems for mobile users: Concepts and research directions," in *ACM International Workshop on Security and Privacy in GIS and LBS*. Association for Computing Machinery, 2011, p. 49–52.
[25] (2021) I control my space (icms). https://github.com/akhalil-qa/ICMS.
[26] (2020) Android architecture — android open source project. https://source.android.com/devices/architecture.
[27] (2021) Java — oracle. https://www.java.com/en/.
[28] (2021) Node.js. https://nodejs.org/.
[29] (2021) Cloud application platform — heroku. https://www.heroku.com/.
[30] (2021) The most popular database for modern apps — mongodb. https://www.mongodb.com/.
[31] (2021) Managed mongodb hosting — database-as-a-service — mongodb. https://www.mongodb.com/cloud/atlas.
[32] (2021) React - a javascript library for building user interfaces. https://reactjs.org/.
[33] (2017) Battery historian tool. https://github.com/google/battery-historian.
[34] (2019) Geekbench 5 - cross-platform benchmark. https://www.geekbench.com/.
[35] (2020) Dyno types — heroku dev center. https://devcenter.heroku.com/articles/dyno-types.
[36] (2021) Pricing — mongodb. https://www.mongodb.com/pricing.
[37] K. Zeng, Y. Shu, S. Liu, Y. Dou, and Y. Yang, "A practical gps location spoofing attack in road navigation scenario," 02 2017, pp. 85–90.

Ahmed Khalil Abdulla received a BEng degree in Computer Systems Engineering from The University of Manchester, United Kingdom, in 2012, and the MSc degree in Cybersecurity from Hamad Bin Khalifa University (HBKU), Qatar, in 2018. Currently, he is a PhD candidate in HBKU with a research interest in access control models and systems. He is a member of the ACM.

Spiridon Bakiras received the B.S. degree in Electrical and Computer Engineering from the National Technical University of Athens, the M.S. degree in Telematics from the University of Surrey, and the Ph.D. degree in Electrical Engineering from the University of Southern California. Currently, he is an associate professor in the Infocomm Technology Cluster at Singapore Institute of Technology. Before that, he held teaching and research positions at Hamad Bin Khalifa University, Qatar, Michigan Technological University, the City University of New York, the University of Hong Kong, and the Hong Kong University of Science and Technology. His current research interests include security, privacy, and applied cryptography. He is a member of the IEEE and ACM, and a recipient of the U.S. National Science Foundation (NSF) CAREER award.

James She is currently an associate professor in the Division of Information and Computing Technology, College of Science and Engineering, Hamad Bin Khalifa University, Doha, Qatar. His current research interests include machine learning and IoT technologies for smart, sustainable and interactive city applications.