2013

The design and implementation of a smartphone and Bluetooth-based criminal tracking system

Sen Wang
Iowa State University

The design and implementation of a smartphone and Bluetooth-based criminal tracking system

by

Sen Wang

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Science

Program of Study Committee:

Ying Cai, Major Professor

Lu Ruan

Zhao Zhang

Iowa State University

Ames, Iowa

2013

DEDICATION

I would like thank Dr. Ying Cai without whose support I would not be able to complete this work. I would also like to thank my friends and family for their loving guidance during the writing of this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this thesis. First and foremost, I appreciate Dr. Ying Cai for his guidance, patience and support throughout this research and the writing of this thesis. His insights and words of encouragement have often inspired me and renewed my hopes for completing my graduate education. I would also like to thank my committee members for their efforts and contributions to this work: Dr. Lu Ruan and Dr. Zhao Zhang.

ABSTRACT

This thesis describes the design and implementation of a real-time criminal tracking system. A criminal under tracking is asked to wear a low energy Bluetooth device and carry a smartphone. The Bluetooth device is secure on his body (e.g., hand or foot) and communicates with the smartphone, which communicates with a central server through cellular networks. The smartphone monitors the status of the Bluetooth device and reports to the server in real time when the status changes (e.g., connection lost or device being taken off). Moreover, it monitors the criminal's movement and reports to the server whenever the criminal moves into an alert zone, a geographic region where the law enforcement wants the criminal's movement to be tracked. Compared to the existing tracking approaches, our system has the following desired features. 1) *Scalable*. Instead of having a criminal to report its location all the time, the system allows one to configure where and when the criminal needs to be tracked, thus minimizing both mobile communication cost and server processing cost. 2) *Low-cost*. The system uses only off-shelf components (e.g., Bluetooth device and smartphone) which communicate through regular wireless networks. 3) *Secure*. The communication between a Bluetooth device and the corresponding smartphone is authenticated through One Time Password.

CHAPTER 1. OVERVIEW

In ancient China, there was no prison. People were said to respect each other and faithful to their own words. When the law enforcement wanted to put someone in custody, it simply drew a circle on the ground and asked the person to stay there. The man would stay there without being guarded until he was told to leave. This earliest concept of jail may sound like a tale today. Prisoners are now jailed in costly and heavily guarded buildings to make sure they cannot escape. For those minor offenders, the law enforcement does not want to put them in jail, but would like to track their movement in real time.

A number of techniques have been developed for real-time personnel tracking. A popular approach is using GPS. An example of such products is $3M^{TM}$ One-Piece GPS Tracking System [6], a small light-weighted tamper-resistant electrical device. The product is tightened on a criminal's ankle using the strap and continuously tracks the criminal in real time. The device saves the location data and transfers them to the monitoring center for processing. The law enforcement can also configure zone restrictions where the criminal is not allowed to enter. Violations are reported to both criminals and officers. Such tracking systems are highly reliable, but are expensive in deployment. The above mentioned 3M product costs $4.30 per day and thousands of dollars a year for each unit. Special spectrum is also needed for the communication between the tracking device and the server.

The wide deployment of cellular wireless technologies makes it possible to develop low-cost tracking devices. The wireless service provider could use the cellular phone signal to localize a user's position through signal triangulation. In some countries like China, this method has been used by the law enforcement for personnel tracking. The minor offenders are required to carry a cellular phone so that their movement can be tracked. This approach is low cost, but has some noticeable drawbacks. It needs to work strictly with the wireless service provider,

because only the provider can do the signal triangulation. In particular, it is hard to verify if a criminal indeed carries the cellular phone all the time. The law enforcement could dial a cellular phone from time to time to check. However, it is difficult to verify if the person who answers the phone is indeed the criminal. It is possible to use voice recognition, but one can still relay his answer with assistance of a third party (e.g., a friend).

In this thesis, we present the design and implementation of a low-cost real-time tracking system. The system consists of three components: server, smartphone, and Bluetooth Low Energy (BLE) device. A person under tracking is given a smartphone and also attached with a BLE device. The BLE device periodically sends signal to the smartphone, which communicates with the server through cellular wireless networks. When the smartphone cannot detect the signal of the BLE device, it reports to the server. The smartphone also tracks the movement of the carrier by monitoring its GPS position and reports the location information to the server when necessary. Instead of sending location information to the server all the time, we allow the system to configure alertzones, each being a circular or rectangular region. The smartphone reports to the server whenever a user moves into or out of an alertzone. To prevent one from using a fake BLE device, we apply the concept of One Time Password (OTP) to authenticate the communication between the BLE device and the smartphone.

The main contributions of this work are summarized as follows:

1. We combine long range communication (between the server and the smartphone) and short range communication (between the smartphone and the BLE device) to develop a new tracking application.

2. We allow the system to configure alertzones for users under tracking. This concept of alertzone avoids having to track a user all the time, also minimizes the mobile communication cost, thus saves its battery consumption. This reduces the privacy concerns and allows the server to track a large number of smartphones.

3. The pairing of a smartphone and a BLE device is authenticated using One Time Password.

CHAPTER 2. RELATED WORK

2.1 Location-Based Services

The global penetration of smartphones has provided a great market for LBS. Smartphones usually have built-in GPS chip, so are location-aware. Location-Based Services (LBS) [18] allow users to retrieve information such as the hotel, restaurant and hospital based on their location. LBS has many important applications:

- Finding the location of people or objects [26] [25] [39] [27];
- Tracking the movement of objects and people [12] [34] [11];
- Providing commercial services such as advertising and sending coupons based on customers' location [10] [17];
- Providing location-based information such as the weather forecast and news [23] [15] [13];
- Providing entertainment services such as location-based games and social network services [38] [33] [16].

In general, there are two strategies to implement LBS:

- The server **pushes** the information. These applications aim at delivering services as soon as users' movement triggers some pre-defined conditions, e.g., entering into a specific region, [14] [19] [32] [20].
- The client **pulls** the information. In such applications, the server responses queries from the client. Examples of such queries include "retrieve the gas station that is nearest to my location", [21] [24] [29] [40].

Our system is basically an application using the first strategy. The smartphone monitors its movement and its connection to the paired BLE device, and reports to the server when certain conditions are met. Examples of these conditions can be the criminal moves into or out of an alertzone, the connection to the BLE device is lost, and so on.

2.2 Bluetooth Low Energy

Bluetooth Core Specification Version 4.0 [1] is the most recent version of Bluetooth wireless technology. This new standard supports two systems of wireless technology: Classic Bluetooth and Bluetooth Low Energy (BLE). Compared to the classic Bluetooth technology, BLE transmits very small data packets with significantly reduced power consumption. The key features of BLE technology include [7]:

- Ultra-low peak, average and idle mode power consumption;
- Ability to run for years on standard, coin-cell batteries;
- Low cost;
- Multi-vendor interpretability;
- Enhanced range.

BLE technologies have been used in many applications [31] [37]:

- Heart Rate Monitor. The system usually consists of a heart rate sensor and a receiver. The heart rate sensor could be fit on the chest by the strap and sends the real-time heart rate data to the receiver. The receiver could store and analyze the data, then give a feedback to the user.
- Running speed and cadence monitor. Nike+ [2] sport watch GPS is such a device. The sport watch could track users' location, pace, distance, calories burned and so on. It could send all these data to the compatible receiver, such as a smartphone. The user can review their running information through the smartphone.

- Bluetooth Tracker. BluTracker [8] is a commercial product that can be used to track people and object. The small electrical device that equipped with a GPS engine could monitor a range of around 2500Ft and work for over 2 months with a charge. The mobile application uses Google maps to show the user's location as well as the tracker's location and it alerts the user if the tracker is out of the monitoring range.

Currently, most BLE applications allow one master device, such as a smart phone, connecting to one slave device, such as a watch or a sensor. These applications work under the 1-to-1 model. Our work is different in that 1) it combines the Bluetooth and wireless cellular networks technologies and 2) the server could track thousands of the BLE devices at the same time.

2.3 One Time Password

One Time Password (OTP) is an approach that generates password that is valid only for one time of use. If a website uses the OTP for authentication, users have to use a different password to login each time. The OTP prevents the replay attack. The password is generated in such a way that an attacker cannot find out the next password even if he has access to all passwords that have been used.

The concept of OTP has been used in authentication and there are a number of different implementations [36] [22] [30] [35]. One way to implement the OTP is paper-based, for example, some banks send a paper with a list of OTPs to the user. In each online transaction, the user is asked to enter a specific password in the list. Another way of implementation is SMS-based, for instance, some banks use the Short Message Service (SMS) to deliver a specific OTP to the user's cellular phone to finish a transaction. Some more advanced approaches are also available, the smartphone today has high-performance computing capability and supports more different functionality. An OTP application could be installed on the smartphone and works as tokens to provide the authentication to more than one resource.

The generation of the OTP is typically implemented by the pseudo-random algorithm or randomized algorithm. There are two common approaches:

- **Mathematical Algorithms**. This approach generates the password based on the mathematical formula. A chain of passwords may look like as below:

  $$f(s), f(f(s)), f(f(f(s)))...$$

  Where s is the initial seed, $f()$ could be a hash function. The first password is $f(s)$, and second password is based on the first one, which is $f(f(s))$, and so on. Namely, the generation of next password is based on the previous one. It can be seen that the choose of $f()$ is the most important part in the method. The $f()$ could be an one-way function, which means it is impossible to find the inverse function $f^{-1}()$. Another option is to choose a cryptographic hash function. The inverse function is possible to be found out, but it could be an computationally infeasible task.

- **Time-synchronized**. This approach uses the current time, rather than the previous password, as the seed to generate new password. Normally, this method is related to a piece of hardware. There is an accurate clock inside this hardware and it is synchronized with another clock on the authentication server. The generation of the new password is based on the current time from the clock.

Considering the memory size and the computation capacity of the Bluetooth chip, we chose the mathematical algorithm to generate the OTP in our application. Instead of using complicated algorithm that may consume significant amount of power, we implemented a simple pseudo-randomized algorithm on the Bluetooth chip. The BLE application runs the algorithm and sends the OTP to the smartphone periodically. The details will be explained shortly.

CHAPTER 3. SYSTEM OVERVIEW

The hardware platform of our system includes a central server and a set of position-aware smartphones, each paired with one BLE device. We will use terms smartphone and mobile client interchangeably. As Figure 3.1 shows, a mobile client receives signals from its paired BLE device and communicates with the server. The server manages the connection with all mobile clients and maintains a database that stores information such as location data reported by mobile clients. The Console component is a web-based GUI that allows one to interact with the system, such as monitoring clients' location and configuring alertzones.

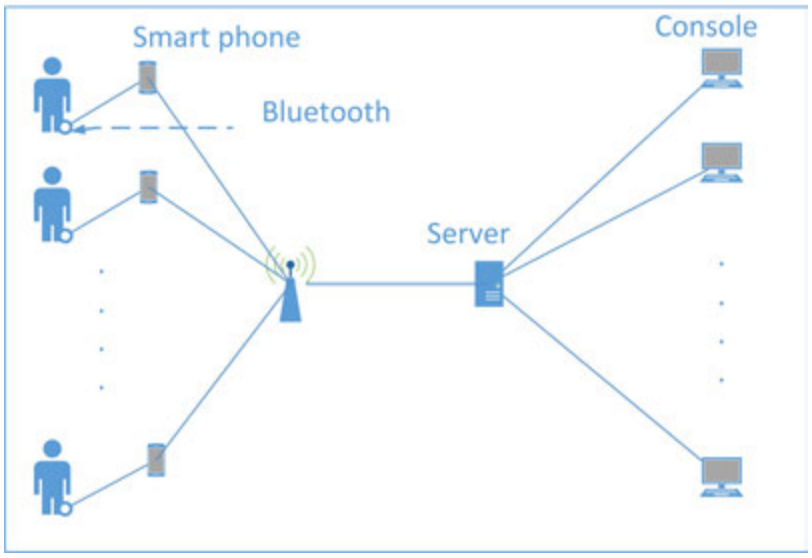

Figure 3.1 System Overview

3.1 Smartphone and BLE device

Each smartphone is paired with a BLE device, which is worn by a criminal all the time. The BLE device sends signals to the smartphone periodically. When the smartphone receives

the single from its BLE device, it knows that the criminal is within a certain range (e.g., around 50 meters). When the connection is lost, the smartphone sends an alert message to the server. To prevent the communication between the BLE device and the smartphone from being hijacked, we use the OTP for communication authentication. The smartphone and the BLE device are assigned the same seed to generate the password. The BLE device calculates a new password every certain time period and sends the password to the smartphone. After receiving the password from the BLE device, the smartphone computes its own password using the same seed and checks if the two passwords match. Besides the OTP, the BLE device also sends other data, such as the battery level value and the accelerometer value.

3.2 Central Server

The server manages all communications with mobile clients through wireless cellular networks. It manages a set of *Alertzone*, each being a user-defined geographic region. The law enforcement wants to know and track whenever a criminal enters into or moves out of an alertzone.

After connections to mobile clients are established, the server sends alertzones to them. Once the mobile client detects its current location is in any alertzone, it sends a message to the server. This mechanism not only reduces the server's workload, but also protects the criminal's location privacy, since the mobile client does not reveal its location to the server when it is outside the alertzone.

The server also maintains a database that stores all information of mobile clients, such as their historical and current location data, the status of the BLE device, and so on.

3.3 Console

The Console is basically a web-based GUI, which shows mobile clients' information in a map application. It loads mobile clients' data from the database and displays them on the webpage. It allows one to monitor criminals' movement and create/remove alertzones on the webpage.

CHAPTER 4. DETAILED SYSTEM DESIGN

In this section, we present the design of the mobile client, the BLE device, the server and the console in detail.

4.1 Mobile Client Design

4.1.1 Mobile Client Data Structure

The mobile client consists of several data structures, Figure 4.1 and Figure 4.2 show the relationship between each other.

Figure 4.1 Mobile Client Data Structure

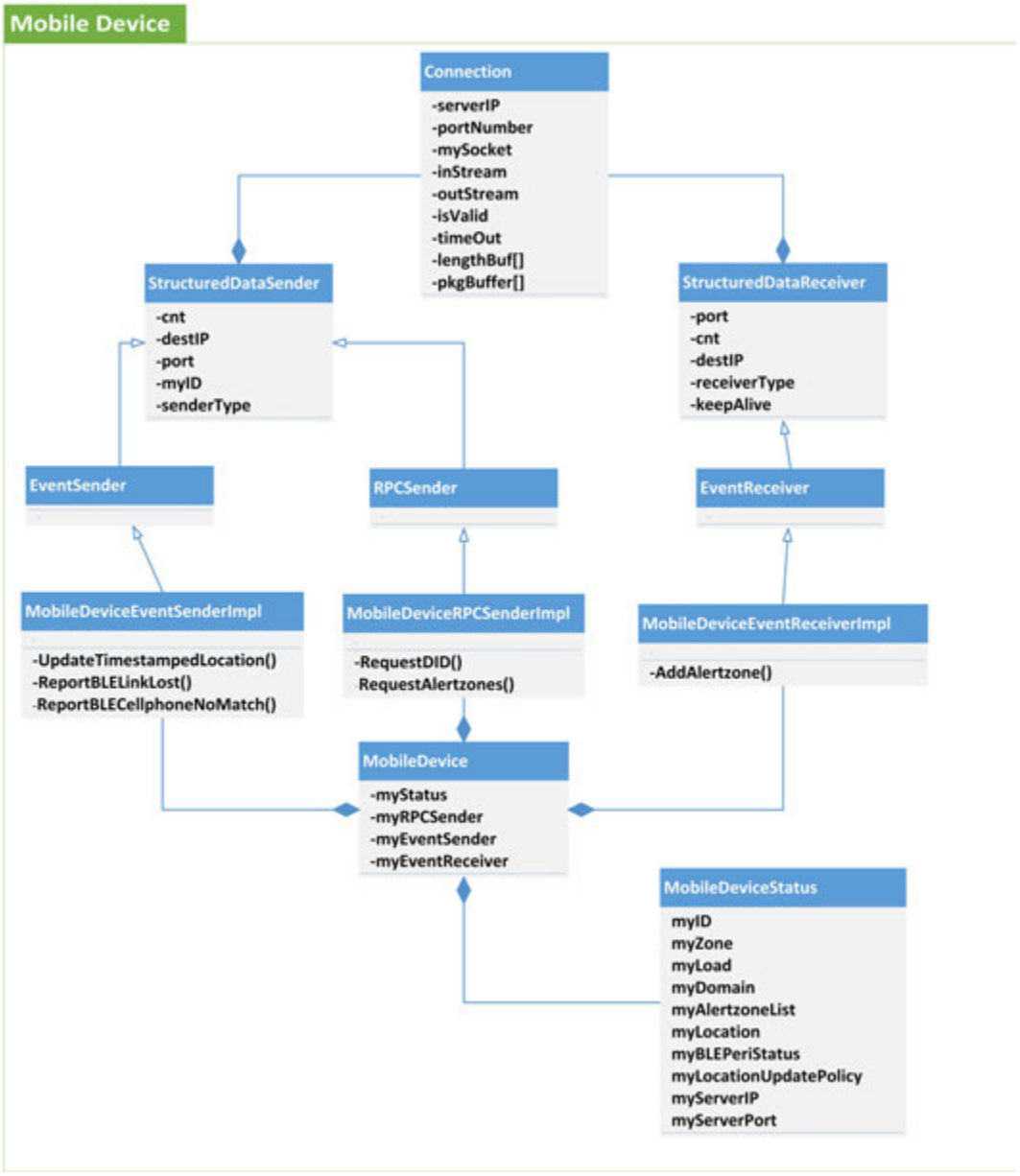

Figure 4.2 Mobile Client Design

The definition of each class is below:

- Class **TimeStampedLocation** has four attributes:
  - latitude: The latitude of the mobile client's current location.
  - longitude: The longitude of the mobile client's current location.
  - altitude: The altitude of the mobile client's current location.

  - time: The timestamp of the mobile client's current location.

- Class **Region** has eight attributes:

  - regionShape: Indicates the shape of the region, *circle* or *rectangle*.
  - centerX: Indicates the latitude of the central point.
  - centerY: Indicates the longitude of central point.
  - radius: Indicates the radius of the region, only if it is circle.
  - upperX: Indicates the latitude of the upper corner point, only if the region is rectangle.
  - upperY: Indicates the longitude of the upper corner point, only if the region is rectangle.
  - lowerX: Indicates the latitude of the lower corner point, only if the region is rectangle.
  - lowerY: Indicates the longitude of the lower corner point, only if the region is rectangle.

- Class **Trigger** has two attributes:

  - inOut: Indicates how the trigger is fired, moving in the region or moving out of the region.
  - direction: Indicates in which direction the trigger is fired.

- Class **Alertzone** has three attributes:

  - region: Indicates the bounding box of this alert zone.
  - trigger: Indicates how and when to download a warning message.
  - url: the link of a warning message, which could be downloaded when the trigger is fired.

- Class **LocationUpdatePolicy** has three attributes:

  - updatePolicyType: Indicates how the mobile client update its real-time location, by *periodically* or *on demand*.
  - timeDuration: Indicates how frequently the mobile client updates the location if the policy type is on demand.
  - updateDerivation: Indicates how far away the mobile client updates the location.

- Class **BLEPeriStatus** records the status of the paired BLE device, such as the IEEE address of the BLE device, the current battery level, the accelerometer value and so on.
  - BID: The unique ID of the BLE device.
  - ieeeAddress: The MAC address of the BLE device.
  - batteryLevel: The current battery level of the BLE device.
  - accelX: The x-axis value of the accelerometer.
  - accelY: The y-axis value of the accelerometer.
  - accelZ: The z-axis value of the accelerometer.
  - isConnected: Indicates if the BLE device is connected to the smartphone.
  - isBounded: Indicates if the BLE device is authenticated by the smartphone.

- Class **MobileDeviceStatus** records the current state of the mobile client with ten attributes:
  - myID: Indicates the unique ID of the mobile client.
  - myZone: Indicates the awareness zone of the mobile client using an instance of Class Region.
  - myLoad: Indicates the number of alertzones it can carry at most.
  - myDomain: Indicates the domain that carries all alertzones using an instance of Class Region.
  - myAlertzoneList: The set of alertzones.
  - myLocation: Indicates the current location of the mobile client using an instance of Class TimeStampedLocation.

  - myBLEPeriStatus: Indicates the status of the paired BLE peripheral device using an instance of Class BLEPeriStatus.
  - myLocationUpdatePolicy: Indicates the policy to update location to the server using an instance of Class LocationUpdatePolicy.
  - myServerIP: Indicates the IP address of the central server.
  - myServerPort: Indicates the port number of the central server.

- Class **MobileClient** handles the message exchange with the central server. It has four attributes:
  - myStatus: Indicates the current state of the mobile client with an instance of Class MobileDeviceStatus.
  - myRPCSender: Sends message to the central server and receives the responding message, such as $REQUEST\_DID$.
  - myEventSender: Sends the message to the central server, such as $UPDATE\_LOCATION$.
  - myEventReceiver: Receives the message from the central center, such as $ADD\_ALERTZONE$.

4.1.2 Mobile Client Operations

The application running on the mobile client consists of four main components: *Initialization*, *MessageListener*, *RegionMonitor* and *BLEMonitor*. The following notations are used in the discussion of these components:

- *myID* : the unique ID of the mobile client.
- *myBLEUUID* : the unique ID of the BLE device that the mobile client pairs with.
- *myAlertzoneList* : the list of alertzones.

Initialization: This procedure is called when the mobile client is powered on:

1. Spawn thread $MessageListener$;
2. Send a $REQUEST\_ALERTZONES$ message to the server;
3. Call procedure $BLEPairing$;
4. Spawn thread $RegionMonitor$;
5. Spawn thread $BLEMonitor$.

MessageListener: The client listens to messages and handles them as follows:

- $SetAlertzones(l)$:
  - Set $myAlertzoneList = l$;
- $AddAlertzone(z)$:
  - Add z to $myAlertzoneList$;
- $DeleteAlertzone(z)$:
  - Remove z from $myAlertzoneList$.

RegionMonitor: While the mobile client is moving, it runs the following task:

- If the current location xyz is inside any alertzone z in $myAlertzoneList$, then it sends a $UPDATE\_LOCATION$ message to the server.

BLEMonitor: When the mobile client powers on, it runs the following tasks:

- If the BLE device is not connected, it calls $BLEPairing$;
- If the BLE device is connected, the mobile client checks the OTP from the BLE device, if the password matches the one computed by itself, it listens to the next OTP;
- if passwords are not matched, it sends the $REPORT\_BLE\_CELLPHONE\_NO\_MATCH$ message to the server and re-initialize the connection to the BLE device.

BLEPairing: The procedure is called when the mobile client searches the paired BLE peripheral device:

1. Search all BLE devices around;
2. Search for the specific BLE device with $myBLEUUID$;
3. If the specific device is not found, the mobile client sends $REPORT\_BLE\_CELLPHONE\_NO\_MATCH$ message to the server; if the specific BLE device is found, initialize the connection to the BLE device.

4.2 BLE Device Design

The main role of the BLE device is sending the OTP to the mobile client continuously. The application running on the BLE device consists of three components, *Initialization*, *Sending Notification*, and *OTP Design*.

4.2.1 Initialization

The BLE device powers on and advertises itself. The mobile client searches for the device by checking the information of the advertisement, such as the device ID. Once the mobile client finds out the matched ID, it requests a connection. If the connection is established, the BLE device starts to run services.

4.2.2 Sending Notifications

During the connection, the BLE device calculates the OTP and sends it to the mobile client periodically. Moreover, the BLE device also sends other data, such as the battery level value and the accelerometer value. There are two BLE profiles used in the application:

- **Proximity Profile** [5] defines how the device behaves when the connection to the peer device is lost and if there is an alert triggered, which includes two services below:
  - **Link Loss Service** [4] triggers an alert when the connection is lost.
  - **Immediate Alert Service** [3] configures the alert type.
- **BLETracker profile** defines the OTP generation and exchange during a connection, which is developed by ourselves.

4.2.3 OTP Design

We choose the mathematical algorithm to implement the OTP. Since the complex algorithm drains the battery quickly, we choose a simple pseudo-random algorithm, the Linear Congruential Generator (LCG) [28], which uses a linear equation to generate a sequence of randomized numbers:

$$X_{n+1} = (aX_n + c)\ (mod\ m)$$

where

- X is the randomized value.
- m is a positive modulus .
- a is the multiplier, $0 < a < m$.
- c is the increment, $0 \leq c < m$.
- X_0 is the initial value.

To prevent the frequency attack, the BLE device only sends the 16 high order bits of X, instead of the whole 32 bits. Although the LCG algorithm is simple, it runs fast and needs little memory space, as well as little power consumption.

4.3 Server Design

The server is responsible for handling all connections to mobile clients and maintaining the database. Figure 4.3 shows the diagram of the server design. The definition of each class is listed below:

- Class **TrackingServer** is the access point of the application, it contains one instance of Class $MobileDeviceManager$.
- Class **MobileDeviceManager** creates and maintains several instances of Class $MobileDeviceAgent$.

- Class **ConnectionRequestHandler** is responsible to allocate a connection for each new instance of $MobileDeviceAgent$ after getting called by $MobileDeviceManager$.
- Class **MySQLDAO** contains the basic CRUD operations in database.
- Class **SQLManager** inherits $MySQLDAO$ and is responsible for updating mobile clients' information in the database.
- Class **MobileDeviceAgent** manages the connection with the corresponding mobile client. It responses requests from the mobile client and updates the status of the mobile client in the database using $SQLManager$.
- Class **MobileDeviceAgentRPCReceiverImpl** listens to the message from the RPC Sender of the mobile client, such as $REQUEST\_ALERTZONES$, then it sends alertzones back.
- Class **MobileDeviceAgentEventReceiverImpl** listens to the message from the Event Sender of the mobile client, such as $UPDATE\_LOCATION$, without sending message back.
- Class **MobileDeviceAgentEventSenderImpl** sends messages to the mobile client, such as $ADD\_ALERTZONE$.

Figure 4.3 Server Design

4.3.1 Database Design

The database manages the information of mobile clients, including their real-time location, the connection session to the server and the paired BLE device status. There are 9 tables in the database, as Figure 4.4 shows :

Figure 4.4 Database Design

- **MobileDeviceProfile**: This table records the information of each mobile client, including the mobile client ID, the last updated location, and so on, as shown in Table 4.1.

Table 4.1 Mobile Device Profile Table

ColumnName	**DataType**
ID	int(11)
DID	int(11)
Latitude	double
Longitude	double
Altitude	double
Timestamp	bigint(20)
TimestampText	text
DomainType	int(11)
DomainCenterX	double
DomainCenterY	double
DomainRadius	double
DomainUpperX	double
DomainUpperY	double
DomainLowerX	double
DomainLowerY	double
ZoneType	int(11)
ZoneRadius	double
ZoneLength	double
ZoneWidth	double
MaxLoad	int(11)
LocationUpdatePolicy	int(11)
UpdatePeriod	int(11)
UpdateDeviation	int(11)
SessionStartTime	bigint(20)
SessionStartTimeText	text
SessionProfileTableName	text

- **SessionProfile_XYZ**: This table stores the connection information for each mobile client, where XYZ is the unique ID of the mobile client. It contains the starting time and ending time, as well as the name of the table which stores the XYZ's location during this connection session, as shown in Table 4.2.

Table 4.2 Session Profile Table

ColumnName	**DataType**
ID	int(11)
StartTime	bigint(20)
StartTimeText	text
EndTime	bigint(20)
EndTimeText	text
SessionTableName	text

- **Location_XYZ_SID** This table records the time and location information for the mobile client XYZ during a connection session, where SID is the unique session ID, as shown in Table 4.3.

Table 4.3 Mobile Client Location Table

ColumnName	**DataType**
ID	int(11)
Latitude	double
Longitude	double
Altitude	double
Time	bigint(20)

- **Alertzone**: This table records the information of alertzones, including the shape of the associated region, the coordinates, and so on, as shown in Table 4.4.

Table 4.4 Alertzone Table

ColumnName	**DataType**
GID	int(11)
CenterX	double
CenterY	double
Radius	double
UpperX	double
UpperY	double
LowerX	double
LowerY	double
URL	varchar(1000)
Trig	varchar(50)
Shape	int(2)
Author	int(11)
Recipient	varchar(1000)
Creation	text
Expiration	text

- **ConsoleProfile**: This table records the information of the Console, such as the number of criminals under tracking, the time interval to retrieve the database, and so on, as shown in Table 4.5.

Table 4.5 Console Profile Table

ColumnName	**DataType**
CID	int(11)
DID	int(11)
CenterX	double
CenterY	double
ZoomLevel	int(11)
DIDList	text
RangeTopLeftX	double
RangeTopLeftY	double
RangeBottomRightX	double
RangeBottomRightY	double
KNN	int(11)
UpdateInterval	int(11)
Inbox	varchar(1000)
Map	char(11)
Type	char(11)
Query	char(11)
FocusNode	char(11)

- **BLEPeriDeviceProfile**: This table records the information of the BLE device, including their ID, paired mobile client ID, the IEEE address, and so on, as shown in Table 4.6.

Table 4.6 BLE Device Table

ColumnName	**DataType**
ID	int(11)
DID	int(11)
IEEEAddress	text
BatteryLevel	int(3)
AccelX	double
AccelY	double
AccelZ	double
BatteryLevelTable	text
AccelValueTable	text
BLESessionTable	text

- **XYZ_BLE_SessionTable**: This table records the connection session between the BLE device and the mobile client, where XYZ is the ID of the BLE device. It includes the starting time and ending time, as shown in Table 4.7.

Table 4.7 BLE Session Table

ColumnName	**DataType**
ID	int(11)
Link_Start_Time	text
Link_End_Time	text

- **XYZ_BLE_BatteryLevel**: This table records the battery level of the BLE device, where XYZ is the ID of the BLE device, as shown in Table 4.8.

Table 4.8 BLE Battery Level Table

ColumnName	DataType
ID	int(11)
Time	text
BatteryLevel	text

- **XYZ_BLE_AccelValue**: This table records the accelerometer values of the BLE device during its connection period with the smartphone, where XYZ is the ID of the BLE device, as shown in Table 4.9.

Table 4.9 BLE Peripheral Accel Value Table

ColumnName	DataType
ID	int(11)
Time	text
X	int(5)
Y	int(5)
Z	int(5)

4.4 Console Design

The Console component is basically a web-based GUI which displays the real-time information of criminals in the web page. In our design, the Console is independent with the mobile client and the server and it only connects to the database. The web application periodically retrieves records from the database, such as the criminals' location and alertzones, and displays them in the map application. It also detects the criminals' location against alertzones and alerts the administrator if the criminal is in any of them.

Moreover, our design supports multiple Console running at the same time. There is the $ConsoleProfile$ table in the database that stores the configuration for each console, such as

the update interval, map type, the amount of criminals being displayed, and so on. The web application also allows one to create/remove alertzones in real time. The administrator could create alertzones in the webpage and save them in the database. The server creates the $ADD\_ALERTZONE$ messages based on the information of the newly created alertzones and sends them to the corresponding mobile clients.

4.5 Message Design

All messages exchanged between the mobile client and the server follow the same format, which is shown in Table 4.10. Figure 4.5 lists all messages.

Table 4.10 Message Format

PacketLength	**MessageID**	**MessageContent**
4 bytes	4 bytes	n bytes

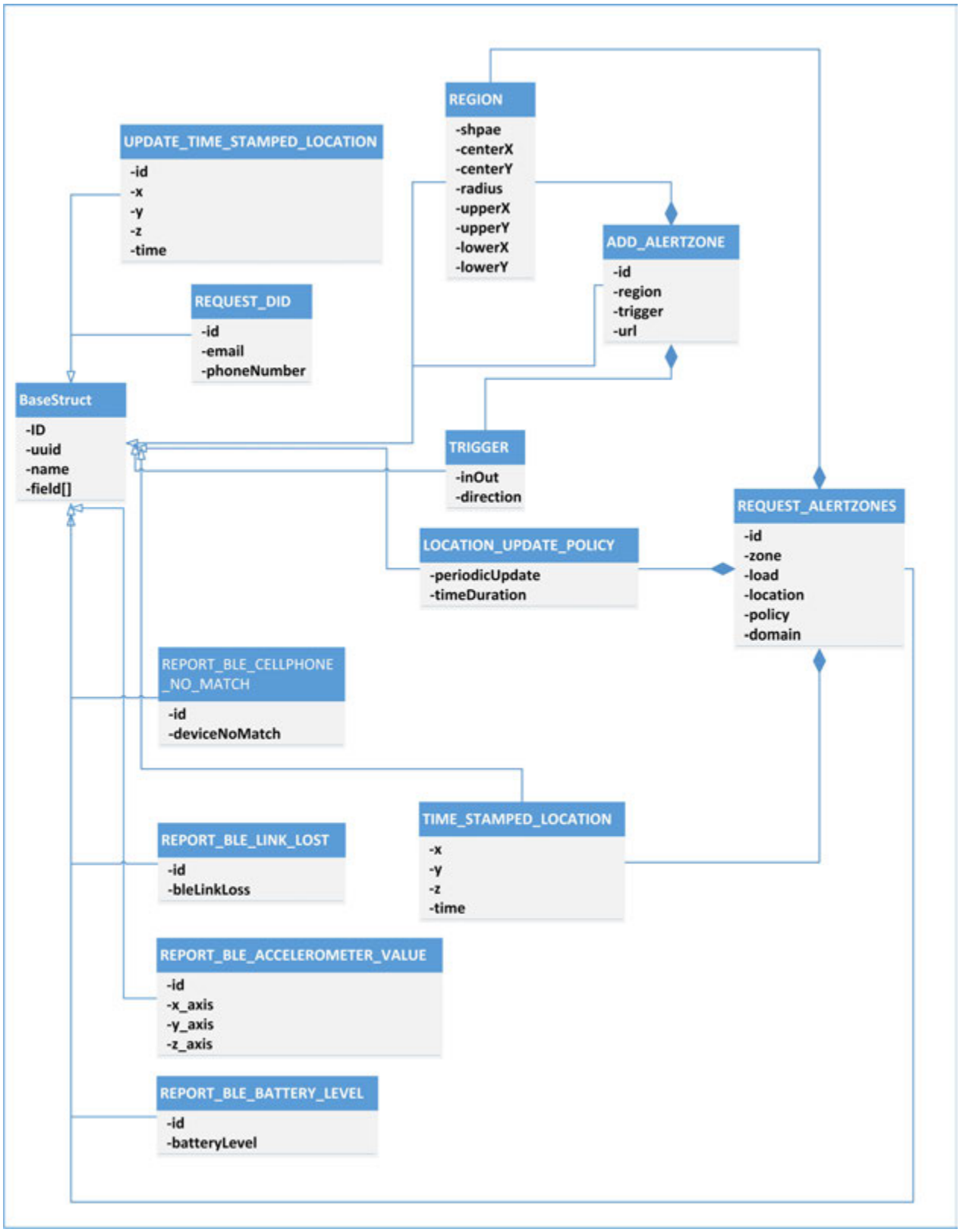

Figure 4.5 Message Design

All messages consist of bytes. The data structures are listed below:

- **TIME_STAMPED_LOCATION** corresponds to Class TimeStampedLocation, which has 32 bytes:

Table 4.11 TIME_STAMPED_LOCATION Structure

FieldName	**FieldLength(bytes)**
x	8
y	8
z	8
time	8

- **REGION** corresponds to Class Region, which has 57 bytes:

Table 4.12 REGION Structure

FieldName	**FieldLength(bytes)**
shape	1
centerX	8
centerY	8
radius	8
upperX	8
upperY	8
lowerX	8
lowerY	8

- **TRIGGER** corresponds to Class Trigger, which has 2 bytes:

Table 4.13 TRIGGER Structure

FieldName	**FieldLength(bytes)**
inOut	1
direction	1

- **LOCATION_UPDATE_POLICY** corresponds to Class LocationUpdatePolicy, which has 9 bytes:

Table 4.14 LOCATION_UPDATE_POLICY Structure

FieldName	**FieldLength(bytes)**
periodicUpdate	1
timeDuration	8

Messages sent by the mobile client are listed below:

- There are two different types of messages from the client, one type of messages needs responses from the server:
  - *REQUEST_DID*: The mobile client requests an unique ID from the server. After processing, the server sends a unique ID back to the mobile client. The message format is listed below:

Table 4.15 REQUEST_DID Message

FieldName	**FieldLength(bytes)**
pkgLength	4
id	4
email	50
phoneNumber	50

– *REQUEST_ALERTZONES*: When the mobile client connects to the server, it sends this message. The server sends alertzones back in response. The message format is listed below:

Table 4.16 REQUEST_ALERTZONES Message

FieldName	**FieldLength(bytes)**
pkgLength	4
id	4
zone	57
load	4
location	32
policy	9
domain	57

- Another type of messages does't have responses from the server:

  – *UPDATE_TIME_STAMPED_LOCATION*: When the mobile client detects its location is in any alertzone, it sends this message to the server. The message format is listed below:

Table 4.17 UPDATE_TIME_STAMPED_LOCATION Message

FieldName	**FieldLength(bytes)**
pkgLength	4
id	4
x	8
y	8
z	8
time	8

  – *REPORT_BLE_LINK_LOST*: When the mobile client detects its connection to

the BLE device is lost, it sends this message to the server. The message format is listed below:

Table 4.18 REPORT_BLE_LINK_LOST Message

FieldName	**FieldLength(bytes)**
pkgLength	4
id	4
bleLinkLoss	1

– *REPORT_BLE_CELLPHONE_NO_MATCH*: When the mobile client detects the OTP from the BLE device is not matched, it sends this message to the server. The message format is listed below:

Table 4.19 REPORT_BLE_CELLPHONE_NO_MATCH Message

FieldName	**FieldLength(bytes)**
pkgLength	4
id	4
deviceNoMatch	1

– *REPORT_BLE_ACCELEROMETER_VALUE*: When the mobile client receives the accelerometer values from the BLE peripheral device, it sends this message to the server. The message format is listed below:

Table 4.20 REPORT_BLE_ACCELEROMETER_VALUE Message

FieldName	**FieldLength(bytes)**
pkgLength	4
id	4
x_axis	1
y_axis	1
z_axis	1

– *REPORT_BLE_BATTERY_LEVEL*: When the mobile client receives the battery level value from the BLE device, it sends this message to the server. The message format is listed below:

Table 4.21 REPORT_BLE_BATTERY_LEVEL Message

FieldName	**FieldLength(bytes)**
pkgLength	4
id	4
batteryLevel	1

Another category of messages is sent by the server, which has one message below:

- *ADD_ALERTZONE*: After a new alertzone is created, the server sends this message to the client. The message format is listed below:

Table 4.22 ADD_ALERTZONE Message

FieldName	**FieldLength(bytes)**
pkgLength	4
id	4
region	57
tirgger	2
url	255

CHAPTER 5. IMPLEMENTATION

5.1 Mobile Client Implementation

The mobile client application is implemented in Objective-C, which runs on the iOS platform. We test the code on Apple's iPhone 4S, which has a built-in GPS receiver and communicates with the server through Verizon 's wireless data service.

When the smartphone is powered on, as Figure 5.1 (a) shows, the application starts to run and displays its current location, including the latitude, the longitude and the altitude. After initialization, the application could display the client's position in a map, as Figure 5.1 (b) shows. The screen also shows alertzones that the client has received. When criminals move into an alertzone, a corresponding warning message will be automatically downloaded and displayed in the browser. The mobile client application can be set to run as a background process such that it could launch the browser to display warning messages as soon as the criminal enters into an alertzone. In Figure 5.1 (c), it shows the connection status with the BLE device. The application displays the OTP from the BLE device as well as the OTP computed by itself. If passwords do not match or the connection is lost, the mobile client sends messages to the server. Moreover, the BLE device could send the accelerometer value to the mobile client, which is used to indicate the motion of the criminal. The battery level value is used to indicate how much power left in the BLE device.

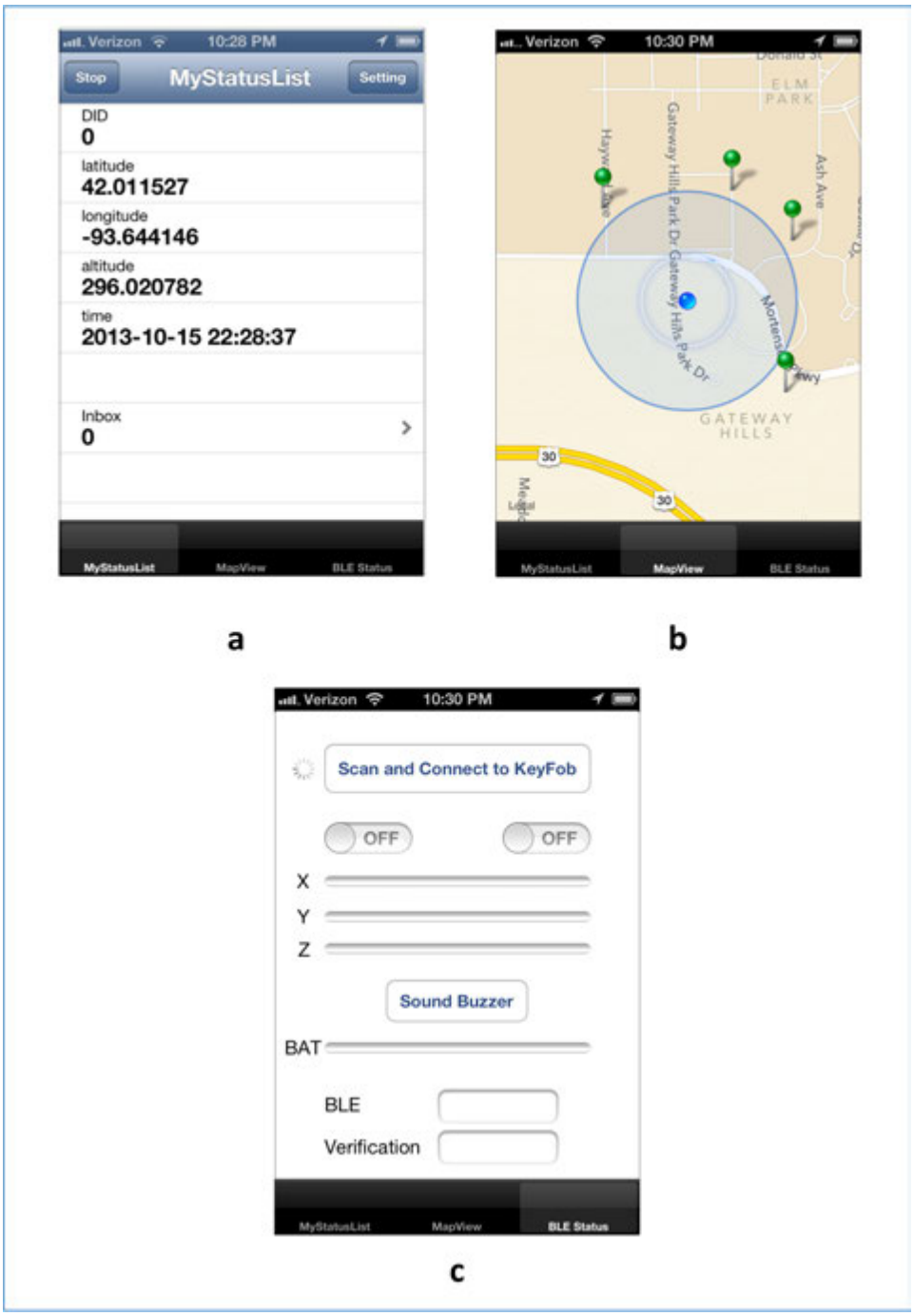

Figure 5.1 Client GUI

5.2 BLE Device Implementation

We use Texas Instrument CC2540 MINI Development Kit [9] to develop the BLE application. Figure 5.2 shows components of CC2540 MINI Development Kit.

- **Keyfob** works as a peripheral device(BLE Slave) and is powered by a single CR2032 coin cell battery.
- **USB Dongle** connects to a Windows PC USB port and emulates as a central device (BLE Master).
- **CC Debugger** is used to program the Keyfob as well as the USB Dongle.

Figure 5.2 CC 2540 Mini Development Kit

We have developed a new profile *BLE Tracker* in our system. Instead of PC machine, we built an application running on the smartphone as a BLE master. Figure 5.3 shows the Keyfob as the BLE Slave device and iPhone 4S as the BLE master device.

Figure 5.3 CC 2540 Keyfob and iPhone 4S

5.3 Server Implementation

The server application is implemented using Java programming language, which indexes the alertzones and stores other information such as criminals' profiles in a database managed by MySQL 5.5.

5.4 Console Implementation

Figure 5.4 shows the Console, which displays criminals' information and all alertzones on the webpage. The console periodically reads criminals' records in database, such as the last updated location, the battery level value of the BLE device, and displays them on the map application. Moreover, the console allows one to create/remove alertzones through regular web access. To create an alertzone, the administrator first creates an HTML page, which may contain any content (including text, audio, and video clips) that is supported by client browsers, and then associates the page with a bounding box. The bounding box can be defined by giving two points or clicking two points on the map displayed on the screen, as figure 5.5 shows. Once the alertzone is created, it is saved in the database.

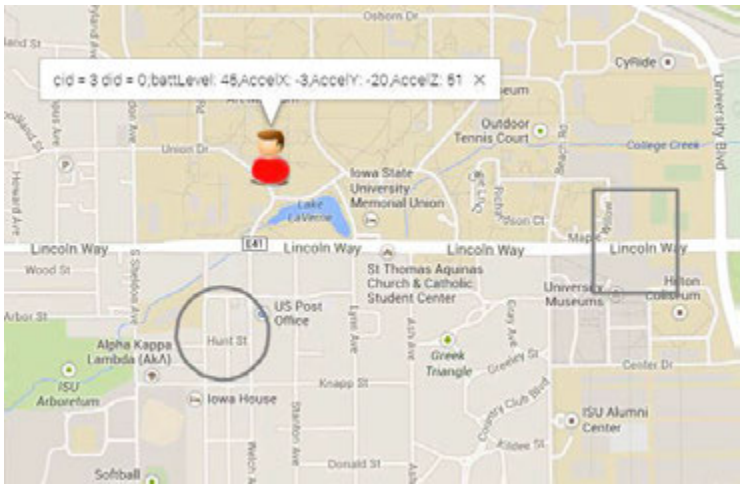

Figure 5.4 Console

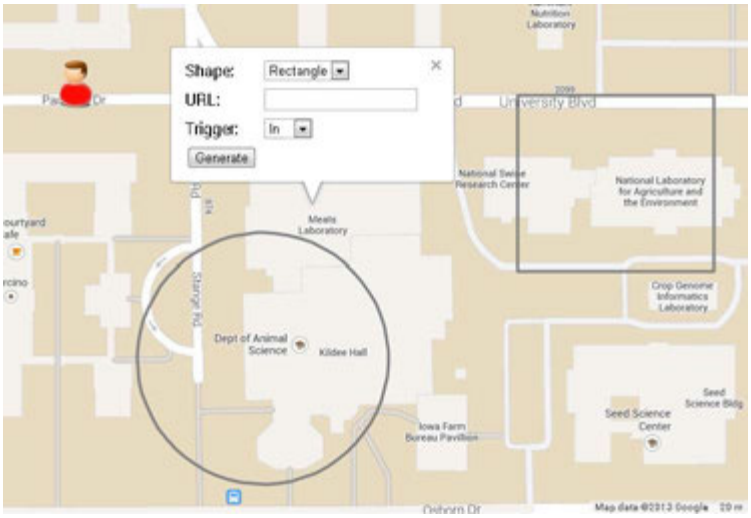

Figure 5.5 Alertzone

CHAPTER 6. CONCLUSION

We have presented the design and implementation of a large-scale, low-cost and real-time criminal tracking system. Our system consists of a central server and smartphones, each paired with a Bluetooth device that is bundled on the criminal's body part. The smartphone connects to the Bluetooth device using BLE technology. If the smartphone can connect to its corresponding BLE device, it indicates the criminal is in a specific range of the smartphone. The smartphone monitors the connection status with the BLE device and sends a message to the server when the connection is lost. Besides that, the smartphone monitors its own location against alertzones, which are geographical regions that the criminal is not allowed to access. Instead of sending location to the server continuously, the smartphone checks its location and reports to the central server only when needed, e.g., losing connection to the BLE device, moving into or out of an alertzone. By reducing location updates, our system allows a same server to track a large number of mobile devices. Moreover, our system incurs very low cost in production since it uses only off-shelf hardware components and the regular cellular wireless infrastructures.

BIBLIOGRAPHY

[1] Bluetooth low energy. http://www.bluetooth.org, 2010.

[2] Nike+. http://nikeplus.nike.com/plus/, 2010.

[3] Immediate alert service. https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=239390, 2011.

[4] Link loss service. https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=239391, 2011.

[5] Proximity profile. https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=239392, 2011.

[6] 3m one piece gps tracking system. http://solutions.3m.com/wps/portal/3M/en_US/ElectronicMonitoring/Home/ProductsServices/OurProducts/GPSTracking/OnePieceGPSTrackingSystem/, 2013.

[7] Bluetoothdeveloper. https://developer.bluetooth.org/TechnologyOverview/Pages/BLE.aspx, 2013.

[8] Blutracker. http://blutracker.com/, 2013.

[9] Ti cc2540 mini development kit. http://www.ti.com/tool/cc2540dk-mini, 2013.

[10] Lauri Aalto, Nicklas Göthlin, Jani Korhonen, and Timo Ojala. Bluetooth and wap push based location-aware mobile advertising system. In *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 49–58. ACM, 2004.

[11] Ganesh Ananthanarayanan, Maya Haridasan, Iqbal Mohomed, Doug Terry, and Chandramohan A Thekkath. Startrack: a framework for enabling track-based applications. In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, pages 207–220. ACM, 2009.

[12] P. Bahl and V.N. Padmanabhan. Radar: an in-building rf-based user location and tracking system. 2:775–784 vol.2, 2000.

[13] Maged Boulos, Artur Rocha, Angelo Martins, Manuel Vicente, Armin Bolz, Robert Feld, Igor Tchoudovski, Martin Braecklein, John Nelson, Gearoid O Laighin, Claudio Sdogati, Francesca Cesaroni, Marco Antomarini, Angela Jobes, and Mark Kinirons. Caalyx: a new generation of location-based services in healthcare. *International Journal of Health Geographics*, 6(1):9, 2007.

[14] Ying Cai and Toby Xu. Design, analysis, and implementation of a large-scale real-time location-based information sharing system. In *Proceedings of the 6th international conference on Mobile systems, applications, and services*, pages 106–117. ACM, 2008.

[15] Chih-Ming Chen. Intelligent location-based mobile news service system with automatic news summarization. *Expert Systems with Applications*, 37(9):6651–6662, 2010.

[16] S. Mascetti C. Bettini D. Freni, C. R. Vicente and C. S. Jensen. Preserving location and absence privacy in geo-social networks. In *ACM CIKM*, 2010.

[17] Subhankar Dhar and Upkar Varshney. Challenges and business models for mobile location-based services and advertising. *Communications of the ACM*, 54(5):121–128, 2011.

[18] Thomas D'Roza and George Bilchev. An overview of location-based services. *BT Technology Journal*, 21(1):20–27, 2003.

[19] Gabriel Ghinita, Panos Kalnis, and Spiros Skiadopoulos. Prive: anonymous location-based queries in distributed mobile systems. In *Proceedings of the 16th international conference on World Wide Web*, pages 371–380. ACM, 2007.

[20] W. G. Griswold, P. Shanahan, S. W. Brown, R. S. Boyer, M. Ratto, B. R. Shapiro, and T. M. Truong. Activecampus: Experiments in community-oriented ubiquitous computing. *IEEE Computer*, 37(10):73–81, 2004.

[21] Marco Gruteser and Dirk Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 31–42. ACM, 2003.

[22] Neil Haller, Craig Metz, Phil Nesser, and Mike Straw. A one-time password system. Technical report, RFC 1938, May, 1996.

[23] M. F. Mokbel J. Bao and C.-Y. Chow. Geofeed: A locationaware news feed system. In *IEEE ICDE*, 2012.

[24] Panos Kalnis, Gabriel Ghinita, Kyriakos Mouratidis, and Dimitris Papadias. Preventing location-based identity inference in anonymous spatial queries. *Knowledge and Data Engineering, IEEE Transactions on*, 19(12):1719–1733, 2007.

[25] A. Khoshgozaran and C. Shahabi. Private buddy search: Enabling private spatial queries in social networks. In *IEEE SIN*, 2009.

[26] H. Kido, Y. Yanagisawa, and T. Satoh. An anonymous communication technique using dummies for location-based services. In *Pervasive Services, 2005. ICPS '05. Proceedings. International Conference on*, pages 88–97, 2005.

[27] Aleksandra Kovacevic, Nicolas Liebau, and Ralf Steinmetz. Globase. kom-a p2p overlay for fully retrievable location-based search. In *Peer-to-Peer Computing, 2007. P2P 2007. Seventh IEEE International Conference on*, pages 87–96. IEEE, 2007.

[28] P. L'Ecuyer. Efficient and portable combined random number generators. *Communications of the ACM*, 31(6):742–751, 1988.

[29] Mohamed F Mokbel, Chi-Yin Chow, and Walid G Aref. The new casper: query processing for location services without compromising privacy. In *Proceedings of the 32nd international conference on Very large data bases*, pages 763–774. VLDB Endowment, 2006.

[30] D MRaihi, M Bellare, F Hoornaert, D Naccache, and O Ranen. Hotp: An hmac-based one-time password algorithm. *The Internet Society, Network Working Group. RFC4226*, 2005.

[31] Alf Helge Omre. Bluetooth low energy: wireless connectivity for medical monitoring. *Journal of diabetes science and technology*, 4(2):457, 2010.

[32] P. Persson and P. Fagerberg. Geonotes: A real-use study of a public location-aware community system. In *Technical Report SICS-T2002/27-SE, SICS, University of Goteborg*, Sweden, 2002.

[33] Omer Rashid, Ian Mullins, Paul Coulton, and Reuben Edwards. Extending cyberspace: location based games using cellular phones. *Computers in Entertainment (CIE)*, 4(1):4, 2006.

[34] Gerhard Reitmayr and Tom W Drummond. Going out: robust model-based tracking for outdoor augmented reality. In *Mixed and Augmented Reality, 2006. ISMAR 2006. IEEE/ACM International Symposium on*, pages 109–118. IEEE, 2006.

[35] Takasuke Tsuji and Akihiro Shimizu. An impersonation attack on one-time password authentication protocol ospa. *IEICE Transactions on Communications*, 86(7):2182–2185, 2003.

[36] YEH Tzu-Chang, SHEN Hsiao-Yun, and Jing-Jang Hwang. A secure one-time password authentication scheme using smart cards. *IEICE Transactions on Communications*, 85(11):2515–2518, 2002.

[37] Alan Wong, Mark Dawkins, Gabriele Devita, Nick Kasparidis, Andreas Katsiamis, Oliver King, Franco Lauria, Johannes Schiff, and Alison Burdett. A 1v 5ma multimode ieee 802.15. 6/bluetooth low-energy wban transceiver for biotelemetry applications. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International*, pages 300–302. IEEE, 2012.

[38] X. Xie Y. Zheng and W.-Y. Ma. Geolife: A collaborative social networking service among user, location and trajectory. *IEEE Data Engineering Bulletin*, 33(2):32–39, 2010.

[39] Seiji Yokoji, Katsumi Takahashi, and Nobuyuki Miura. Kokono search: A location based search engine. In *WWW Posters*, 2001.

[40] Jun Zhang, Manli Zhu, Dimitris Papadias, Yufei Tao, and Dik Lun Lee. Location-based spatial queries. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 443–454. ACM, 2003.