پروژه باریم

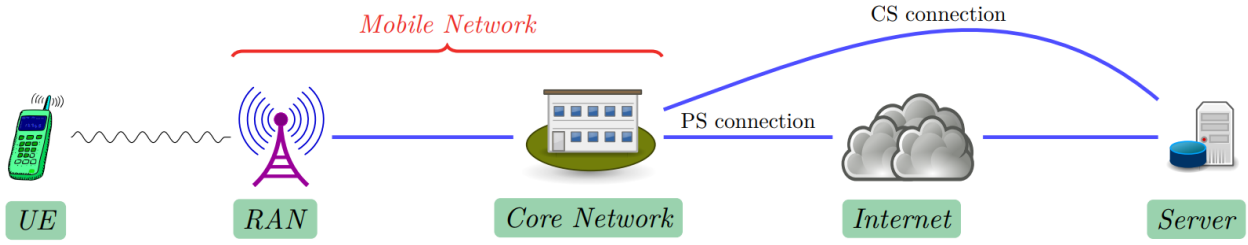
درس آشنایی با شبکه های تلفن همراه

غزل عربعلی - ۹۷۵۲۱۳۹۶، بهاره کاوسی نژاد - ۹۹۴۳۱۲۱۷

آخرین ویرایش: ۱۶ تیر ۱۴۰۳ در ساعت ۲ و ۳۶ دقیقه

۱ شرح پروژه

گسترش روزافزون شبکه های تلفن همراه به ویژه شبکه های نسل چهار و پنج، موجب شده است که این شبکه ها به عنوان بزرگترین شبکه دسترسی [۱] ، برای دستیابی به خدمات اینترنت بشمار آید. پرواضح است که در این بین، مساله امنیت [۲] برنامه های کاربردی [۳] و ساخت یک برنامه کاربردی با یک ارتباط امن، یکی از مهم ترین مسایل این حوزه خواهد بود. گرچه باید به این نکته توجه داشت که امنیت در یک ارتباط از طریق شبکه های تلفن همراه را، نباید تنها به مساله امنیت در دو سوی مشتری [۴] و خدمت گزار [۵] تقلیل داد؛ بلکه در جای جای این ارتباط، ما می توانیم با حملات متعددی مواجه شویم، که می تواند محرمانگی [۶] ، یکپارچگی [۷] و حریم خصوصی [۸] ما را هدف قرار دهد. شکل ۱.۱ نمایی از ارتباط یک مشتری با خدمت گزار را در بستر های مختلف از طریق شبکه های تلفن همراه به زیبایی نشان می دهد.



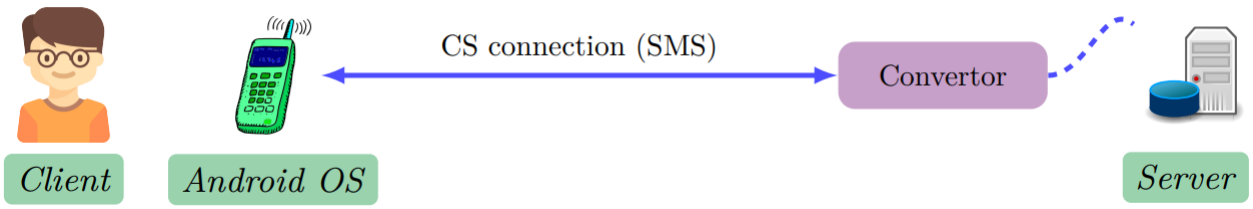
شکل ۱.۱: ارتباط بین مشتری با خدمت گزار از طریق شبکه های تلفن همراه بر روی بسترهای مختلف

در مساله پیش رو، فرض می کنیم که یک برنامه کاربردی داریم، که توسط برنامه UE می شود. UE از دیدگاه ما هر ابزاری است که توسط آن بتوان به شبکه های تلفن همراه متصل شد. UE می تواند گوشی تلفن همراه، تبلت و یا حتی هر شی در IoT [۹] باشد. گرچه در این پروژه، ما تنها بر روی گوشی های تلفن همراه و تبلت ها تمرکز خواهیم کرد.

برنامه کاربردی UE قرار است تا از طریق بسترهای موجود در شبکه های تلفن همراه به یک خدمت گزار مشخص متصل شوند و با آن تبادل اطلاعات داشته باشند. در این جا ما دو راه کار برای اتصال به خدمت گزار داریم. در راه کار نخست و بدیهی ترین شیوه، ما از طریق بستر اینترنت با خدمت گزار به تبادل داده مبادرت می ورزیم. ما اصطلاحا به این شیوه اتصال از طریق PS [۱۰] می گوییم.

[۱] Access Network
[۲] Security
[۳] Application
[۴] Client
[۵] Server
[۶] Confidentiality
[۷] Integrity
[۸] Privacy
[۹] Internet of Things
[۱۰] Packet-switched

بالاخره باید پذیرفت که دنیای اینترنت، مخاطرات پیدا و پنهان فراوانی دارد. اتصال از طریق خدمات CS [11] [12] نظیر تماس [13] و SMS [14] ، می تواند راه فراری از مخاطرات دنیای اینترنت باشد. در این پروژه، ما فرض می کنیم که اتصال مشتری به خدمت گزار را از طریق SMS ، برقرار خواهد شد.



شکل ۲.۱: معماری سطح بالای سامانه

در این جا برای سادگی فرض کنید که دو گوشی داریم. گوشی سمت مشتری و گوشی که ما به عنوان خدمت گزار از آن استفاده می کنیم. در سمت خدمت گزار (که در حقیقت یک گوشی معمولی است)، یک برنامه Android ای با کارکرد Backend نصب می شود. مشتری از طریق SMS فرمان ها را به سمت مقابل (خدمت گزار) ارسال می کند. مشتری می بایست به صورت مداوم اطلاعات مربوط به توان دریافتی و تکنولوژی سلول خدمتگزار [15] و مکان دریافت این اطلاعات را در صورتی که توان از یک سطح آستانه معین پایین بیاید در قالب یک پیام برای خدمت گزار ارسال کند. در این سامانه می بایست به نکات زیر دقت کنید:

- برنامه سمت خدمت گزار می بایست به صورت یک سرویس در Android باشد، البته برای مدیریت و پیکربندی آن می توان یک برنامه UI دار نیز داشته باشیم.
- فرض کنید که همگان پروتکل ارتباطی شما را که مبتنی بر SMS است می دانند. اگر اجازه دهیم SMS از هر شماره ای به سمت خدمت گزار ارسال شود، رویه ای در نظر بگیرید که جلوی دسترسی های غیرمجاز را بگیرد. شاید یک رویه ساده، ارسال یک رمز عبور [16] در ابتدای SMS است. تلاش کنید تا رویه های بهتری برای حل این چالش در نظر بگیرید.
- در هنگامی که مشتری درخواست خود را برای خدمت گزار ارسال می کند، خدمت گزار درخواست را می بایست اجرا کند و پاسخ را در یک SMS جداگانه برای مشتری ارسال کند. دقت کنید اگر بتوانید باید تشخیص بدهید که Delivery بر می گردد یا خیر. اگر برنگشت باید پیام را دوباره ارسال کنیم.
- در پیام ارسالی از سوی مشتری، می بایست مکان اندازه گیری، مقداری اندازه گیری و اطلاعات سلولی که به آن متصل است را ارسال کند.
- پروتکل ارتباطی را باید به صورت کامل مستند بکنید، و باید مبتنی بر پروتکل SMPP [17] باشد.

[11] Service
[12] Circuit-switched
[13] Call
[14] Short Message Service
[15] Serving Cell
[16] Password
[17] Short Message Peer-to-Peer

۲ توسعه برنامه ارتباطی امن مبتنی بر SMS

۱.۲ مقدمه

با گسترش روزافزون شبکه‌های تلفن همراه، به ویژه شبکه‌های نسل چهار و پنج، این شبکه‌ها به عنوان بزرگترین شبکه‌های دسترسی به خدمات اینترنت شناخته می‌شوند. در این میان، امنیت برنامه‌های کاربردی و ساخت یک ارتباط امن از مهم‌ترین مسائل است. امنیت در ارتباطات شبکه‌های تلفن همراه نباید فقط به امنیت مشتری و خدمت‌گزار محدود شود، بلکه در تمام نقاط این ارتباط ممکن است با حملات متعددی مواجه شویم که می‌تواند محرمانگی، یکپارچگی و حریم خصوصی ما را هدف قرار دهد.

در این پروژه، ما به توسعه یک برنامه کاربردی می‌پردازیم که از طریق بسترهای موجود در شبکه‌های تلفن همراه به یک خدمت‌گزار مشخص متصل شده و با آن تبادل اطلاعات دارد. در اینجا دو روش برای اتصال به خدمت‌گزار وجود دارد:

- از طریق بستر اینترنت
- از طریق خدمات مدارگرا (CS) نظیر تماس و پیامک (SMS)

ما در این پروژه، اتصال از طریق SMS را انتخاب کرده‌ایم.

این سامانه شامل دو گوشی تلفن همراه است: گوشی مشتری و گوشی خدمت‌گزار. برنامه اندرویدی با کارکرد Backend بر روی گوشی خدمت‌گزار نصب می‌شود و مشتری از طریق SMS فرمان‌ها را ارسال می‌کند. مشتری باید اطلاعات مربوط به توان دریافتی، تکنولوژی سلول خدمت‌گزار و مکان دریافت این اطلاعات را در قالب پیام به خدمت‌گزار ارسال کند. برنامه خدمت‌گزار به صورت یک سرویس در Android عمل می‌کند و می‌تواند برای مدیریت و پیکربندی آن یک رابط کاربری (UI) نیز داشته باشد.

۱.۱.۲ کلیات کار انجام شده

در این پروژه، مراحل زیر برای توسعه و پیاده‌سازی برنامه ارتباطی امن مبتنی بر SMS انجام شده است:

تحلیل نیازمندی‌ها و طراحی سیستم:

- شناسایی نیازمندی‌های امنیتی و عملکردی سیستم.
- طراحی معماری سامانه شامل مشتری و خدمت‌گزار.
- انتخاب پروتکل SMPP برای ارسال پیام‌های SMS .

توسعه برنامه‌های اندرویدی:

- توسعه برنامه مشتری برای ارسال اطلاعات از طریق SMS .
- توسعه برنامه خدمت‌گزار برای دریافت و پردازش پیام‌ها.
- پیاده‌سازی سرویس‌های Backend در خدمت‌گزار.

مدیریت امنیت ارتباطات:

- پیاده‌سازی روش‌های امنیتی برای جلوگیری از دسترسی‌های غیرمجاز.
- ارسال رمز عبور در ابتدای پیام‌های SMS .
- مدیریت مجوزهای دسترسی در برنامه‌های اندرویدی.

آزمایش و ارزیابی:

- تست برنامه‌ها در شرایط مختلف شبکه.
- ارزیابی عملکرد و امنیت سامانه.
- بهبود و بهینه‌سازی برنامه‌ها براساس نتایج آزمایش‌ها.

مستندسازی:

- مستندسازی جزئیات کدها و متدهای استفاده شده.
- ارائه راهنمای استفاده و پیکربندی سامانه.

این فصل شامل توضیحات کامل مراحل انجام شده و جزئیات فنی پیاده‌سازی می‌باشد.

۳ راهنمای استفاده و پیکربندی سامانه

۱.۳ روش اجرای پروژه

۲.۳ روش استفاده شده برای جلوگیری از استفاده غیرمجاز

کد زیر در کلاس SettingActivation قرار دارد:

```
// Initialize EditText fields with stored or default valuesval hostEditText =
    findViewById<EditText>(R.id.editTextHost)
hostEditText.setText(sharedPref.getString("host", "172.20.10.13"))
val portEditText = findViewById<EditText>(R.id.editTextPort)portEditText.
    setText(sharedPref.getInt("port", 9500).toString())
val usernameEditText = findViewById<EditText>(R.id.editTextUsername)
usernameEditText.setText(sharedPref.getString("username", "smppuser"))
val passwordEditText = findViewById<EditText>(R.id.editTextPassword)
    passwordEditText.setText(sharedPref.getString("password", "Ouo5nQM8"))
val keyEditText = findViewById<EditText>(R.id.editKey)
keyEditText.setText(sharedPref.getString("key", ""))
```

در این کد، تمامی مقادیر ورودی مربوط به اتصال به سرور، به جز key ، به صورت پیش‌فرض (شکل ۲.۳) تنظیم شده‌اند تا امکان اتصال به اطلاعات سروری که Ozeki در اختیار ما قرار داده است، فراهم شود. این مقادیر شامل host ، port ، username و password می‌شوند که از تنظیمات پیش‌فرض یا مقادیر ذخیره‌شده در shared preferences خوانده می‌شوند. برای جلوگیری از دسترسی‌ها و اتصال‌های غیرمجاز، یک متغیر به نام key تعریف کرده‌ایم. اگر مشتری در سمت اپلیکیشن کلاینت، این مقدار را به درستی وارد کند، امکان ورود به جلسه (session) و ارسال پیام به سرور را خواهد داشت. در کد زیر به طور مشخص تر این بخش توضیح داده می شود.

```
val host = hostEditText.text.toString()val port = portEditText.text.toString()
val username = usernameEditText.text.toString()val password = passwordEditText.
    text.toString()
val key = keyEditText.text.toString()val fixedKey = "1Q79Babrt983JHGS09312nsKJ!
    "
```

```
// Check if the entered key matches the fixed key
if (key != fixedKey) {    // Show toast message for incorrect key
    Toast.makeText(this, "The entered key is incorrect.", Toast.LENGTH_SHORT).
        show()    keyEditText.text.clear()
} else {    // Save configuration to shared preferences
    saveConfig(host, port, username, password, key)
    // Navigate to MainActivity upon successful save    val intent = Intent(
        this, MainActivity::class.java)
    startActivity(intent)}
```

در کد بالا در بخش

```
if (key != fixedKey)
```

اگر کاربر کدی را وارد کند که با کد ثابت fixedKey مطابقت نداشته باشد، دسترسی به تنظیمات و ادامه فرآیند مسیریابی به MainActivity متوقف خواهد شد. به جای آن، یک پیام Toast نمایش داده می‌شود (شکل ۳.۳) که به کاربر اعلام می‌کند که کد وارد شده اشتباه است. در صورت مطابقت درست کد ورودی با کد ثابت (شکل ۴.۳) fixedKey ، تنظیمات ورودی (مانند host ، port ، username و password) در shared preferences ذخیره می‌شوند و سپس کاربر به MainActivity منتقل می‌شود.

۳.۳ اطلاعات دریافتی از سلول

```
for (cellInfo in cellInfoList) {
            when (cellInfo) {
                is CellInfoGsm -> {
                    val cellIdentityGsm = cellInfo.cellIdentity
                    val cellSignalStrengthGsm = cellInfo.cellSignalStrength
                    val info = "0:${cellIdentityGsm.cid},${
                        cellSignalStrengthGsm.dbm};"
                    cellInfoStr.append(info)
                    Log.i(TAG, info)
                }
                is CellInfoCdma -> {
                    val cellIdentityCdma = cellInfo.cellIdentity
                    val cellSignalStrengthCdma = cellInfo.cellSignalStrength
                    val info = "1:${cellIdentityCdma.basestationId},${
                        cellSignalStrengthCdma.dbm};"
                    cellInfoStr.append(info)
                    Log.i(TAG, info)
                }
                is CellInfoLte -> {
                    val cellIdentityLte = cellInfo.cellIdentity
                    val cellSignalStrengthLte = cellInfo.cellSignalStrength
                    val info = "2:${cellIdentityLte.ci},${cellSignalStrengthLte
                        .dbm};"
```

```
                    cellInfoStr.append(info)
                    Log.i(TAG, info)
                }
                is CellInfoWcdma -> {
                    val cellIdentityWcdma = cellInfo.cellIdentity
                    val cellSignalStrengthWcdma = cellInfo.cellSignalStrength
                    val info = "3:${cellIdentityWcdma.cid},${
                        cellSignalStrengthWcdma.dbm};"
                    cellInfoStr.append(info)
                    Log.i(TAG, info)
                }
                else -> {
                    val info = "Unknown Cell Type\n"
                    cellInfoStr.append(info)
                    Log.i(TAG, info)
                }
            }
        }
```

الگوی enumeration که در این بخش از کد فایل LocationAndCellInfo.kt استفاده شده، یک رویکرد کارآمد برای شناسایی و مدیریت انواع مختلف سلول‌های شبکه مخابراتی است. در این الگو، انواع مختلف سلول‌ها مانند GSM ، CDMA ، LTE و WCDMA با استفاده از اعداد صحیح کوتاه نمایش داده می‌شوند. این عدد بیانگر نوع سلول است و همراه با اطلاعات دیگری نظیر شناسه سلول و قدرت سیگنال به سرور ارسال می‌شود.

۱.۳.۳ نحوه عملکرد الگوی enumeration :

CellInfoGsm: در صورتی که cellInfo یک نمونه از CellInfoGsm باشد، از اطلاعات cellIdentity و cellSignalStrength آن استفاده می‌شود. مقدار ۰ به عنوان نوع سلول GSM مشخص می‌شود و اطلاعات به صورت "۰:شناسه_سلول،قدرت_سیگنال؛" به سرور ارسال می‌شود.

CellInfoCdma: اگر cellInfo یک نمونه از CellInfoCdma باشد، از اطلاعات cellIdentity و cellSignalStrength آن استفاده می‌شود. مقدار ۱ به عنوان نوع سلول CDMA مشخص می‌شود و اطلاعات به صورت "۱:شناسه_سلول،قدرت_سیگنال؛" به سرور ارسال می‌شود.

CellInfoLte: در صورتی که cellInfo یک نمونه از CellInfoLte باشد، از اطلاعات cellIdentity و cellSignalStrength آن استفاده می‌شود. مقدار ۲ به عنوان نوع سلول LTE مشخص می‌شود و اطلاعات به صورت "۲:شناسه_سلول،قدرت_سیگنال؛" به سرور ارسال می‌شود.

CellInfoWcdma: اگر cellInfo یک نمونه از CellInfoWcdma باشد، از اطلاعات cellIdentity و cellSignalStrength آن استفاده می‌شود. مقدار ۳ به عنوان نوع سلول WCDMA مشخص می‌شود و اطلاعات به صورت "۳:شناسه_سلول،قدرت_سیگنال؛" به سرور ارسال می‌شود.

Unknown Cell Type:

اگر نوع cellInfo تشخیص داده نشود، یک پیام " Unknown Cell Type " به cellInfoStr اضافه می‌شود و به سرور ارسال می‌شود.

مزایای استفاده از الگوی enumeration :

- **کد کوتاه‌تر و خواناتر:** با استفاده از این الگو، کد شما کوتاه‌تر می‌شود و قابلیت خواندن و درک آن برای برنامه‌نویسان دیگر نیز آسان‌تر است.
- **کاهش حجم داده:** ارسال اعداد کوتاه برای نوع سلول‌ها به جای استفاده از رشته‌های طولانی، باعث کاهش حجم داده ارسالی می‌شود که به بهینگی در استفاده از پهنای باند و منابع شبکه کمک می‌کند.
- **سهولت در تشخیص:** با استفاده از اعداد، سریع‌تر می‌توان نوع سلول را تشخیص داد و فرآیندهای بعدی را مدیریت کرد.

این الگوی enumeration به شما کمک می‌کند تا به طور کارآمدتر و سازمان‌یافته‌تر انواع مختلف سلول‌های شبکه را در برنامه خود شناسایی و مدیریت کنید.

از این روش برای ایجاد یک لایه اضافی امنیتی استفاده می‌شود، به طوری که تنها کاربرانی که دارای کد دسترسی صحیح هستند، به اطلاعات حساس دسترسی داشته باشند.

در شکل ۷.۳ لاگ های گرفته شده نشان داده شده اند. توجه کنید همان طور که بیان شد در SMS داده ها به صورت enum شده قرار می گیرند زیرا با محدودیت ۲۵۵ بایت در ارسال پیام مواجه هستیم. همچنین در شکل ۸.۳ دریافت مکان در لاگ نشان داده شده است. پارامترهای مربوطه به شرح زیر می باشند:

در سیستم‌های مخابراتی، " Identity Cell " یا " CI " به شناسه‌ای منحصربه‌فرد اشاره دارد که برای شناسایی یک سلول خاص در شبکه مخابراتی استفاده می‌شود. CI معمولاً به عنوان بخشی از اطلاعات مکان (Location Information) استفاده می‌شود و شامل پارامترهای زیر است:

۱. Code Country Mobile : (MCC)

- کدی که کشور محل فعالیت شبکه را مشخص می‌کند. MCC یک کد سه‌رقمی است.

۲. Code Network Mobile : (MNC)

- کدی که شبکه مخابراتی خاص در یک کشور را مشخص می‌کند. MNC معمولاً یک کد دو یا سه‌رقمی است.

۳. Code Area Location : (LAC)

- کدی که یک منطقه جغرافیایی خاص در شبکه را شناسایی می‌کند. هر LAC مجموعه‌ای از سلول‌ها را در بر می‌گیرد.

۴. Identity Cell : (CI)

- کدی که یک سلول خاص در داخل LAC را شناسایی می‌کند. CI به صورت یک عدد منحصر به فرد برای هر سلول در نظر گرفته می‌شود.

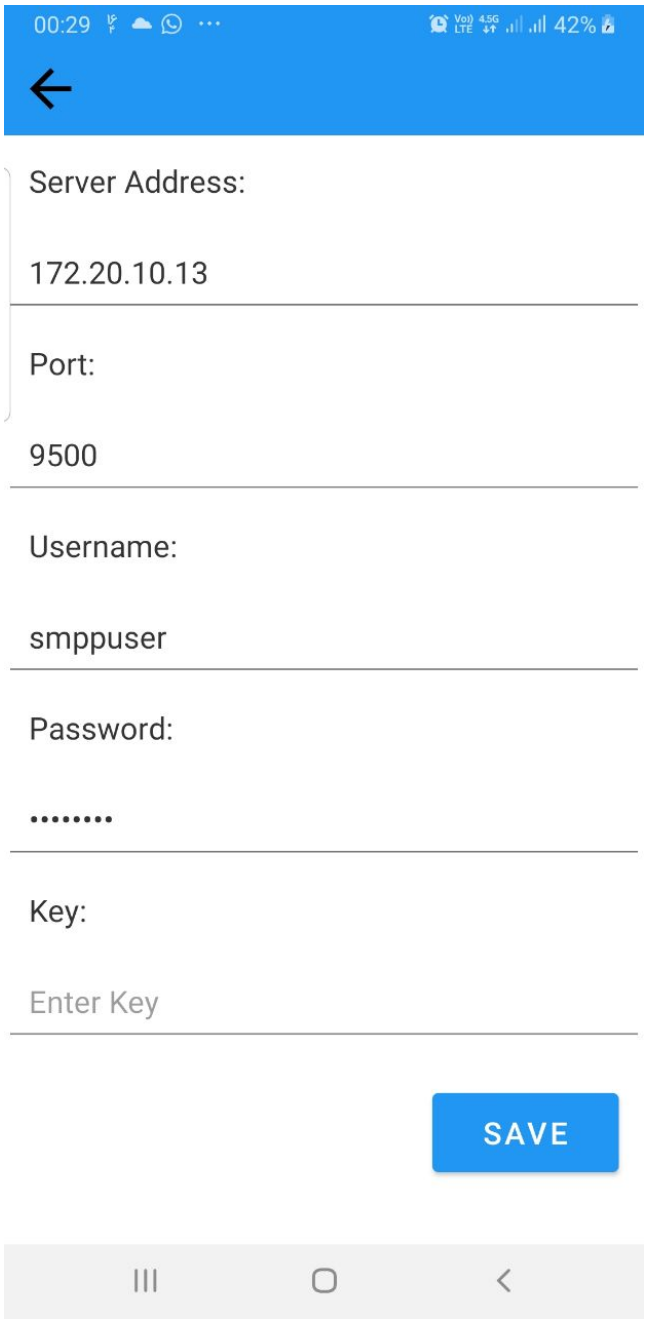
۵. Code Area Tracking (TAC) (در شبکه‌های : LTE)

- مشابه LAC، ولی در شبکه‌های LTE استفاده می‌شود.

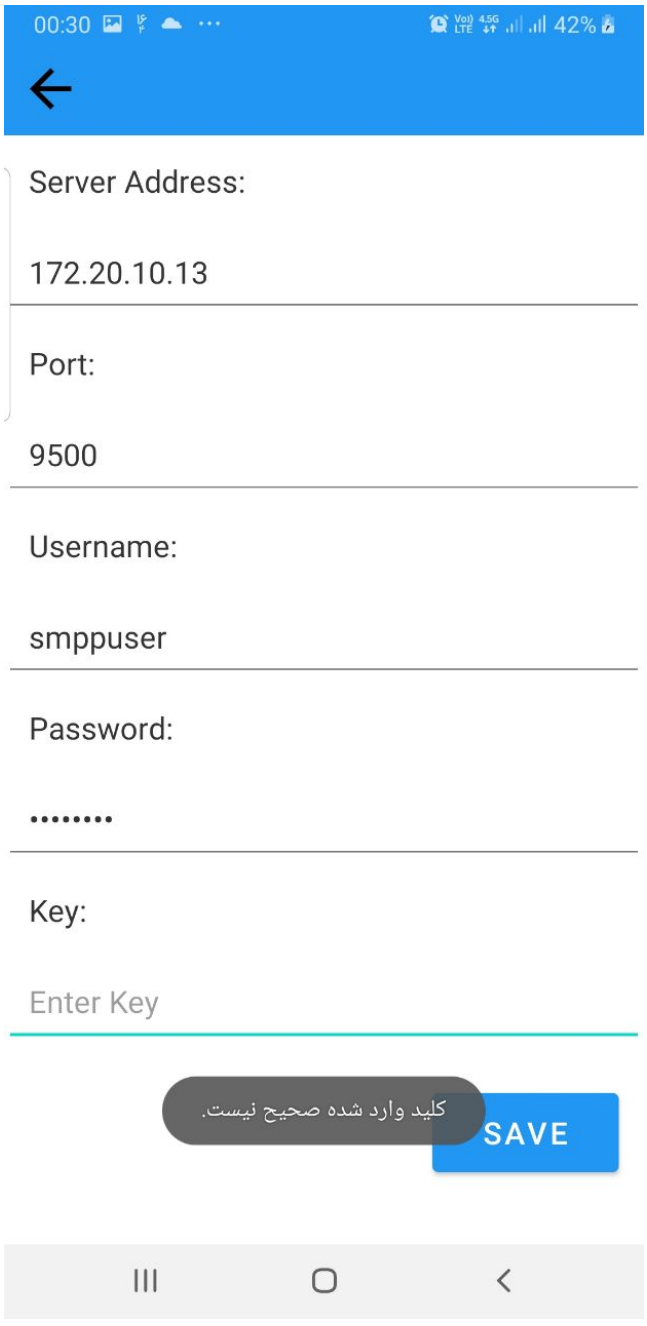
ترکیب MCC، MNC، LAC (یا TAC) و CI یک شناسه منحصربه‌فرد را برای یک سلول در یک شبکه مخابراتی تشکیل می‌دهد.

به طور خلاصه، پارامترهای لوکیشن شامل موارد زیر هستند: - MCC: کد کشور - MNC: کد شبکه- LAC: کد ناحیه مکانی - CI: شناسه سلول- TAC: کد ناحیه ردیابی (در شبکه‌های LTE) این پارامترها به اپراتورها و دستگاه‌های موبایل کمک می‌کنند تا مکان سلول‌ها را مشخص کرده و مدیریت اتصالات شبکه را انجام دهند.

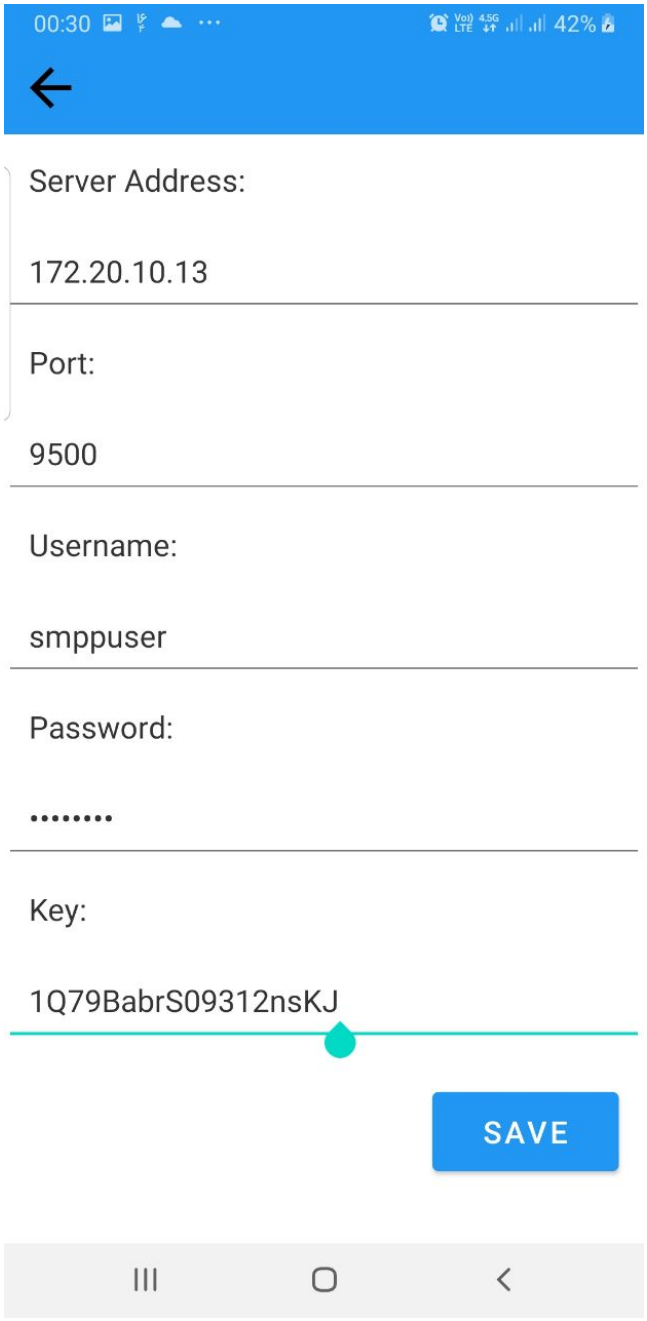
شکل ۱.۳: برنامه ozeki



شکل ۲.۳: قرار گرفتن مقادیر پیشفرض



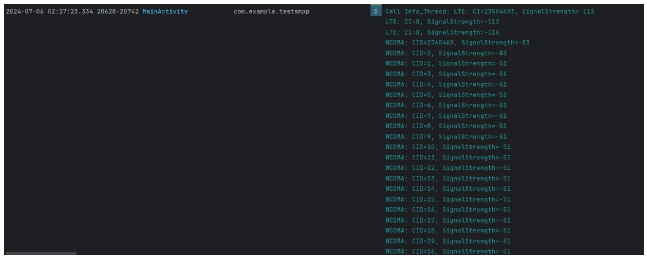
شکل ۳.۳: وارد کردن key نادرست



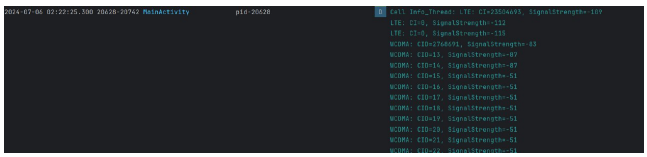
شکل ۴.۳: کد صحیح (مطابق با کد ذخیره شده در پروژه)

شکل ۵.۳: پیام دریافت شده

شکل ۶.۳: پیام دریافت شده در سرور



شکل ۷.۳: لاگ



شکل ۸.۳: اطلاعات مربوط به مکان

۴ توضیحات کدها

۱.۴ فایل MainActivity.kt

این کلاس MainActivity به عنوان فعالیت اصلی برنامه اندرویدی تعریف شده است. این فعالیت شامل قابلیت‌های ارسال و دریافت پیام‌های متنی (SMS) و درخواست مجوزهای مورد نیاز برای انجام این عملیات‌ها است. در ابتدا، برخی متغیرهای مربوط به ارسال و دریافت پیام‌های SMS تعریف شده‌اند.

۱.۱.۴ متد onCreate

در متد onCreate ، ابتدا نوار ابزار پنهان می‌شود و نمونه‌ای از LocationAndCellInfo برای دریافت اطلاعات مکان و سلول اولیه‌سازی می‌شود. فیلد ورودی شماره تلفن و دکمه‌های مختلف برای تنظیمات، نمایش پیام‌های SMS و خروج از برنامه نیز پیکربندی می‌شوند. یک HandlerThread برای ارسال پیام‌های SMS به صورت دوره‌ای راه‌اندازی می‌شود و اگر مجوزهای لازم صادر نشده باشند، درخواست مجوز از کاربر انجام می‌شود.

۲.۱.۴ متد sendSMS

متد sendSMS مسئول ارسال پیام‌های متنی است. در این متد، پیام‌های SMS با استفاده از SmppClient ارسال می‌شوند و وضعیت ارسال در لاگ ثبت می‌شود. در صورت موفقیت‌آمیز بودن ارسال، وضعیت ارسال در رابط کاربری به‌روز می‌شود.

۳.۱.۴ متد onDestroy

متد onDestroy ، زمانی که فعالیت از بین می‌رود، HandlerThread مربوط به ارسال پیام‌های SMS را به طور ایمن خاتمه می‌دهد.

۴.۱.۴ متد hasPermissions

متد hasPermissions بررسی می‌کند که آیا همه مجوزهای مورد نیاز برای برنامه صادر شده‌اند یا خیر. اگر همه مجوزها صادر شده باشند، مقدار true و در غیر این صورت مقدار false بازمی‌گرداند.

۵.۱.۴ متد onRequestPermissionsResult

متد onRequestPermissionsResult ، نتیجه درخواست مجوزها را بررسی می‌کند و در صورت صادر شدن همه مجوزها، شنونده پیام‌های SMS را شروع می‌کند. در صورت عدم صدور مجوزها، یک خطا در لاگ ثبت می‌شود.

۶.۱.۴ متد onNewIntent

متد onNewIntent ، زمانی که یک پیام جدید دریافت می‌شود، اطلاعات پیام دریافت شده را از Intent استخراج می‌کند و آنها را در لاگ ثبت می‌کند. همچنین، وضعیت تأیید دریافت پیام‌ها را به‌روز می‌کند و رابط کاربری را به‌روزرسانی می‌کند.

۷.۱.۴ متد startSMSListener

متد startSMSListener ، یک گیرنده (SMSReceiver) SMS را ثبت می‌کند که برای شنیدن پیام‌های دریافتی SMS استفاده می‌شود. این گیرنده با استفاده از IntentFilter برای اقدام SMS_RECEIVED_ACTION ثبت می‌شود.

۸.۱.۴ متد showAckStatus

متد showAckStatus ، وضعیت تأیید پیام‌های دریافتی را در رابط کاربری به‌روزرسانی می‌کند. این متد دو TextView را بر اساس وضعیت تأیید پیام‌ها قابل مشاهده یا پنهان می‌کند.

متد showSendStatus ، وضعیت ارسال پیام‌های SMS را در رابط کاربری به‌روزرسانی می‌کند. این متد دو TextView را بر اساس وضعیت ارسال پیام‌ها قابل مشاهده یا پنهان می‌کند.

۲.۴ فایل SMPP.kt

کلاس SMPP که از DefaultSmppSessionHandler به ارث برده شده، برای مدیریت جلسات SMPP در یک برنامه Android طراحی شده است. این کلاس شامل متغیرهایی برای نگهداری تنظیمات SMPP (مانند میزبان، پورت، کاربر، و رمز عبور) و همچنین روشی برای پیکربندی این تنظیمات می‌باشد. متد configure برای تنظیم جزئیات اتصال SMPP استفاده می‌شود.

۱.۲.۴ متد sendSMS

متد sendSMS تلاش می‌کند تا یک پیام SMS به شماره داده شده ارسال کند. این متد ابتدا یک SMPP client و session ایجاد می‌کندو پیام را به قسمت‌هایی تقسیم می‌کند. با دو آستانه مقدار قدرت سیگنال را تست می کند. اگر سیگنال قدرت سلول سرویس‌دهنده کمتر از 50- و یا 110- باشد، قسمت‌های پیام را ارسال می‌کند. سپس به مدت ۱۵ ثانیه منتظر می‌ماند و در نهایت session و client را می‌بندد و از بین می‌برد. اگر خطایی رخ دهد، آن را در log ثبت می‌کند و مقدار false را بازمی‌گرداند.

بخش آستانه:

```
if (servingpower != null){
    if (servingpower < -50)    {
        for (part in parts) {            val sm = createSubmitSm("989126211842"
            , "98$number", part, "UCS-2")
            println("Try to send message part")            session.submit(sm,
                TimeUnit.SECONDS.toMillis(50))
            Log.v("smpp", "hello")            println("Message part sent")
        }    }
    else        println("No need to change serving cell")
}
```

بخش تاخیر:

```
sendSmsRunnable = object : Runnable {    override fun run() {
        number = "9126211842"        text = locationAndCellInfo.getCellInfo()
        sendVal = true        sendHandler.post {
            sendSMS(smppClient)        }
        handler.postDelayed(this, 15000) // 15 seconds delay    }
}
```

۲.۲.۴ متد getSessionConfig

متد getSessionConfig پیکربندی جلسه SMPP را با استفاده از تنظیمات ذخیره شده در shared preferences بازمی‌گرداند. این متد یک شیء SmppSessionConfiguration را ایجاد می‌کند و نوع جلسه، میزبان، پورت، شناسه سیستم، و رمز عبور را تنظیم می‌کند.

۳.۲.۴ متد createSubmitSm

متد createSubmitSm یک شیء SubmitSm ایجاد می‌کند که برای ارسال پیام SMS استفاده می‌شود. این متد آدرس‌های مبدا و مقصد، کدینگ داده، و پیام کوتاه را تنظیم می‌کند و متن پیام را با استفاده از کاراکترست داده شده کدگذاری می‌کند.

۴.۲.۴ متد splitMessage

متد splitMessage پیام را به قسمت‌هایی با حداکثر طول بایت مشخص تقسیم می‌کند. این متد از یک کدکننده کاراکتر برای تبدیل پیام به یک buffer بایت استفاده می‌کند و پیام را به قسمت‌هایی تقسیم می‌کند که هر کدام طولی کمتر از maxByteLength دارند.

۵.۲.۴ متد extractServingCellSignalStrength

متد extractServingCellSignalStrength قدرت سیگنال سلول سرویس‌دهنده را از یک پیام استخراج می‌کند. این متد پیام را به بخش‌هایی بر اساس جداکننده ؛ تقسیم می‌کند و سپس اولین بخش را به عناصر جداگانه بر اساس ، تقسیم می‌کند و قدرت سیگنال

را به عنوان یک عدد صحیح بازمی‌گرداند.

۶.۲.۴ متد firePduRequestReceived

در نهایت، متد firePduRequestReceived که از DefaultSmppSessionHandler به ارث برده شده، درخواست‌های PDU دریافت شده را پردازش می‌کند. اگر کدینگ داده پیام ۰ باشد، آدرس مبدا، مقصد، و محتوای پیام را چاپ می‌کند و یک پاسخ ایجاد می‌کند و بازمی‌گرداند.

۳.۴ فایل SMSActivation.kt

در کلاس SMSActivation که از AppCompatActivity به ارث برده شده، متغیرهای لازم برای مدیریت مجوزهای دسترسی به پیامک‌ها، آداپتور چت و آیتم‌های چت تعریف می‌شود. در متد onCreate ، ابتدا چک می‌شود که آیا مجوز خواندن پیامک‌ها به برنامه داده شده است یا خیر. اگر مجوز داده نشده باشد، درخواست مجوز ارسال می‌شود و سپس پیامک‌ها خوانده می‌شوند. اگر مجوز قبلاً داده شده باشد، پیامک‌ها مستقیماً خوانده می‌شوند. سپس RecyclerView با استفاده از LinearLayoutManager مقداردهی اولیه می‌شود و آداپتور چت به آن متصل می‌شود. همچنین یک دکمه برگشت تنظیم می‌شود که با کلیک بر روی آن، فعالیت فعلی خاتمه می‌یابد.

۱.۳.۴ متد updateChatItems

در متد updateChatItems ، آیتم‌های چت جدید جایگزین آیتم‌های فعلی می‌شوند و به آداپتور اطلاع داده می‌شود که داده‌ها تغییر کرده‌اند. این متد برای به‌روزرسانی آیتم‌های چت استفاده می‌شود. متد loadChatItems نمونه‌ای از نحوه استفاده از این متد را نشان می‌دهد. در این متد، پیامک‌های جدید خوانده می‌شوند و سپس با استفاده از updateChatItems آیتم‌های جدید به‌روزرسانی می‌شوند.

۲.۳.۴ متد readSms

متد readSms پیامک‌های دستگاه را خوانده و آن‌ها را به لیستی از آیتم‌های چت تبدیل می‌کند. این متد ابتدا ستون‌های مورد نیاز برای خواندن پیامک‌ها (آدرس، متن و نوع پیامک) را مشخص می‌کند و سپس یک کوئری برای دریافت این داده‌ها از contentResolver ارسال می‌کند. در حالی که cursor به پیامک‌ها اشاره می‌کند، پیامک‌ها یکی یکی خوانده می‌شوند و اگر متن پیامک شامل عبارت " SMPP " باشد، متن پیامک پاک‌سازی می‌شود و سپس به لیست آیتم‌های چت اضافه می‌شود. در نهایت cursor بسته می‌شود و لیست آیتم‌های چت بازگردانده می‌شود.

۳.۳.۴ متد SMSActivation

در کلاس SMSActivation متغیرهای permission و requestCode برای مدیریت مجوزهای دسترسی به پیامک‌ها استفاده می‌شوند. همچنین recyclerView برای نمایش لیست پیامک‌ها، chatAdapter برای مدیریت آیتم‌های چت و chatItems برای نگهداری لیست پیامک‌ها تعریف شده‌اند. این متغیرها در متد onCreate مقداردهی اولیه می‌شوند و تنظیمات اولیه برای نمایش پیامک‌ها و مدیریت تعاملات کاربر با دکمه برگشت انجام می‌شود.

۴.۴ فایل SMSReceiver.kt

کلاس SMSReceiver که از BroadcastReceiver به ارث برده شده است، برای مدیریت پیام‌های ورودی SMS طراحی شده است. در این کلاس، متد onReceive بازنویسی شده تا پیام‌های SMS ورودی را پردازش کند. وقتی یک پیام SMS دریافت می‌شود، این متد فراخوانی می‌شود و پیام‌ها را از طریق intent دریافت می‌کند.

ابتدا، اگر bundle که حاوی داده‌های پیام است، غیر null باشد، مجموعه‌ای از PDUs (Protocol Data Units) از bundle استخراج می‌شوند. هر PDU به یک پیام SMS تبدیل می‌شود و از آن شماره فرستنده و متن پیام استخراج می‌گردد. سپس این اطلاعات در log برای اهداف اشکال‌زدایی ثبت می‌شود.

در مرحله بعد، چک می‌شود که آیا متن پیام حاوی عبارت "پیام شما دریافت شد" است یا خیر. اگر این عبارت در پیام یافت شود، یک intent جدید برای شروع MainActivity ایجاد می‌شود. این intent شامل flag هایی است که نحوه اجرای MainActivity را تعیین می‌کنند و اطلاعات مربوط به پیام را به صورت extras به آن اضافه می‌کنند.

در نهایت، MainActivity با استفاده از intent ایجاد شده و شامل اطلاعات پیام، آغاز می‌شود. این اقدام به MainActivity اجازه می‌دهد که پیام را دریافت کرده و به آن پاسخ دهد یا آن را نمایش دهد. این فرآیند به طور کامل، امکان پردازش و مدیریت پیام‌های SMS را در برنامه فراهم می‌آورد.

۵.۴ فایل SettingsActivation.kt

در این کد، ابتدا در onCreate ، تنظیمات اولیه انجام می‌شود. این فعالیت از AppCompatActivity ارث‌بری می‌کند و از XML فایل activity_sms را به عنوان طرح برای نمایش استفاده می‌کند. نوار ابزار این فعالیت مخفی شده است تا به نمایشگر متصل شود. سپس، دسترسی به خواندن پیام‌های SMS چک می‌شود و اگر اجازه دسترسی صادر نشده باشد، درخواست مجوز از کاربر گرفته می‌شود. پیام‌های SMS خوانده شده از readSms بازیابی و در RecyclerView نمایش داده می‌شوند.

در onCreate ، RecyclerView برای نمایش موارد چت مقداردهی اولیه می‌شود. از LinearLayoutManager برای مدیریت ترتیب عناصر استفاده می‌شود و یک ChatAdapter جهت اتصال به RecyclerView ساخته و تنظیم می‌شود. دکمه بازگشت (backButton) برای بستن فعالیت فعلی پیکربندی شده است.

در updateChatItems ، لیست موارد چت به روزرسانی می‌شود و تغییرات به ChatAdapter اطلاع داده می‌شود تا موارد نمایش

داده شده در RecyclerView به روز شوند.

loadChatItems یک مثال از تابع updateChatItems است که هر زمان که موارد چت جدیدی در دسترس باشد، برای به روزرسانی اطلاعات استفاده می‌شود.

در readSms ، از Content Provider مربوط به SMS برای بازیابی پیام‌های متنی استفاده می‌شود. این متد اطلاعاتی از شماره گیرنده (ADDRESS)، متن پیام (BODY) و نوع پیام (TYPE) را دریافت می‌کند. سپس پیام‌هایی که عبارت " SMPP " را در متن خود دارند را فیلتر و به لیست ChatItem . ChatAdapter اضافه می‌کند و در نهایت مجموعه داده را به عنوان خروجی بازمی‌گرداند.

۶.۴ فایل LocationAndCellInfo.kt

این کد یک کلاس به نام LocationAndCellInfo را نشان می‌دهد که برای دریافت اطلاعات مکانی و اطلاعات سلولی استفاده می‌شود. الگوهایی مانند LocationManager برای مدیریت خدمات مکانی و TelephonyManager برای خدمات تلفنی استفاده می‌شوند.

۱.۶.۴ شرح بخش‌های کد:

کلاس و ویژگی‌های آن:

LocationAndCellInfo: یک کلاس که مسئول مدیریت و دریافت اطلاعات مکانی و سلولی است.

context: متغیری که مربوط به context اکتیویتی یا سرویس است که این کلاس در آن ساخته می‌شود.

متغیرها:

- **locationManager:** برای مدیریت خدمات مکانی، از جمله درخواست‌های مکانی و شنود تغییرات مکانی.
- **telephonyManager:** برای دسترسی به اطلاعات تلفنی، مانند اطلاعات سلولی (CellInfo).
- **LocationListener:**

یک LocationListener به نام locationListener که برای شنیدن تغییرات مکانی و مدیریت آن‌ها استفاده می‌شود. این شامل توابعی برای زمانی که مکان تغییر کرده، غیرفعال شده یا فعال شده است.

- **initialize():**

متد initialize() که locationManager و telephonyManager را مقداردهی اولیه می‌کند با استفاده از context.getSystemService() .

- **startLocationUpdates():**

متد startLocationUpdates() که شروع به دریافت به‌روزرسانی‌های مکانی از GPS_PROVIDER می‌کند، پس از بررسی مجوزهای لازم.

- **stopLocationUpdates():**

متد stopLocationUpdates() که متوقف کردن دریافت به‌روزرسانی‌های مکانی را انجام می‌دهد.

- **getLastKnownLocation():**

متد getLastKnownLocation() که آخرین مکان شناخته شده را برمی‌گرداند، اگر مجوز مکانی فراهم باشد.

- **getCellInfo():**

متد getCellInfo() که اطلاعات سلولی را برمی‌گرداند، شامل نوع سلول (GSM ، CDMA ، LTE یا WCDMA) و اطلاعات شناسه سلول و قدرت سیگنال آن. این کلاس کمک می‌کند تا به راحتی از ویژگی‌های مکانی و اطلاعات سلولی دسترسی پیدا کرد و از آن‌ها در برنامه استفاده کرد.

۵ مراجع

۱.