

Universitatea POLITEHNICA din București

Facultatea de Automatică și Calculatoare,
Departamentul de Calculatoare



LUCRARE DE DIPLOMĂ

Securizarea interacțiunilor online

Conducător Științific:

As. Dr. Ing. Mihai Bucicoiu

Autor:

Bucur Florin

București, 2015

Aş dorii să îi mulţumesc coordonatorului meu, Mihai Bucicoiu, pentru suportul acordat pe parcursul acestui proiect de licenţă.

Abstract

Având în vedere că în acest moment foarte multe persoane folosesc rețelele de socializare pentru a interacționa, riscurile de securitate a datelor personale sunt destul de mari. Astfel, acest proiect de licență își dorește îmbunătățirea metodelor de securizare oferite de platformele sociale, prin criptarea datelor postate, folosind o aplicație pentru dispozitivele mobile Android.

Cuprins

Mulțumiri	i
Abstract	ii
Listă de figuri	v
1 Introducere	1
1.1 Motivația proiectului	1
1.2 Obiectivele proiectului	2
2 Aplicații similare	3
2.1 OTR	3
2.1.1 Xabber	4
2.1.2 Beem	4
2.2 PGP	4
2.2.1 Android Privacy Guard	5
2.2.2 PGP SMS	5
2.2.3 PGP Viewer	6
3 Tehnologii folosite	7
3.1 Android	7
3.1.1 Prezentare generală	7
3.1.2 Arhitectura Android	7
3.1.3 Event Bus	8
3.1.4 Notification Listener Service	8
3.1.5 Recycler View	9
3.2 Facebook SDK	9
3.2.1 Prezentare generală	9
3.2.2 Graph API	11
4 Proiectarea arhitecturii	12
4.1 Conectarea la rețeaua socială	13
4.2 Conversații criptate	14
4.3 Grupuri protejate	15
4.4 Partajare chei	16
5 Implementare	17

5.1	Implementarea aplicației	17
5.1.1	Interfață grafică	17
5.1.2	Autentificare	19
5.1.3	Listă de prieteni	20
5.1.4	Grupuri	20
5.1.5	Trimitere - Recepție mesaje	22
5.2	Integrarea criptării transmisiei de date	24
5.2.1	Criptarea	26
5.2.2	Decriptarea	26
6	Testarea și evaluarea soluției	28
6.1	Autentificare	28
6.2	Trimitere - Recepție mesaje	29
6.3	Grupuri protejate	30
6.4	Evaluarea rezultatelor	31
6.4.1	Probleme întâmpinate	32
7	Concluzii	33
7.1	Activități viitoare	34
	Bibliografie	35

Listă de figuri

2.1	Criptarea OTR	3
2.2	Criptarea PGP	5
3.1	Arhitectura Android	8
3.2	Principiul de funcționare Event Bus	9
3.3	ID-ul aplicației	10
3.4	Cheia pentru dezvoltare	10
3.5	Token de acces	10
4.1	Arhitectură	12
4.2	Autentificare	13
4.3	Schimbul de mesaje	14
4.4	Gestionare grupuri protejate	15
5.1	Interfață grafică	18
5.2	Criptarea folosind CBC	25
5.3	Decriptarea folosind CBC	25
6.1	Rezultatul autentificării	28
6.2	Schimbul de mesaje	29
6.3	Mesajele criptate de pe un grup	30

Capitolul 1

Introducere

În această lucrare de licență se va discuta despre securizarea interacțiunilor online, accentul fiind pus pe comunicarea din cadrul unei rețele de socializare. Securitatea datelor din mediul virtual a devenit un subiect tot mai important, abordat de foarte multă lume.

Rețelele sociale dețin foarte multe informații personale despre utilizatorii lor, de aceea au apărut atacuri asupra rețelelor de socializare. Acestea au ca scop aflarea de informații private despre persoane.

Având în vedere faptul că dispozitivele mobile au devenit foarte răspândite, acest proiect dorește să aducă un plus de securitate comunicației inițiate de pe dispozitivele care folosesc Android ca sistem de operare. Rețeaua de socializare aleasă este Facebook.

1.1 Motivația proiectului

În momentul actual rețelele de socializare nu pun la dispoziție soluții de criptare a datelor postate de membrii lor. Astfel, dacă un atacator reușește să obțină credențialele unui utilizator, acesta poate accesa toate informațiile private ale victimei fără nicio problemă. De aceea, criptarea datelor pune în dificultate atacatorul, acesta neputând să înțeleagă datele.

Pe lângă faptul că un atacator nu poate descifra datele, criptarea se face doar pe dispozitivele utilizatorilor, astfel echipamentele rețelelor sociale nu ar fi solicitate în plus pentru procesul de criptare sau decriptare a datelor.

Dispozitivele mobile au evoluat foarte mult în ultimii ani, acestea beneficiind de procesoare mult mai puternice, astfel procesul de criptare sau decriptare în timp real a unor date de dimensiuni mici, nu reprezintă o problemă.

Un atacator poate încerca să se folosească de faptul că dispozitivele mobile folosesc medii de transmitere fără fir. Acestea sunt destul de nesigure în momentul actual, iar un atacator poate găsi o modalitate de a intercepta transmisiile de date. Astfel dacă datele sunt transmise și recepționate sub formă criptată, cel care interceptează comunicațiile nu are cum să descifreze informațiile.

Am ales sistemul de operare Android pentru că este un proiect open-source în continuă dezvoltare care câștigă din ce în ce mai mulți utilizatori, deci este o țintă a atacurilor.

Facebook, a fost ales deoarece este printre cele mai mari rețele de socializare la momentul scrierii acestei lucrări. De obicei, platformele cele mai populare devin ținta unui număr mare de atacuri.

1.2 Obiectivele proiectului

Principalele obiective ale acestei lucrări de licență sunt:

1. Crearea unei aplicații Android care să ofere posibilitatea interacționării cu rețeaua de socializare Facebook.
2. Schimbul de mesaje în cadrul rețelei.
3. Gestionarea grupurilor protejate și postarea de mesaje în cadrul acestora.
4. Criptarea datelor postate.
5. Decriptarea datelor postate din această aplicație.

Capitolul 2

Aplicații similare

2.1 OTR

Criptarea OTR (Off-the-Record), este un protocol criptografic folosit pentru securizarea canalelor de comunicație. Este întâlnit în special în aplicațiile care presupun schimb de mesaje. Acest model de criptare poate fi folosit peste orice protocol de mesagerie, precum Yahoo Messenger, Facebook Messenger, Google Talk.

OTR folosește chei de criptare simetrice, aceeași cheie și pentru criptare și pentru decriptare. Procesul este reprezentat în figura 2.1. Datorită faptului că, nu se pot trimite cheile pe un canal considerat nesigur, se aplică algoritmul Diffie-Hellman pentru schimbul de chei. Astfel schimbul de chei se face în așa fel încâ un eventual atacator care interceptează mesajele de pe canalul de comunicație, nu poate descifra mesajul pentru că nu are acces la cheile folosite pentru criptarea mesajelor [9].

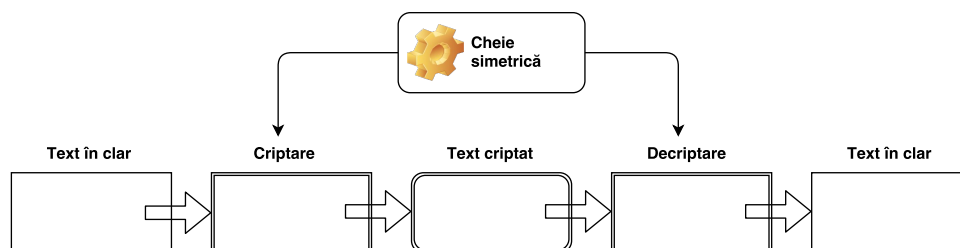


Figura 2.1: Criptarea OTR

Un alt aspect bun, din punct de vedere al securității, este faptul că pentru fiecare mesaj nou transmis, cheia de criptare este schimbată. Deci, chiar dacă cineva ar avea acces la schimbul de mesaje, pentru acesta ar fi imposibil să descifreze conținutul.

O problemă a protocolului o reprezintă faptul că nu se folosesc semnături digitale, astfel este imposibil de demonstrat dacă cineva a reușit să falsifice niște mesaje. Această problemă se poate rezolva prin introducerea conceptului de autentificare a utilizatorilor. Folosind autentificarea, se pot găsi mesajele falsificate [9].

Această criptare este disponibilă în mai multe aplicații de chat. Una foarte importantă este Pidgin, aceasta fiind disponibilă pe Linux, Windows dar și pe OS X. Pentru dispozitivele mobile, cum ar fi cele care folosesc ca sistem de operare Android, există ChatSecure. Astfel tot ce trebuie făcut este să se instaleze aceste aplicații în funcție de platformele folosite, activată opțiunea de utilizare OTR și bineînțeles autentificarea utilizatorului. Alte aplicații Android care folosesc criptarea OTR pentru schimbul de mesaje sunt Xabber și Beem. Amândouă pot fi descărcate și folosite gratis.

2.1.1 Xabber

Xabber, este o aplicație Android care folosește pe partea de comunicație protocolul XMPP, protocol ce este conceput în special pentru schimbul de mesaje și este bazat pe limbajul XML. XMPP este un proiect open-source, care este standardizat și dă posibilitatea oricui să dezvolte aplicații bazate pe XMPP.

Această aplicație vine însoțită de servicii preconfigurate ce folosesc XMPP și oferă posibilitatea integrării contactelor din sistemul de operare cu orice cont XMPP. De asemenea este compatibil cu mai multe standarde, precum servere Ejabberd sau Openfire. Schimbul de mesaje se face similar cu Internet Relay Chat(IRC), iar mesajele pot fi criptate folosind OTR [1].

2.1.2 Beem

Beem, este un proiect open-source care își dorește să ofere posibilitatea de a folosi toate funcționalitățile puse la dispoziție de Jabber și XMPP. Este compatibil cu orice standard bazat pe XMPP(Google Talk, Prosody, Openfire).

Această aplicație oferă posibilitatea de a schimba mesaje cu diferite persoane din lista de contacte. Mesajele pot fi criptate folosind OTR [2].

2.2 PGP

Criptarea PGP sau criptarea Pretty Good Privacy are la baza un standard numit OpenPGP, care este folosit pentru criptarea de date și semnături digitale. PGP folosește atât chei publice, cât și chei private.

Cu ajutorul cheii publice, se criptează mesajul, iar cu ajutorul cheii private, se decriptează mesajul. Astfel, fiecare utilizator generează o cheie publică pe care o pune la dispoziție altor utilizatori, iar cheia privată este ținută secretă, deci doar cel care deține cheia privată poate decripta mesajele. Procesul este reprezentat în figura 2.2

Un mare avantaj al folosirii de chei publice este posibilitatea de a utiliza semnături digitale. Cu ajutorul acestora, se pot semna datele transmise, astfel încât, orice încercare de falsificare a datelor sau de modificare a datelor este depistată foarte ușor. Semnăturile digitale sunt create cu ajutorul unor algoritmi matematici care combină cheia privată

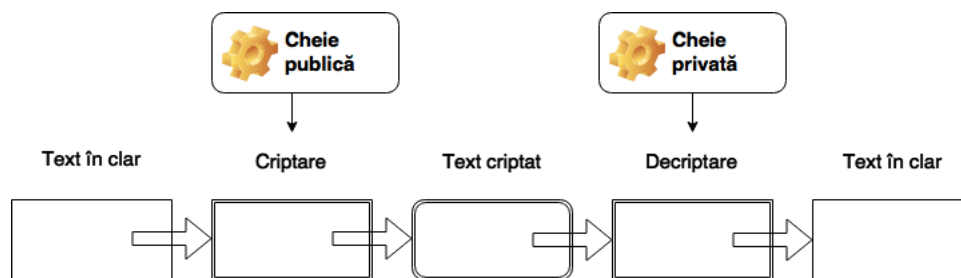


Figura 2.2: Criptarea PGP

cu datele care urmează să fie transmise. Ulterior, dacă cineva dorește să verifice autenticitatea datelor primite sau integritatea lor, va aplica pe datele recepționate, cheia publică a transmițătorului.

Pentru a folosi criptarea PGP, în primul rând trebuie să generăm un certificat, care poate să conțină date extra, cum ar fi numele, adresa de email sau alte semnături digitale ale altor utilizatori. Un certificat se poate genera pe orice sistem de operare, iar acesta poate fi pus la dispoziția altor oameni, făcându-le publice pe anumite servere dedicate cum ar fi, serverul pentru chei publice al celor de la MIT.

Astfel, dacă cineva intenționează să comunice cu o anumită persoană, trebuie să îi caute cheia publică pe aceste servere dedicate, dacă nu cumva o deține deja. Acum poate cripta mesajul și îl poate trimite.

Decriptarea presupune cunoașterea unei parole care securizează cheia privată. Deci mesajele recepționate sunt decriptate folosind cheia privată deținută.

Pentru schimbul de chei publice există mai multe soluții. O soluție este de a obține cheia publică de la partener, chiar în persoană, dar nu este cea mai comodă posibilă. Există conceptul de fingerprint, care reprezintă un cod hash al certificatului în hexazecimal, care poate fi verificat telefonic. O altă modalitate este de a obține cheile publice folosind mai multe surse, verificând dacă toate coincid.

Această criptare este disponibilă în mai multe aplicații care presupun schimb de mesaje. Ca de exemplu, pentru Android există APG, PGP SMS, Symantec PGP Viewer.

2.2.1 Android Privacy Guard

APG este un proiect gratuit și open-source, ce permite criptarea și decriptarea de fișiere și mesaje din email-uri. Pe lângă criptare și decriptarea folosind chei publice și chei private, oferă și posibilitatea de a semna digital fișierele și mesajele. De asemenea, se poate folosi și criptarea pe baza de chei simetrice. Criptarea bazată pe chei publice - private, se realizează folosind protocolul OpenPGP [3].

2.2.2 PGP SMS

Această aplicație folosește criptarea PGP utilizând chei publice pentru criptare și chei private pentru decriptare. Cheia publică este trimisă persoanei care dorește să trimită

un mesaj criptat. Aceasta criptează mesajul folosind cheia primită și transmite mesajul. La recepție, mesajul este decriptat cu ajutorul cheii private [4].

2.2.3 PGP Viewer

PGP Viewer permite decriptarea mesajelor din email-uri, fișierelor sau a atașamentelor din email-uri. Rezultatul decriptării este copiat în directorul de intrare a email-ului. Clientul de email folosit este cel standard al sistemului de operare Android. Astfel nu este necesară instalarea de noi aplicații.

Informațiile criptate din email sunt disponibile, chiar dacă utilizatorul nu este conectat la Internet. Aplicația mai oferă și posibilitatea de a verifica semnăturile digitale ale datelor, pentru a valida faptul că datele nu sunt falsificate [5].

Capitolul 3

Tehnologii folosite

În acest capitol se introduce sistemul de operare Android, arhitectura sa și funcționalitățile puse la dispoziție de acesta. De asemenea sunt prezentate și detalii referitoare la librăria oferită de Facebook pentru dezvoltarea de aplicații Android care interacționează cu rețeaua de socializare. Tot în acest capitol se vor regăsi amănunte legate de pașii ce trebuie urmați pentru crearea unei aplicații Facebook.

3.1 Android

3.1.1 Prezentare generală

Android este un sistem de operare creat pentru dispozitivele mobile, cum ar fi smartphone-uri și tablete. Acest sistem este creat și promovat de Google. De asemenea acest sistem de operare este disponibil pe mai multe tipuri de dispozitive, produse de un număr mare de companii [6].

Android este bazat pe o versiune modificată de Linux, împrumutând tehnologii care se ocupă de gestiunea componentelor hardware, a memoriei sau a proceselor. Tot odată folosește biblioteci Java pentru partea de grafică, telefonie sau pentru partea de conectivitate.

Un aspect important al acestui sistem de operare este faptul că este un produs open-source, deci producătorii de dispozitive mobile, pot modifica sistemul de operare astfel încât, să se poată folosi extensii ale hardware-ului, deși sistemul de operare de bază nu oferă suport pentru aceste extensii.

În această secțiune se vor introduce mai întâi arhitectura Android și apoi se vor detalia componentele sale utilizate.

3.1.2 Arhitectura Android

Arhitectura Android este compusă din patru elemente importante: Linux Kernel, Librării, Frameworks și Aplicații. În continuare vom detalia fiecare nivel al arhitecturii

folosindu-ne și de Figura 3.1.

Primul nivel este Linux Kernel, care este folosit pentru comunicația dintre componentele hardware și componentele software.

Următorul nivel al arhitecturii este reprezentat de librăriile de bază. Aceste librării sunt scrise în limbajele de programare C/C++. Aici găsim librăriile pentru grafică, pentru baze de date, pentru browser-ul WEB.

Android RunTime este compus din mașina virtuală Dalvik și de librării ale procesorului.

Frameworks, este scris în totalitate în limbajul Java și este folosit pentru dezvoltarea de aplicații Android.

Ultimul nivel este reprezentat de aplicațiile Android. Acesta este nivelul cel mai înalt, aici găsim aplicații precum cele pentru contacte, jocuri sau browser-ul.

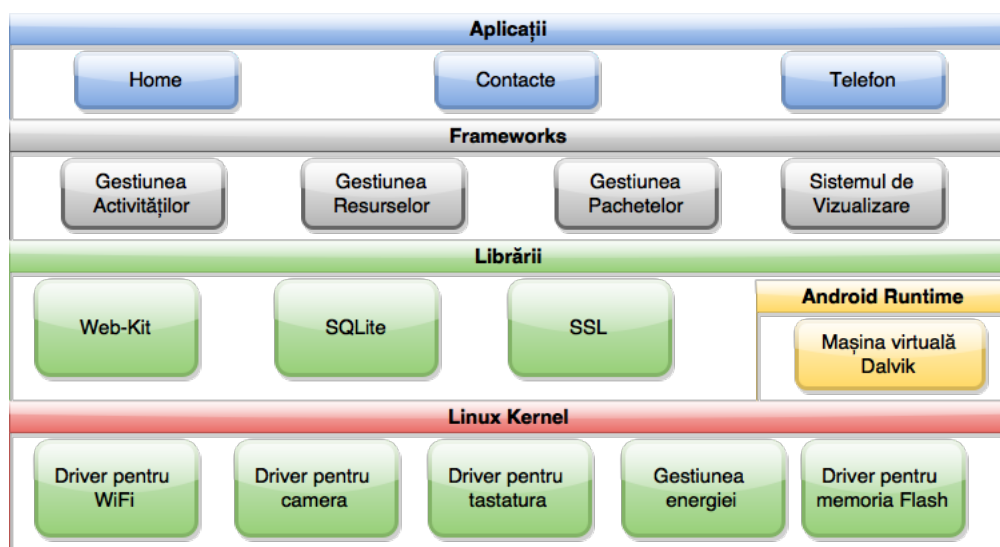


Figura 3.1: Arhitectura Android

3.1.3 Event Bus

Event Bus oferă o modalitate optimizată și simplificată de a comunica între activități, fragmente, fire de execuție sau servicii Android. Folosind această librărie se optimizează procesul de dezvoltare, iar calitatea comunicației este mai bună decât în cazul comunicației standard. Principiul de funcționare Event Bus este descris în Figura 3.2.

3.1.4 Notification Listener Service

Notification Listener Service, este un serviciu folosit pentru gestionarea notificărilor. Cu ajutorul lui la primirea sau recepționarea unor notificări se pot declanșa activități bazate pe datele care vin odată cu notificările.

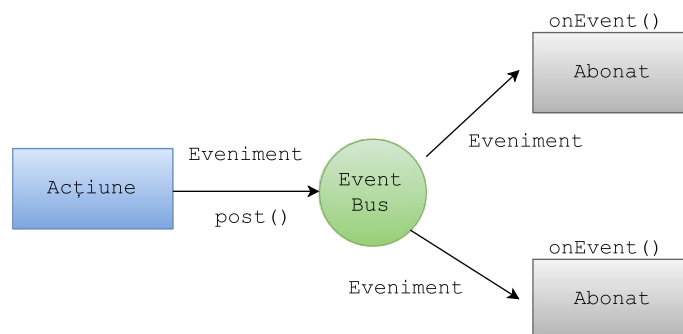


Figura 3.2: Principiul de funcționare Event Bus

Acest serviciu se folosește de un obiect `StatusBarNotification`, prin intermediul căruia se primesc informațiile necesare despre notificările existente.

Pentru a putea folosi acest serviciu, utilizatorul trebuie să activeze opțiunea de a permite aplicației să se folosească de notificările primite în sistem. Opțiunea se găsește în calea "Settings > Security > Notification access"

Comunicația dintre acest serviciu și activitatea care prelucrează informațiile primite odată cu notificările se realizează cu ajutorul librăriei Event Bus, prezentată mai sus.

3.1.5 RecyclerView

`RecyclerView` reprezintă o versiune mult mai flexibilă față de `ListView`, folosită pentru a afișa pe ecran o listă cu un număr mare de elemente, dar într-o modalitate mult mai eficientă față de `ListView`.

Elementele listei nu sunt încărcate toate în memorie, ci doar cele afișate. Se pot adăga elemente sau șterge elemente foarte ușor și într-un mod dinamic, existând în mod implicit activate animații pentru evenimentele de adăgare sau ștergere de elemente.

3.2 Facebook SDK

3.2.1 Prezentare generală

Facebook, este una dintre cele mai mari și importante rețele de socializare din prezent. Platforma Facebook, este disponibilă pe toate tipurile de dispozitive, având suport pentru majoritatea sistemelor de operare.

Cu ajutorul Facebook SDK, se pot crea aplicații care implică autentificarea în rețeaua de socializare și se pot folosi anumite funcționalități ale platformei, direct din aplicațiile care nu sunt create de Facebook. De exemplu, putem crea o aplicație care să ofere posibilitatea de a posta un mesaj pe wall-ul personal.

Pentru a folosi Facebook SDK, este necesar un cont de Facebook, în care se crează o nouă aplicație. Se alege un nume al aplicației și se generează un identificator unic al aplicației care arată ca în Figura 3.3. Pentru ca aplicația să fie funcțională, acest id unic trebuie inclus în codul acesteia. După ce id-ul aplicației este obținut, trebuie să includem în proiect SDK-ul oferit de Facebook și setat id-ul.

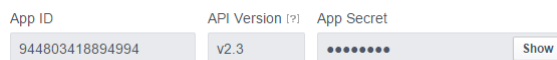


Figura 3.3: ID-ul aplicației

Pe lângă toate acestea, este necesară o cheie pentru dezvoltare și una pentru publicarea aplicației. Acestea pot fi identice și sunt reprezentate în Figura 3.4.



Figura 3.4: Cheia pentru dezvoltare

Fiecare sesiune deschisă, are asociat un token de acces. Acesta poate să difere de la sesiune la sesiune, chiar dacă este vorba despre același user. Access token-ul este generat în funcție de permisiunile cerute de aplicație sau de modificările apărute în profilul utilizatorului. Acest token este o modalitate de identificare, dar și de securizare a sesiunilor. Un exemplu de token este prezent în Figura 3.5.

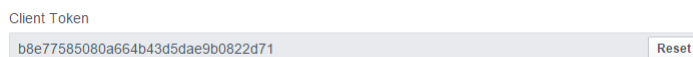


Figura 3.5: Token de acces

Un alt aspect important al SDK-ului de Facebook, este faptul că s-au introdus două tipuri de id-uri ale utilizatorului. Astfel, pentru un utilizator, există un id, global sau real și un id unic pentru fiecare aplicație cu ajutorul căreia s-a autentificat. Cu aceste id-uri unice per aplicație se limitează accesul la anumite funcționalități. De exemplu, deschiderea unui ferestru de Facebook Messenger, direct pentru un utilizator specific, nu este posibilă, pentru că id-ul acestuia nu este cel global, iar aplicația nativă de Facebook Messenger, nu va recunoaște id-ul.

La începutul acestei practici, de a pune la dispoziție dezvoltatorilor de aplicații, funcționalitățile rețelei de socializare, a fost un lucru bine văzut de toată lumea. Cu timpul, securitatea datelor personale și intimitatea utilizatorilor au reprezentat niște puncte de reper foarte importante pentru evoluția SDK-urilor. În momentul scrierii acestei lucrări de licență, cel mai recent SDK este Facebook SDK 4.2.0.

De anul acesta Facebook, a renunțat la ideea de a lăsa dezvoltatorii să își facă propriile lor aplicații pentru schimbul de mesaje private. Prin aceste limitări, se pot evita aplicațiile robot, care fac spam și nu sunt utile cu nimic. O altă modalitate de a evita spam-ul este de a forța folosirea aplicațiilor native pentru anumite operații.

3.2.2 Graph API

Principala metodă de a obține, șterge sau adăuga informații în rețeaua de socializare, este Graph API. Graph API este un API de nivel jos, bazat pe protocolul de comunicație, HTTP. Denumirea acestuia vine de la ideea de graf social, în care găsim elemente cum ar fi, noduri, muchii sau câmpuri de informații. Nodurile reprezintă lucruri de tipul utilizator, fotografie sau comentariu. Muchiile reprezintă conexiunile dintre noduri, ca de exemplu fotografiile dintr-o pagină. Câmpurile de informații reprezintă descrierea nodurilor, de exemplu ziua de naștere a unui utilizator sau numele unei pagini.

Graph API poate fi folosit cu orice limbaj de programare ce oferă suport pentru cereri HTTP. Pentru a obține informații de pe platforma Facebook, trebuie creată o cerere GET. Pentru a adăuga mesaje, imagini, link-uri sau orice alte date, se folosește cereri de tip POST, iar pentru a șterge date, se folosește o cerere specială, DELETE. Fiecare cerere Graph API necesită un token de acces. Token-ul de acces este cel obținut prin implementarea autentificării în rețeaua de socializare.

Versiunea actuală de Graph API este 2.3. Pentru că există mai multe versiuni, se poate specifica în cereri versiunea de API folosită. Totuși, este indicat să se folosească ultima versiune, deoarece, funcționalități din cele vechi pot dispărea sau se pot modifica.

O cerere GET folosind Graph API este de forma `https://graph.facebook.com/v2.3/node-id/edge-name`, unde node-id reprezintă identificatorul nodului, de exemplu id-ul unui prieten, iar edge-name poate fi "photos" pentru pozele acelui prieten. Datele întoarse de Facebook ajung în format JSON, putând fi parsate foarte ușor.

O cerere POST folosind Graph API, poate fi de forma `https://graph.facebook.com/v2.3/node-id/edge-name`, unde node-id poate fi identificatorul unui grup, iar edge-name poate fi șirul de caractere "members". Această cerere are nevoie să i se adauge un câmp extra, cum ar fi "member", acestui câmp având asociat ca valoare identificatorul unei persoane care va fi adăugată în grup.

În cazul în care se dorește ștergerea unui element de pe rețeaua de socializare, cererea arată similar cu cea de POST, dar în loc să se specifice metoda POST, se va seta ca metodă folosită cea de DELETE.

Capitolul 4

Proiectarea arhitecturii

Aplicația își propune să îmbunătățească securitatea conversațiilor din cadrul rețelelor de socializare, precum Facebook sau Google Plus. Mai exact este vorba de criptarea mesajelor private transmise prin intermediul unei rețele de socializare, dar și a postărilor din anumite spații private de comunicare, precum grupurile create de aplicație.

Pentru asigurarea securității, fiecare dispozitiv mobil care utilizează aplicația este considerat un echipament de încredere ce va proteja prin criptare mesajele, înainte de a le trimite către platforma de socializare.

Cheile folosite pentru criptare sunt comunicate între participanți independent de rețeaua socială, folosind alte căi de comunicare, precum SMS, coduri QR sau direct prin telefon, așa cum se poate vedea și în Figura 4.1.

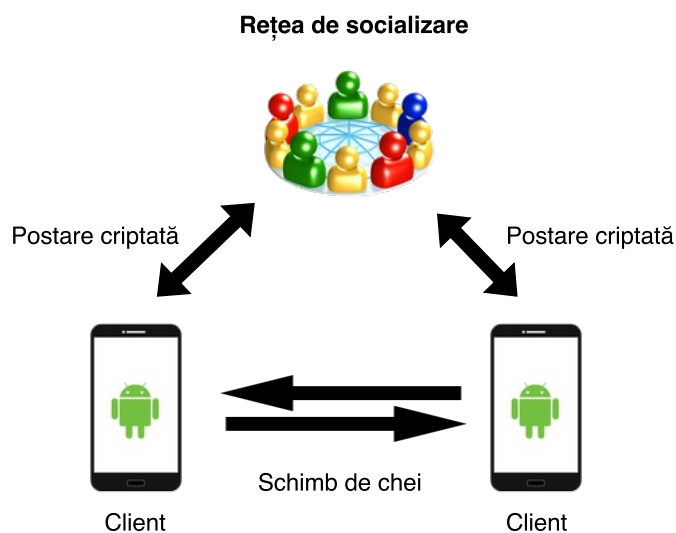


Figura 4.1: Arhitectură

4.1 Conectarea la rețeaua socială

Folosirea aplicației presupune ca utilizatorul să dețină un cont pe rețeaua de socializare, cu ale cărui credențiale se poate realiza autentificarea, așa cum este prezentat și în Figura 4.2. În cazul în care contul este setat deja în sistemul de operare, acesta se preia automat de către aplicație și se autentifică utilizatorul. De asemenea, este necesar ca dispozitivul să aibă instalată aplicația nativă pusă la dispoziție de platforma rețelei.

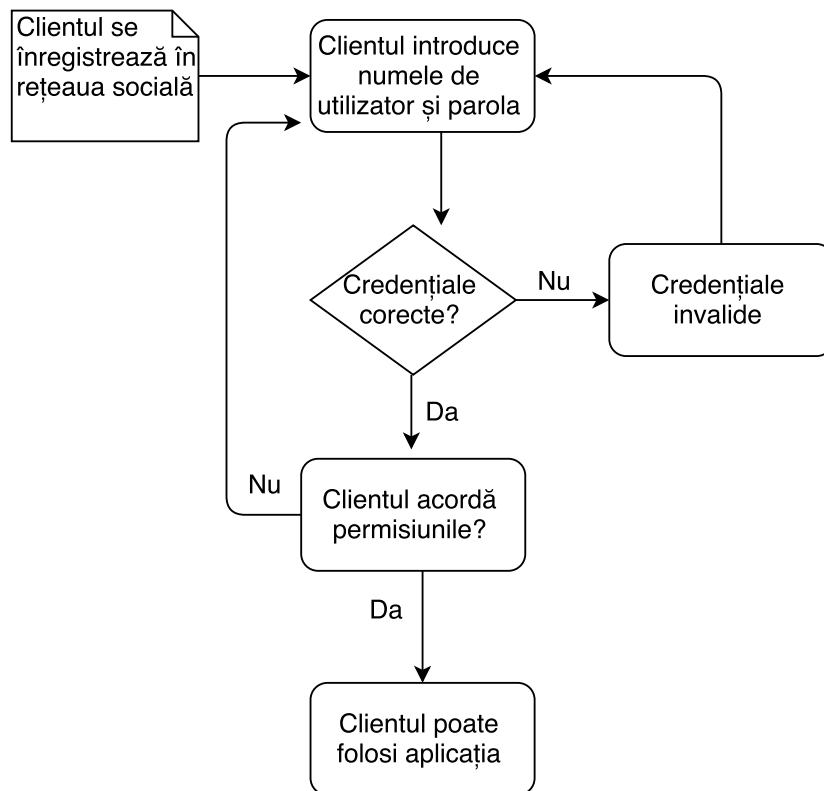


Figura 4.2: Autentificare

Autentificare printr-o altă aplicație decât cea nativă necesită permisiunea utilizatorului ca aplicația să acceseze datele din contul său. Dacă utilizatorul este de acord cu accesarea datelor de către aplicație, înseamnă că procesul s-a terminat cu succes. Dacă nu, utilizatorul este obligat să reînceapă procesul de autentificare.

Dacă autentificare s-a încheiat cu succes, aplicația primește din partea platformei rețelei sociale un token de acces, care a fost decriptat în secțiunea 3.2.1. Cu acesta se vor deschide noi sesiuni de comunicație între aplicația client și rețeaua de socializare.

4.2 Conversații criptate

Pașii ce trebuie urmați pentru a purta o conversație criptată sunt prezentați în Figura 4.3 și îi vom descrie în continuare:

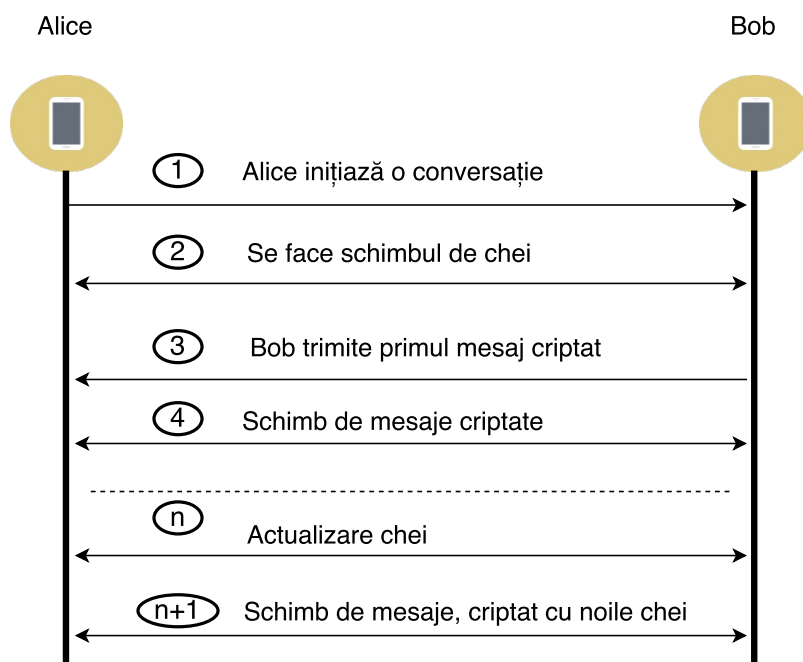


Figura 4.3: Schimbul de mesaje

Pasul 1: Inițierea conversațiilor necesită aplicația nativă de Messenger a rețelei sociale. Mesajul de inițiere este trimis în clar prin intermediul aplicației native și este de preferat să nu reprezinte informații relevante pentru cineva. Acesta poate fi de exemplu șirul de caractere "Hello". La receptor se va primi o notificare ce va conține mesajul de inițiere, notificare pe baza căreia se va putea răspunde la acest mesaj.

Pasul 2: Schimbul de chei se poate face și înainte de mesajul de inițializare dar și după acesta. Pentru a decripta cu succes mesajele, cei care participă la conversație trebuie să dețină cheile de criptare ale partenerilor. Schimbul de chei va fi prezentat mai detaliat în secțiunea 4.4.

Pasul 3: În momentul în care mesajul de inițiere a conversației ajunge la destinatar, se știe că se poate începe schimb de mesaje criptate folosind cheile de criptare propuse de fiecare participant. Acum receptorul poate să răspundă la notificarea primită printr-un mesaj protejat.

Pasul n: Cheile se pot modifica pe parcursul unei conversații. Deci fără o gestiune avansată de chei, mesajele schimbate nu vor mai putea fi decriptate în viitor, pentru că nu se vor mai ști cheile de criptare pentru mesaje. De aceea participanții la conversații pot folosi o singură cheie pentru o perioadă mai lungă de timp, în ideea că se va dori decriptarea informațiilor.

În locul aplicațiilor native de Messenger, se va folosi un serviciu care interceptează notificările primite în sistemul de operare. Vor fi selectate notificările care vin de la rețeaua de socializare și cele care vin de la prietenul selectat din lista de prieteni care folosesc aplicația.

4.3 Grupuri protejate

După procesul de autentificare, utilizatorul poate beneficia de funcționalitățile disponibile. Pentru început se va afișa lista de prieteni care, de asemenea folosesc această aplicație. În continuare vom descrie fluxul de acțiuni prezente în Figura 4.4.

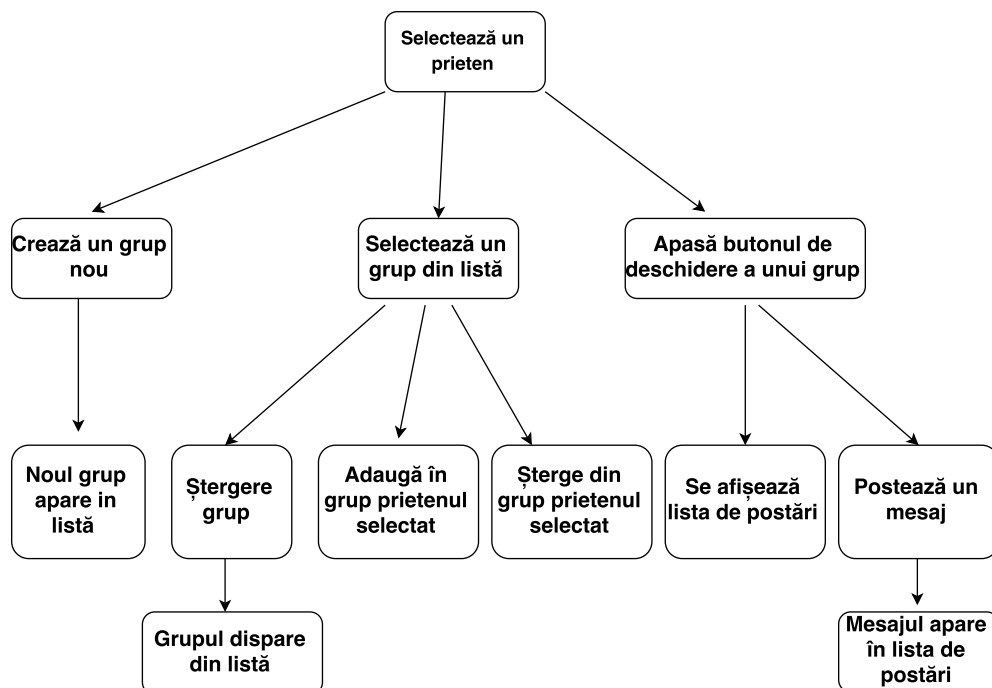


Figura 4.4: Gestionare grupuri protejate

Prin selectarea unui prieten, se va deschide o nouă activitate de unde se pot iniția conversații sau gestiona grupuri.

Grupurile create de aplicație sunt afișate într-o listă. Se oferă posibilitatea de a selecta un grup și de a aplica pe el o operație de ștergere grup, care va avea ca efect eliminarea

din listă a grupului sau adăugare și eliminare membrii din grup.

De asemenea se pot crea noi grupuri ale aplicației sau se pot vedea mesajele postate pe un grup selectat. Mesajele care sunt postate din aplicație sunt criptate.

Pentru vizualizarea sau postarea mesajelor se apasă butonul de deschidere a grupului și este nevoie ca utilizatorul să cunoască o cheie de criptare. Aceasta va fi prezentată în secțiunea 4.4 și poate fi diferită de la grup la grup, astfel este sporit nivelul de securitate. După ce mesajul a fost postat cu succes, acesta apare în lista de postări.

Dacă cineva postează mesaje pe un grup al acestei aplicații, dar folosind un browser sau aplicațiile native, acele mesaje nu vor putea fi afișate folosind această aplicație, deoarece decriptarea aplicată pe un mesaj în clar nu va funcționa.

Pentru gestionarea grupurilor se folosește SKD-ul pus la dispoziție de rețeaua de socializare. Toate cererile sunt trimise prin intermediul Graph API-ului, prezentat în secțiunea 3.2.2.

4.4 Partajare chei

Cheile folosite în aceasta aplicație sunt simetrice, deci utilizatorii trebuie să dețină aceleași chei pentru a putea avea o conversație criptată sau pentru a putea posta sau vizualiza mesaje de pe grupuri.

Faptul că se încearcă securizarea unor canale de comunicație nesigure duce la problema schimbului de chei. Este de evitat trimiterea cheilor pe aceste canale nesigure, de aceea vor fi folosite alte modalități de comunicare cum ar fi SMS sau coduri QR.

În momentul în care se dorește începerea unei conversații, partenerii de dialog trebuie să cadă de acord asupra unei chei de criptare. Aceștia își pot comunica cheia prin SMS sau prin intermediul unui cod QR. Pentru schimbul de mesaje, cheile se pot schimba de câte ori doresc participanții.

Pentru a posta mesaje criptate sau pentru a citi și decripta mesaje din cadrul unui grup protejat, este nevoie ca cei care doresc să participe cu informații să afle cheia de criptare. Cheia se transmite tot prin SMS sau coduri QR, dar este transmisă doar de persoanele care fac parte din grup. Astfel dacă unui membru i se cere cheia de criptare, acesta poate evalua dacă persoana care vine cu această solicitare este și ea membru al grupului și dacă este de încredere.

Capitolul 5

Implementare

În acest capitol se vor prezenta detalii despre modul de implementare a funcționalităților aplicației. Se vor oferi amănunte legate de interfața grafică, procesul de autentificare, gestionare a grupurilor, schimbul de mesaje private și criptarea, respectiv decriptarea datelor transmise.

Pe scurt, aplicația oferă posibilitatea autentificării pe baza datelor de acreditare a unui cont pe rețeaua de socializare Facebook și a folosirii anumitor funcționalități puse la dispoziție de Graph API. Gestionarea grupurilor se face cu ajutorul metodelor oferite de platforma Facebook, iar schimbul de mesaje, respectiv criptarea și decriptarea lor nu au suport pentru dezvoltarea de aplicații Android. Pentru partea de mesaje private este folosit un serviciu de gestionare a notificărilor primite în sistemul de operare Android. Criptarea datelor este de tip AES și folosește chei simetrice.

5.1 Implementarea aplicației

5.1.1 Interfață grafică

Dezvoltarea de aplicații Android este foarte flexibilă putând implementa un lucru folosind mai multe metode. Flexibilitatea se poate observa în faptul că partea de interfață grafică poate fi separată de partea de cod care implementează funcționalitățile interfeței grafice. Astfel pentru a dezvolta interfața grafică se crează fișiere de tip XML în care se definesc elementele grafice, cât și proprietățile lor, cum ar fi culoare, dimensiune sau poziționarea sa pe ecran.

Pentru această aplicație am creat 9 fișiere XML, reprezentând layout-urile interfeței grafice: `login_fragment.xml`, `activity_main.xml`, `activity_friends.xml`, `activity_groups.xml`, `chat_activity.xml`, `friend_row.xml`, `group_row.xml`, `chat_row.xml` și `message_row.xml`.

Primul dintre ele este `login_fragment.xml`, care este folosit pentru fereastra de autentificare în aplicație. Acesta este format dintr-un buton de autentificare, buton pus la dispoziție de API-ul de Facebook, un element de tip `TextView` folosit pentru a afișa un mesaj de întâmpinare după ce procesul de autentificare se termină cu succes și un obiect

de tip `ProfilePictureView`, care este oferit tot de API-ul de Facebook, afișând imaginea de profil a utilizatorului proaspăt autentificat. Acestea sunt parte a unui `LinearLayout` cu orientarea verticală.

Următorul este `activity_main.xml`, în care este inclus fragmentul de autentificare. Conține un `framelayout` care ajută la definirea și includerea fragmentului de autentificare în interfața grafică.

Urmează `activity_friends.xml`, care conține doar un element de tip `RecyclerView` și este folosit pentru a afișa pe ecran o listă cu toți prietenii care folosesc această aplicație. Un rând din listă este reprezentat printr-un layout compus dintr-un element de tip `ProfilePictureView`, care afișează imaginea de profil al prietenului și un element de tip `TextView` folosit pentru afișarea numelui prietenului. Prin selectarea unui prieten din această listă se deschide o nouă activitate, care este reprezentată prin fișierul `chat_activity.xml`.

Layout-ul `chat_activity.xml` este compus dintr-o parte folosită pentru chat și o parte folosită pentru gestionarea de grupuri. Partea de chat conține un element `ProfilePictureView` pentru imaginea de profil al prietenului, un buton, care prin apăsarea lui se trimite răspunsul completat în `EditText`, persoanei de la care a venit mesajul și un buton "Initiate conversation", care prin apăsarea lui se deschide aplicația nativă de Facebook Messenger. Astfel se poate iniția o conversație.

Fereastra de chat este reprezentată și ea printr-un obiect `RecyclerView`, mesajele fiind afișate separat și în ordinea în care au fost create.

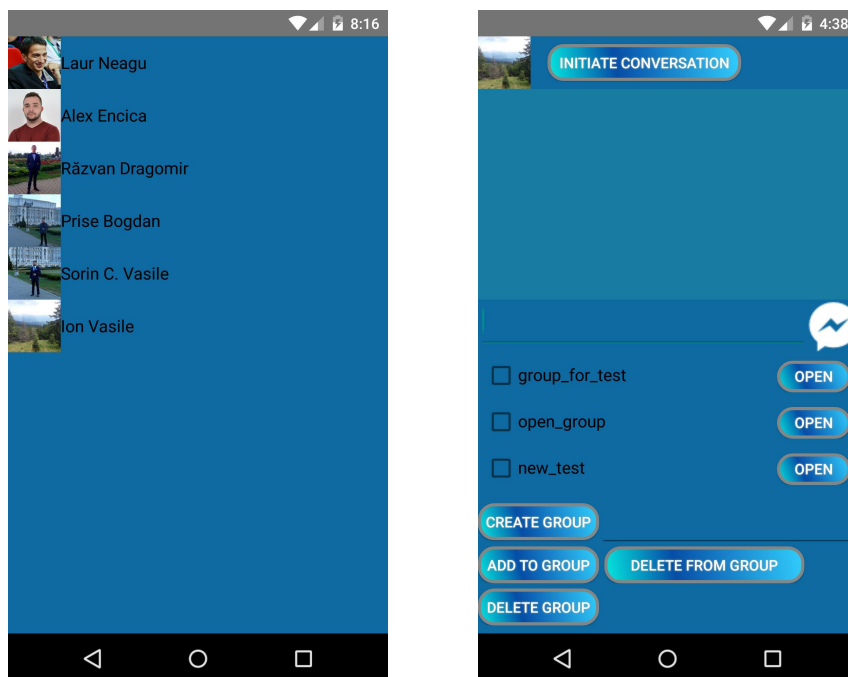


Figura 5.1: Interfață grafică

Partea de grupuri are în componență un buton "Create group", care are asociat un câmp de tip `EditText`. În momentul în care se apasă butonul se preia textul din câmpul text și se crează un nou grup al aplicației care va avea numele regăsit în `EditText`-ul asociat. De asemenea și aici există un element de tip `RecyclerView`, folosit pentru afișarea unei liste de grupuri existente, care se poate actualiza în funcție de operațiile efectuate, de exemplu, în momentul în care se crează un grup nou, lista de grupuri se actualizează imediat ce operația s-a terminat cu succes.

Fiecare rând din obiectul `RecyclerView` folosit pentru lista de grupuri este reprezentat prin trei elemente de grafică. Un element de tip `CheckBox`, folosit pentru selectarea grupului asupra căruia urmează să se efectueze o acțiune, un `TextView` pentru afișarea numelui grupului și un buton pe care dacă se apasă se va deschide o nouă fereastră în care se regăsesc postările din grupul selectat.

Mai sunt prezente butoane pentru adăgare utilizator în grupul selectat sau ștergere din grupul selectat. Persoana care este adăugată sau eliminată din grup este cea selectată din lista de prieteni prezentată anterior. Butonul "Delete group" este folosit pentru ștergerea în întregime a unui grup.

Layout-ul `activity_groups.xml` conține un obiect `RecyclerView` pentru a afișa lista postărilor din grupul selectat. Tot aici se regăsește și un buton care are asociat un câmp text, folosit pentru a posta un nou mesaj.

Pentru implementarea elementelor de tip `RecyclerView` este necesară crearea de mai multe clase. O clasă care va reprezenta un obiect de tip `Adapter` ce se va ocupa de inițializarea listei din interfața grafică. La crearea acestui obiect trebuie pasate ca argumente o listă care conține datele ce vor fi afișate și contextul în care vor fi afișate. De asemenea, ascultătorii de tip apăsare pe un rând al listei sau apăsare a unui anumit element dintr-un rând, sunt definite tot în această clasă.

O altă clasă reprezintă un obiect de tip `ViewHolder`, care va conține elementele grafice folosite. Un rând din listă reprezintă un obiect care conține informațiile ce vor fi completate pe acel rând.

5.1.2 Autentificare

Modulul de autentificare este reprezentat printr-o clasă Java, `FacebookLoginFragment`, care extinde clasa `Fragment`. Procesul de autentificare presupune, completarea câmpurilor user și parolă din interfața grafică pusă la dispoziție de Facebook SDK prin butonul de autentificare. În cazul în care autentificarea a fost efectuată anterior, numele de utilizator și parola acestuia sunt preluate din sistemul de operare, mai exact din secțiunea "Accounts".

Dacă datele de autentificare sunt corecte, se va obține un token de acces valid, căruia îi sunt asociate și permisiunile necesare utilizării normale a aplicației. Cum am spus și în secțiunea 3.2.1, pentru ca procesul de autentificare să funcționeze, trebuie să avem creată o aplicație pe platforma Facebook, inclus Facebook SDK în proiect, generat identificatorul aplicației și adăugat în codul aplicației și generată o cheie pentru profilul dezvoltatorului.

Autentificarea începe în momentul în care se apasă butonul din interfața grafică. Butonul acesta se ocupă și de permisiuni, dar urmărește și starea în care se află procesul. Astfel la terminarea procesului, se afișează un mesaj în funcție de rezultatul autentificării.

În metoda `onCreate` a fragmentului trebuie să se inițializeze Facebook SDK și un obiect de tip `CallbackManager` folosit pentru gestionarea operațiilor de tip `FacebookCallback`. De asemenea, tot aici, datorită faptului că token-urile de acces nu sunt permanente, pot expira în orice moment, se verifică și validitatea token-ului. Acest lucru se verifică folosind un obiect de tip `AccessTokenTracker` și unul de tip `ProfileTracker`. Astfel în cazul în care se schimbă token-ul de acces în timp ce aplicația rulează, obiectul `AccessTokenTracker` va informa `ProfileTracker`-ul de schimbare, astfel, codul din metoda suprascrisă, `"onCurrentProfileChanged"`, actualizează id-ul utilizatorului. Deci având în orice moment token-ul de acces actualizat, nu se vor sesiza schimbările din spate.

Elementele de grafică și parametrii pentru butonul de autentificare se inițializează în metoda `"onViewCreated"`. Se inițializează obiectele de tip `ProfilePictureView` și `TextView` asociindu-le identificatorii din layout-ul de autentificare. Pentru butonul de autentificare, se setează fragmentul curent, se setează un vector de permisiuni și se înregistrează apelul cererii de autentificare. În momentul în care procesul de autentificare se finalizează cu succes, se verifică încă odată validitatea token-ului de acces și se pornește o nouă activitate, cea pentru lista de prieteni.

5.1.3 Listă de prieteni

Prietenii care folosesc aplicația sunt afișați într-o listă de tip `RecyclerView` și este obținută cu ajutorul unei interogări a platformei Facebook prin intermediul Graph API. În metoda `onCreate` a activității `FriendsActivity` se așteaptă ca aceasta să fie invocată prin intermediul unui obiect de tip `Intent`. Dacă activitatea este invocată, se crează un obiect de tip `GraphRequestBatch` prin intermediul căruia se realizează comunicația între aplicație și platforma Facebook.

Cererea pentru lista de prieteni este implementată în metoda `newMyFriendsRequest`, pusă la dispoziție de către Graph API. Pentru ca solicitarea să fie validă această metodă primește ca parametru token-ul de acces al utilizatorului.

Informațiile primite de la Facebook sunt în format JSON și conțin numele prietenilor și identificatorii acestora. În metoda `onCompleted` a cererii se parsează datele recepționate și se populează un obiect de tip `List` care va conține perechi `"nume", "identificator"`. În momentul în care datele sunt gata de afișat, se inițializează obiectul `RecyclerView` cu lista de prieteni.

5.1.4 Grupuri

Grupurile de Facebook sunt create pentru a da posibilitatea persoanelor care au aceleași interese să se reunească într-o comunitate restrânsă de utilizatori. Acestea pun la dispoziție membrilor un spațiu special de comunicare, care poate fi accesat doar de cei care fac parte din grup. Grupurile pot fi de trei feluri. Pot fi publice, adică oricine poate deveni membru dacă face o cerere, la care poate să răspundă oricare membru sau

dacă primește o invitație, trimisă tot de un membru oarecare. Conținutul celor publice poate fi vizualizat și de persoanele din afara grupului. Grupurile închise oferă aceleași privilegii ca și cele publice, doar că, informațiile împărtășite pe grup nu se pot citi din afara acestuia. Al treilea tip de grup este cel secret. În grupurile secrete pot intra doar utilizatorii care primesc invitații de la cei care fac parte deja din grup. De asemenea conținutul grupului nu poate fi văzut în afara lui.

O altă clasificare a grupurilor de Facebook este după modalitatea de creare. Ele pot fi create de aplicații care nu sunt dezvoltate de cei de la Facebook sau create de aplicațiile native ale platformei. Aplicațiile care nu sunt proprietatea platformei rețelei de socializare pot crea grupuri ale aplicației sau grupuri pentru jocuri. Membrii acestor grupuri sunt gestionati doar de aplicația care a creat grupurile. Astfel, în această proiect, în lista de grupuri afișată se regăsesc doar acele grupuri care au fost create de această aplicație.

Listarea grupurilor

Pentru a obține aceste grupuri se face o cerere prin intermediul Graph API, folosind un obiect de tip `GraphRequest`. Acesta trebuie să primească parametrii precum token-ul de acces al aplicației, metoda HTTP folosită, în acest caz `HTTPMethod.GET`, și calea pentru informațiile dorite. Calea este de forma "[identificatorul_aplicației]/groups". Pentru că folosim token-ul de acces al aplicației, nodul din graful rețelei de socializare cerut este reprezentat de identificatorul aplicației, cel obținut folosind pașii din secțiunea 3.2.1. În metoda `onCompleted` a cererii este primită lista de grupuri sub formă de JSON. Aceste informații sunt parsate și este populat obiectul care reține grupurile sub formă de perechi `identificator_grup`, `nume_grup`. Tot aici, după ce datele sunt gata de afișare, se actualizează interfața grafică.

Crearea

Pentru a crea un nou grup, se trimite o solicitare de tip POST tot cu ajutorul unui obiect `GraphRequest`, folosind metoda oferită de Graph API, `newPostRequest`. Acestei metode i se pasează ca argumente, token-ul de acces al aplicației și calea din graful rețelei. Și de această dată calea este de forma "[identificatorul_aplicației]/groups", dar cererii îi mai este asociate și alte date, precum numele pe care o să îl aibă noul grup, identificatorul administratorului, dacă se dorește setat acest lucru și informații despre tipul grupului, cum ar fi tipul public. Datele primite în urma acestei solicitări vin tot sub formă de JSON și conțin identificatorul noului grup creat. După parsarea datelor, se actualizează lista de grupuri din interfața grafică.

Adăugarea de membri

Adăugarea de persoane în grup se realizează tot prin intermediul unei cereri, folosind Graph API. Pentru ca solicitarea să fie validă, metodei `newPostRequest` îi trebuie transmise token-ul de acces al aplicației și calea din graful rețelei, care va fi un șir de caractere

de forma "[identificatorul_grupului]/members". Acestei cereri, îi va fi atașat parametrul "member" având ca valoare identificatorul persoanei care urmează să fie adăugată în grup.

Ștergerea de membri

Ștergerea de persoane din grup se face asemănător cu adăugarea lor. Și aici este nevoie de token-ul de acces al aplicației, iar calea din graful rețelei este la fel, adică "[identificatorul_grupului]/members". Diferența vine din faptul că este specificată metoda HTTP folosită. În acest caz, HttpMethod.DELETE. Parametrul atașat cererii este tot "member" și are ca valoare tot identificatorul utilizatorului care urmează să fie șters.

Ștergerea de grupuri

Grupurile create de aplicației pot fi șterse și ele, dar condiția este ca acestea să nu mai aibă niciun membru. Astfel, dacă un utilizator dorește să șteargă un grup, trebuie mai întâi să elimine toți membrii din acesta. Astfel printr-o cerere de tip GraphRequest, se obține lista membrilor. Pentru această solicitare, este nevoie de token-ul de acces al aplicației, iar calea din graful rețelei este "[identificatorul_grup]/members" și de specificarea metodei folosite, adică HttpMethod.GET. Lista este întoarsă sub formă de JSON, este parsată și pentru fiecare membru se folosește metoda de ștergere al unei persoane prezentată mai sus. După această operație se actualizează și lista de grupuri din interfața grafică.

Postarea de mesaje text

Pentru postarea de mesaje text în grup se folosește din nou GraphRequest cu metoda newPostRequest, care va primi ca argumente, token-ul de acces al aplicației și calea din graful rețelei de socializare, "[identificatorul_grupului]/feed". Acestei cereri îi este atașat parametrul "message" care va avea ca valoare mesajul text care urmează să fie postat. Aceste postări apar pe grup, având ca autor această aplicație.

Listarea postărilor

Postările regăsite pe grupuri pot fi preluate în aplicație folosind tot Graph API. Se trimite o solicitare de tip GraphRequest care primește token-ul de acces curent al aplicației, calea din graful rețelei, "[identificatorul_grupului]/feed" și metoda HTTP folosită, HttpMethod.GET. Datele primite în urma acestei cereri sunt sub formă de JSON. Acestea sunt parsate, căutându-se după cuvântul cheie "message".

5.1.5 Trimitere - Recepție mesaje

Schimbul de mesaje folosind infrastructura Facebook, nu se mai poate implementa în aplicațiile care folosesc instrumentele puse la dispoziție de această platformă. Această

idee, de a bloca dezvoltatorii să creeze aplicații pentru schimbul de mesaje prin intermediul rețelei de socializare, a fost adoptată în acest an, începând cu 30 aprilie 2015. Decizia de a opri suportul pentru schimbul de mesaje private a fost luată datorită numărului foarte mare de aplicații robot care nu făceau decât să deranjeze prin trimiterea de mesaje. Astfel a trebuit să găsim o modalitate de a evita folosirea SDK-ului de la Facebook.

Soluția găsită se bazează pe gestionarea notificărilor din sistemul de operare Android. Toate notificările primite pe un anumit sistem, pot fi interceptate de o aplicație Android. Acest lucru este posibil, doar dacă utilizatorul acordă privilegii de acces la notificări pentru aplicație. Pe lângă acest fapt, pentru o funcționare corectă este necesară prezența aplicației native de Facebook Messenger pentru că inițierea de conversații nu se poate face fără aceasta.

În momentul în care un utilizator dorește să poarte o discuție privată și criptată cu un alt utilizator, acesta trebuie să inițieze conversația din aplicația nativă de Facebook Messenger. Astfel, utilizatorul de la celălalt capăt al canalului de comunicație va primi o notificare de la "com.facebook.orca" care va conține mesajul pe care trebuia să îl primească. Având această notificare se poate răspunde la mesaj pe baza datelor primite prin intermediul notificării. Răspunsul va fi primit tot sub formă de notificare, deci se poate trimite înapoi un mesaj partenerului de comunicație tot pe baza notificărilor. Mesajul de inițiere va fi întotdeauna necriptat și va fi un mesaj de tip atenționare, a faptului că cineva dorește să poarte o conversație criptată.

Notificările din sistemul de operare sunt interceptate prin intermediul unui serviciu, NotificationListenerService. Acesta implementează o metodă de a extrage notificări dintr-un obiect de tip StatusBarNotification. Din acest obiect se poate obține numele pachetului care a produs notificarea. În acest proiect ne interesează doar pachetul "com.facebook.orca". Acesta este folosit de Facebook la schimbul de mesaje private.

Astfel, având notificarea prezentă în bara de stare a sistemului, putem crea un obiect de tip WearableExtender. Acesta este folosit pentru a extrage din notificare informațiile suplimentare, cum ar fi, acțiuni sau pagini existente în notificare. Notificarea preluată din bara de acțiuni conține un câmp extra în care regăsim informații precum mesajul text primit, numele transmitătorului sau identificatorul notificării. Tot de aici putem extrage și eticheta notificării sau conținutul intenției care va trebui să gestioneze datele recepționate.

În momentul în care o notificare apare în bara de acțiuni, aceasta se extrage și se transmite mai departe cu ajutorul unui obiect EventBus. Acesta transmite informațiile preluate, activității care gestionează acest tip de evenimente. În acest caz, activitatea se numește ChatActivity.

În ChatActivity se implementează metoda onEvent, care va executa codul doar atunci când o notificare a sosit. Aici se verifică numele pachetului pe care îl folosește notificarea. Acesta trebuie să fie "com.facebook.orca" pentru ca ne interesează doar datele primite de la Facebook Messenger. Deoarece, metoda onEvent rulează în fundal, pentru a actualiza interfața grafică trebuie să folosim metoda runOnUiThread pentru codul de actualizare. Mesajul primit se găsește în câmpul "android.text", iar numele transmitătorului în câmpul "android.title".

Pentru a răspunde unui mesaj, este nevoie de un obiect `RemoteInput`, ce ajută la specificarea datelor care vor fi preluate de la aplicație și pasate unei intenții. Deoarece, în cazul acestui proiect nu se crează notificări, ci doar se răspunde la ele pe baza datelor primite, nu trebuie să construim noi obiecte de tip `RemoteInput`, pentru că acestea vin odată cu notificările. Trebuie doar modificate datele transmise intenției.

Astfel, în câmpul extra al notificării se adaugă mesajul de răspuns. Când răspunsul este gata de transmis, se apelează metoda `addResultsToIntent` care va primi ca argumente, obiectul `RemoteInput` modificat, o nouă intenție care va gestiona transmisia și câmpul extra care va conține mesajul. După toate acestea, se poate trimite mesajul de răspuns, folosind metoda `send` a intenției care gestionează această notificare.

Acest serviciu, este pornit, doar dacă aplicația este în activitatea `ChatActivity`. Deci notificările sunt gestionate doar dacă ambii parteneri de conversație au activat serviciul. În metoda `onResume` a activității trebuie verificat dacă serviciul este activ sau nu. În cazul în care serviciul nu este activ, se încearcă activarea lui. În cazul în care activitatea este oprită, o să se oprească și serviciul. Acest lucru se face în metoda `onDestroy` a activității.

5.2 Integrarea criptării transmisiei de date

Criptarea mesajelor transmise folosește algoritmul AES, iar cheile sunt simetrice, adică și criptarea și decriptarea este realizată pe baza aceluiași șir de caractere. Pentru a obține cheia secretă se utilizează două șiruri de caractere. Unul ce reprezintă cheia în clar și unul care este folosit la inițializarea unui obiect de tip `Cipher`.

Bibliotecile Java pun la dispoziție metode de a implementa partea de criptare a datelor, ușurând foarte mult munca depusă pentru acest proces. Astfel, se poate folosi un obiect de tip `Cipher` pentru obținerea unui cifru dintr-un șir de octeți. Acesta nu poate fi instanțiat direct, ci printr-o metodă numită `getInstance`. Metoda primește ca argument un șir de caractere de forma "algorithm/mod/padding", în cazul acestei aplicații se folosește "AES/CBC/PKCS5Padding".

Algoritmul folosit este AES, care reprezintă un standard ce poate procesa blocuri de date de 128 de biți, folosind chei simetrice de tip cifru de lungime 128, 192 sau 256 de biți. Fiecare intrare și ieșire a acestui algoritm reprezintă o secvență de 128 de biți de 1 sau 0. [11]. Această aplicație utilizează o cheie simetrică de 128 de biți.

Modul CBC, reprezintă înlănțuirea blocurilor de cifru. Pentru acest mod este necesar un vector de inițializare. În criptarea care folosește CBC, primul bloc este format prin aplicarea funcției logice SAU-exclusiv între primul bloc de text clar și vectorul de inițializare. Primul bloc de text criptat este obținut prin aplicarea algoritmului de criptare pe blocul rezultat din procesul anterior. Următorul bloc este obținut prin aplicarea funcției SAU-exclusiv între primul bloc criptat și următorul bloc de text în clar și așa mai departe [10]. Procesul de obținere a blocurilor este prezentat în figura 5.2.

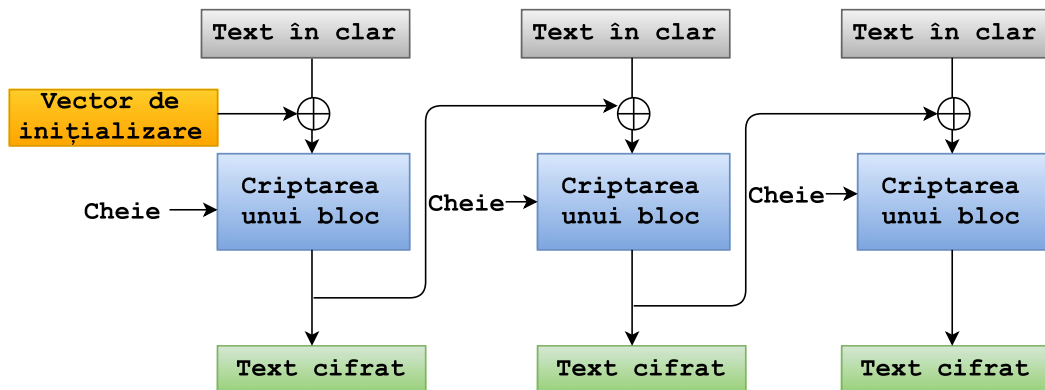


Figura 5.2: Criptarea folosind CBC

Decriptarea în modul CBC se face aplicând funcția de decriptare a obiectului Cipher pe primul bloc de text criptat. Între blocul rezultat și vectorul de inițializare se aplică funcția logică SAU-exclusiv, rezultând primul bloc de text clar. Următorul bloc este obținut prin aplicarea funcției de decriptare pe al doilea bloc, iar între rezultat și primul bloc criptat se aplică tot funcția logică SAU-exclusiv [10]. Procesul de obținere a blocurilor de text în clar este reprezentat în figura 5.3.

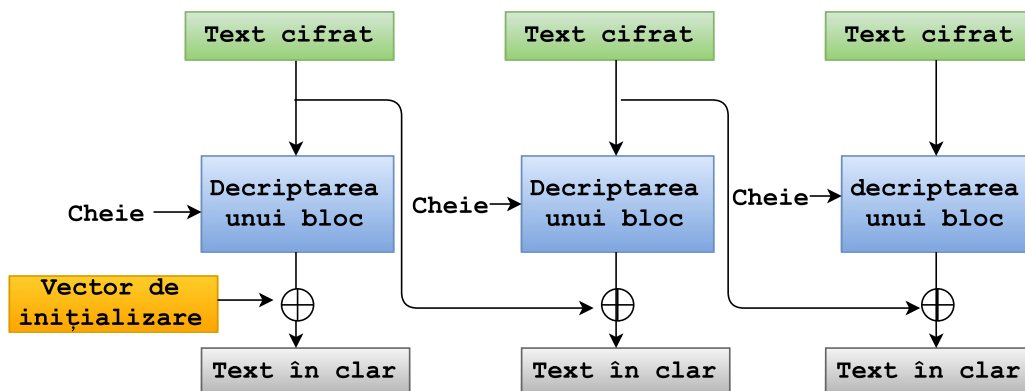


Figura 5.3: Decriptarea folosind CBC

PKCS5Padding, reprezintă o schemă de formatare a unui text, astfel încât lungimea acestuia să fie multiplu de opt. Trebuie să fie multiplu de opt, deoarece un bloc de text are lungimea de opt octeți. Este necesară această metodă de completare a textului până dimensiunea lui este multiplu de opt, deoarece se folosește modul CBC, care se bazează pe criptarea înlănțuită a blocurilor de text.

Criptarea și decriptarea unor date, presupune ca acestea să fie reprezentate sub formă de octeți, în acest caz, vectori de octeți. Deoarece informațiile sunt transmise folosind protocolul HTTP, datele trimise sau recepționate sunt sub formă de șiruri de caractere. De aceea, în această aplicație se va folosi codificarea Base64 a datelor ce va fi detaliată în continuare [8].

În codificarea Base64, 3 octeți de intrare vor fi reprezentați prin 4 octeți [7]. Dacă se dorește codificarea unui vector de n octeți va rezulta un șir de caractere a cărui dimensiunea este reprezentată în următoare schemă:

$$\text{octeți}[n] \rightarrow \text{caractere} \left[\left\lceil \frac{4 * n}{3} \right\rceil \right]$$

De exemplu, pentru $n = 32$, rezultatul este 42.6, care este rotunjit superior la cel mai apropiat multiplu de 4, deci șirul de caractere va avea lungimea de 44 de octeți.

5.2.1 Criptarea

Pentru criptare, se crează un obiect de tip Cipher prin metoda getInstance care primește ca argument șirul de caractere "AES/CBC/PKCS5Padding", după cum se poate observa în secțiunea de cod 5.1. Se instanțiază un obiect SecretKeySpec din cheia de criptare. Șirurile de caractere nu pot fi procesate direct, ele trebuie convertite la un vector de octeți, acest lucru făcându-se cu ajutorul metodei getBytes, căruia se specifică setul de caractere, în acest caz "UTF-8".

```

1 Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
2 SecretKeySpec key = new SecretKeySpec(encryptionKey.getBytes("
    UTF-8"), "AES");
3 cipher.init(Cipher.ENCRYPT_MODE, key, new IvParameterSpec(IV.
    getBytes("UTF-8")));
4 byte[] encryptedBytes = cipher.doFinal(plainText.getBytes("UTF
    -8"));
5 String result = android.util.Base64.encodeToString(
    encryptedBytes, android.util.Base64.DEFAULT);

```

Listing 5.1: Cod folosit pentru criptare

Cifrul trebuie inițializat, acest lucru realizându-se prin metoda init. Aceasta primește ca argumente, modul de inițializare, aici ENCRYPT_MODE, obiectul SecretKeySpec și un obiect de tipul IvParameterSpec ce reprezintă vectorul de inițializare. Este nevoie de acest vector pentru că se folosește modul CBC, care necesită un astfel de obiect pentru a procesa primul bloc din text. În această aplicație vectorul de inițializare are valoarea "AAAAAAAAAAAAAAAAAAAA", dar poate avea oricare altă valoare.

Rezultatul criptării este un vector de octeți, returnat de metoda doFinal a cifrului. Această metoda primește ca argument textul care urmează a fi criptat, dar sub formă de vector de octeți. Pentru a obține un format de afișare a vectorului de octeți se folosește codificare Base64.

5.2.2 Decriptarea

Pentru ca decriptarea să funcționeze, trebuie să se primească un șir de caractere care reprezintă criptarea unui text clar, dar folosind aceeași cheie de criptare, același algoritm și aceeași metodă de codificare, adică Base64. Codul folosit pentru decriptare este prezentat în secțiunea de cod 5.2.

```
1 byte[] cipherText = android.util.Base64.decode(text.getBytes("
    UTF-8"), android.util.Base64.DEFAULT);
2 Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
3 SecretKeySpec key = new SecretKeySpec(encryptionKey.getBytes("
    UTF-8"), "AES");
4 cipher.init(Cipher.DECRYPT_MODE, key,new IvParameterSpec(IV.
    getBytes("UTF-8")));
5 String result = new String(cipher.doFinal(cipherText));
```

Listing 5.2: Cod folosit pentru decriptare

Aplicația primește un șir de caractere ce trebuie decodificat și convertit la un vector de octeți prin metoda Base64, folosind setul de caractere UTF-8. După această decodificare se crează obiectul de tip Cipher tot prin getInstance("AES/CBC/PKCS5Padding"). Și aici este nevoie de SecretKeySpec, format pe baza aceleiași chei, ca la criptare. Se inițializează cifrul, la fel ca la partea de criptare, doar ca aici metoda specificată este DECRYPT_MODE.

Rezultatul este întors tot sub formă de vector de octeți, deci este necesară conversia la șir de caractere, pentru a putea fi afișat în interfața grafică.

Capitolul 6

Testarea și evaluarea soluției

Testarea a fost realizată pe dispozitive care folosesc Android 5.0.2. Versiunea minimă de Android pe care poate fi rulată această aplicație este Android 5.0.0 și este necesară instalarea aplicației Facebook Messenger.

6.1 Autentificare

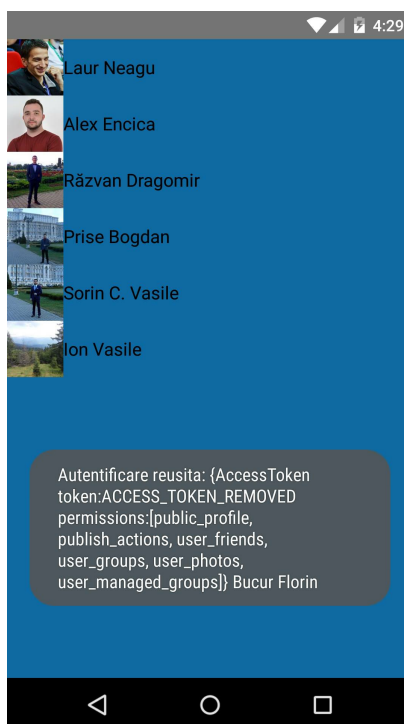


Figura 6.1: Rezultatul autentificării

Faptul că fereastra cu lista de prieteni care folosesc aplicația a apărut pe ecran asigură terminarea procesului de autentificare.

6.2 Trimitere - Recepție mesaje

Primul pas pentru schimbul de mesaje este reprezentat de selectarea unui prieten și inițierea conversației care s-a realizat prin apăsarea butonului "Initiate Conversation". Acest buton a deschis aplicația nativă, Facebook Messenger din care s-a trimis mesajul "Hello". Așa cum se poate vedea și în Figura 6.2, primul mesaj este cel de inițiere a conversației.

La receptor se primește o notificare, pe baza căreia se răspunde cu primul mesaj criptat din conversație. Acesta se vede și în Figura 6.2, el fiind "First message".

Răspunsurile au fost completate în câmpul text care se află sub lista de mesaje, iar prin apăsarea butonului din dreapta câmpului text, mesajul se preia de către aplicație, este criptat și trimis mai departe prin rețeaua socială, destinatarului.

Schimburile de mesaje sunt salvate pe platforma socială criptate, așa cum se poate observa în fereastra din dreapta a Figurii 6.2, iar cheile de criptare sunt setate în codul aplicației.

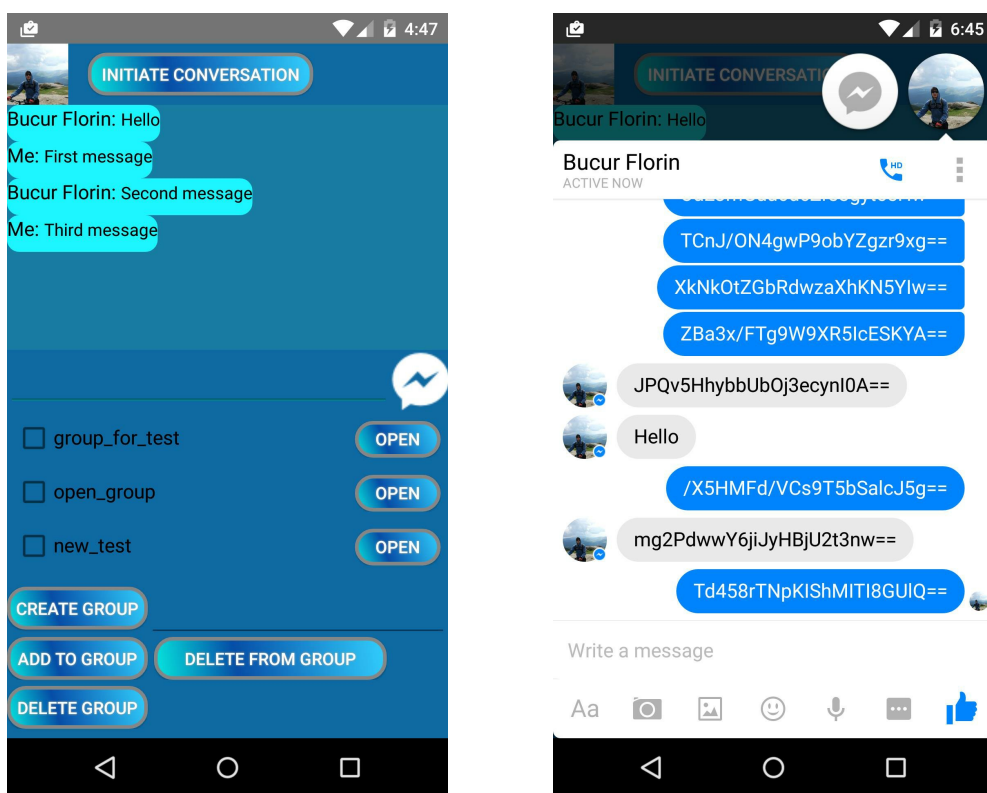


Figura 6.2: Schimbul de mesaje

6.3 Grupuri protejate

Lista de grupuri se poate vedea în Figura 6.2. În acest exemplu, s-a apăsât pe buton "Open" al grupului denumit "open_group", acest lucru putând fi observat în partea de sus a Figurii 6.3, unde este afișat numele grupului selectat.

După cum se poate observa în figura 6.3, mesajele postate sunt afișate sub formă de listă. Aplicația preia postările din baza de date a platformei sociale, le decriptează și le afișează.

Pentru postarea mesajului "Third message", s-a completat câmpul text din dreapta butonului "POST TO GROUP", iar prin apăsarea lui aplicația preia mesajul, îl criptează și îl trimite platformei sociale.

Postările sunt stocate în forma criptată în cadrul rețelei de socializare, așa cum se poate observa în fereastra din dreapta Figurii 6.3.

Mesajele au fost criptate cu aceeași cheie ca și mesajele din conversațiile protejate, acestea fiind setate direct în codul aplicației.

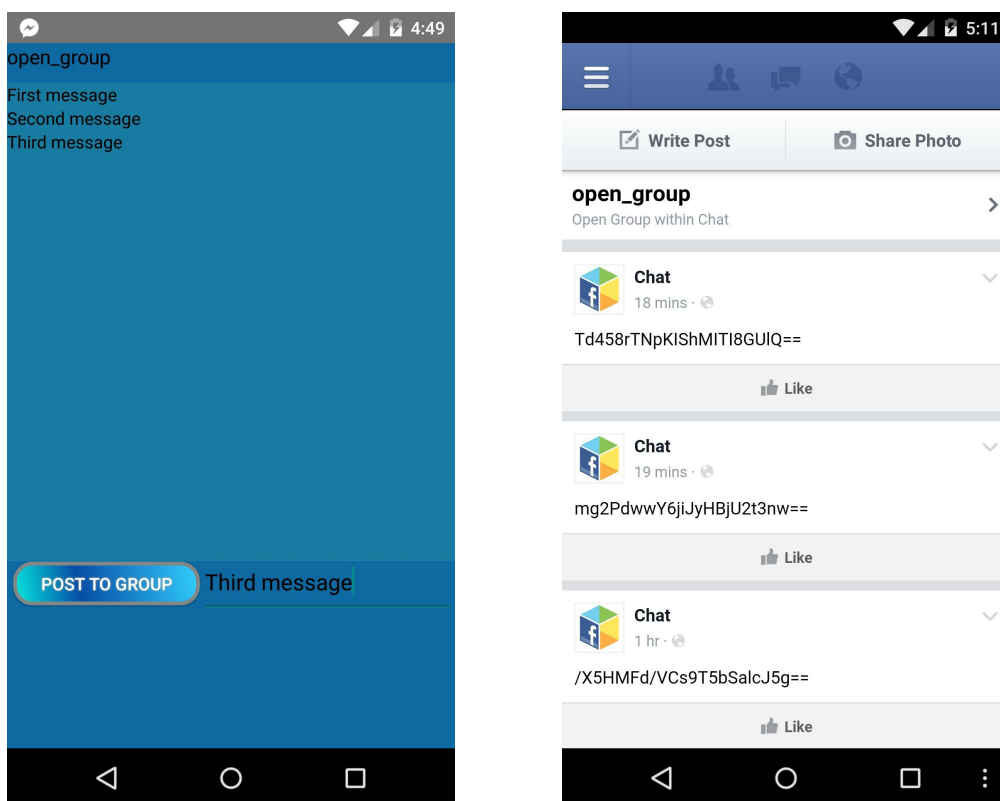


Figura 6.3: Mesajele criptate de pe un grup

6.4 Evaluarea rezultatelor

Pe lângă testarea funcțională a aplicației, am încercat să demonstrez faptul că procesul de criptare sau de decriptarea în timp real al unor mesaje de dimensiune mică, nu afectează în mod evident schimbul de mesaje.

Astfel, am măsurat timpul necesar pentru decriptarea, respectiv criptarea unor mesaje cu lungimi diferite, dar nu foarte mari, deoarece reflexul utilizatorilor este de a trimite mesaje cât mai scurte.

Tabela 6.1: Timpuri de criptare / decriptare

Mesaj	Octeți necriptați	Octeți criptați	Timp criptare	Timp decriptare
Acesta este un mesaj scurt.	27	44	0.4 ms	5.4 ms
Acesta este un mesaj puțin mai lung decât primul.	49	76	0.39 ms	0.25 ms
Acesta este un mesaj mai lung și decât primul și decât al doilea aproape cât amandoua împreună.	95	153	0.41 ms	0.38 ms
Acesta este un mesaj mai lung și decât primul și decât al doilea aproape cât amandoua împreună. Acesta este un mesaj mai lung și decât primul și decât al doilea aproape cât amandoua împreună. Acesta este un mesaj mai lung și decât primul și decât al doilea aproape cât amandoua împreună.	285	388	4.4 ms	5.2 ms

În Tabelul 6.1 sunt prezente rezultatele măsurătorilor pentru 3 mesaje. În continuare o să interpretăm datele.

Primul rând din tabel reprezintă un mesaj cu lungimea de 27 de octeți, mai exact 216 biți. Deoarece mesajul trebuie să fie multiplu de dimensiunea cheii, respectiv 128 de biți, acesta va fi completat până la 256 de biți cu terminatorul de șir "\0". Astfel, mesajul transmis metodei care se ocupă de criptare, va avea 32 de octeți. Mesajul rezultat în urma criptării va avea tot 32 de octeți, dar nu este într-un format care poate fi afișat sau transmis prin HTTP, el fiind doar un vector de octeți. Pentru transformarea din vector de octeți în șir de caractere se folosește codificarea Base64. În urma codificării rezultă mesajul final, criptat, care poate fi trimis platformei sociale, cu dimensiunea de 44 de octeți.

Timpii de criptare, respectiv decriptare sunt destul de mici și nu sunt diferențe sesizabile între un mesaj de lungime 27 și unul cu lungimea 285. De asemenea, acești timpi sunt dependenți și de încărcarea procesorului la acel moment.

6.4.1 Probleme întâmpinate

Principala problemă întâmpinată a fost faptul că cei de la Facebook nu mai permit dezvoltatorilor să își creeze propriile lor aplicații pentru schimbul de mesaje prin intermediul platformei sociale. Datorită acestui fapt am găsit soluția care se bazează pe interceptarea și gestionarea notificărilor din sistemul de operare.

O altă problemă este că nu am găsit o metodă de a posta imagini de pe dispozitiv pe grupurile protejate. Cei de la Facebook permit doar postarea de imagini prin intermediul link-urilor WEB, deci nu se pot posta imagini criptate.

Postările de pe grupurile protejate se fac din partea aplicației, ci nu a utilizatorului, deoarece pe grupurile protejate se poate posta folosind doar token-ul de acces al aplicației, nu al utilizatorului.

Capitolul 7

Concluzii

Securitatea informațiilor este un subiect sensibil care este în continuă mișcare, deoarece de-a lungul timpului, atacatorii găsesc metode noi de a produce pagube sau de a încălca dreptul la intimitate a persoanelor ce utilizează mediul virtual.

În prezent, foarte multe persoane folosesc rețelele de socializare pentru a interacționa în spațiul virtual. Faptul că numărul de utilizatori este mare reprezintă o atracție și o provocare pentru atacatori.

Scopul proiectului a fost de a îmbunătăți securitatea interacțiunilor din cadrul unei rețele sociale, precum Facebook, acest lucru realizându-se printr-o aplicație pentru dispozitivele mobile care folosesc Android ca sistem de operare. Deși implementarea actuală nu oferă toate funcționalitățile puse la dispoziție de Facebook, aceasta poate fi dezvoltată în viitor prin adăugarea treptată de noi funcționalități.

Securitatea interacțiunilor a fost îmbunătățită criptând datele transmise prin intermediul platformei sociale. Obiectivele principale au fost de a proteja mesajele din conversațiile private și cele din cadrul grupurilor de Facebook. La nivel de funcționalitate, aceste obiective au fost atinse.

Soluția de criptare aleasă nu necesită resurse multe, procesul fiind destul de scurt din punct de vedere al timpului. Astfel, faptul că mesajele ce sunt gestionate de aplicație sunt criptate și decriptate nu este un lucru sesizabil pentru utilizator. În plus, pentru implementarea acestui proiect s-au folosit biblioteci și module disponibile în Android care ajută la reducerea timpului de reacție a interfeței grafice la acțiunile realizate de utilizator.

Mesajele sunt stocate pe platforma socială în forma lor criptată, dacă se dorește citirea lor este nevoie de cheia privată folosită la criptare, deci un atacator nu ar trebuie să poată decripta mesajele.

În concluzie, scopul lucrării de a oferi o aplicație pentru interacționarea protejată în cadrul unei rețele de socializare a fost atins.

7.1 Activități viitoare

O activitate viitoare destul de importantă ar fi îmbunătățirea serviciului care se ocupă de gestionarea notificărilor. Acesta ar trebui să ruleze tot timpul cât aplicația este pornită, iar în momentul în care se deschide activitatea folosită pentru schimbul de mesaje să înștiințeze activitatea în cazul în care s-au primit notificări înainte de deschiderea acesteia.

O altă activitate presupune implementarea unei soluții de trimitere automată a cheilor private prin intermediul unor SMS-uri. Acest lucru implică aflarea numărului de telefon al utilizatorilor.

La partea de schimb de mesaje se va descărca, afișa și decripta mesajele trimise până în acel moment.

Pentru partea de grupuri, se va încerca găsirea unor soluții de a posta imagini criptate în cadrul acestora, iar postările să aibă drept autor numele utilizatorului, nu cel al aplicației. De asemenea se va încerca introducerea de noi funcționalități, precum afișarea comentariilor de la postări sau a unei liste cu membrii grupurilor.

În plus, se va modifica interfața grafică pentru ca utilizarea aplicației să fie cât mai ușoară și intuitivă celor care o folosesc.

Bibliografie

- [1] <http://www.xabber.com>.
- [2] <http://beem-project.com/projects/beem/wiki>.
- [3] <http://www.thialfihar.org/projects/apg>.
- [4] <https://play.google.com/store/apps/details?id=com.woodkick.pgpsms&hl=en>.
- [5] <https://play.google.com/store/apps/details?id=com.symantec.pgviewersymantec&hl=en>.
- [6] <http://www.androidcentral.com/best-android-phones>.
- [7] <http://java-performance.info/base64-encoding-and-decoding-performance/>.
- [8] Mohammad A. AlAhmad, Imad Fakhri Al Shaikhli, and Hanady Mohammad Ahmad. Protection of the texts using base64 and md5. *Journal of Advanced Computer Science and Technology Research*, 2(1), 2012.
- [9] Nikita Borisov, Ian Goldberg, and Eric Brewer. Off-the-record communication, or, why not to use pgp. In *Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society*, WPES '04, pages 77–84, New York, NY, USA, 2004. ACM.
- [10] Morris J. Dworkin. Sp 800-38a 2001 edition. recommendation for block cipher modes of operation: Methods and techniques. Technical report, Gaithersburg, MD, United States, 2001.
- [11] Frederic P. Miller, Agnes F. Vandome, and John McBrewster. *Advanced Encryption Standard*. Alpha Press, 2009.