



УНИВЕРЗИТЕТ СИНГИДУМУМ
ФАКУЛТЕТ ЗА ИНФОРМАТИКУ И РАЧУНАРСТВО

Развој крипто модула за размену кључева на Андроид платформи



УНИВЕРЗИТЕТ СИНГИДУНУМ
ФАКУЛТЕТ ЗА ИНФОРМАТИКУ И РАЧУНАРСТВО

УНИВЕРЗИТЕТ СИНГИДУНУМ
ФАКУЛТЕТ ЗА ИНФОРМАТИКУ И РАЧУНАРСТВО
БЕОГРАД, ДАНИЈЕЛОВА 32

Кандидат: Петар Степановић
Број индекса: 2010/200405
Смер: Информатика и рачунарство

Тема: Развој крипто модула за размену кључева на Андроид платформи

Задатак: Реализовати вишекорисничку апликацију за размену кључева преко Интернета коришћењем изабраних технологија. Дати објашњење коришћених технологија и функционалности реализованог решња.

Датум одобрења рада:
Београд, __. __. ____.

МЕНТОР

Доц. Др Саша Адамовић

ДЕКАН

Драган Цветковић

Развој крипто модула за размену кључева на Андроид платформи

Сажетак: Услед наглог развоја Интернета неопходно долази и до развоја заштите информација преко Интернета. Коришћењем следећих технологија: Android, Java, Google Cloud Messaging развили смо крипто модул за размену кључева преко Интернета са свим функционалностима које су потребне за реализацију ове апликације. У овом раду ће бити објашњене основе поменутих технологија и биће дат преглед апликације која омогућава сигурну размену кључева преко Интернета.

Кључне речи: кључ, апликација, Интернет, заштита, размена

The development of crypto module for key exchange on Android platform

Abstract: Because of the sudden development of the Internet there is urge in increase in information security. By using the fallowing technologies: Android, Java, Google Cloud Messaging We have developed an crypto module for key exchange over the Internet with all the functionalities needed for implementation of this application. Basic of aforementioned technologies will be explained in this paper together with overview of application that enables secure key exchange over the Internet.

Keywords: key, applicaton, Internet, security, exchange

Садржај

Садржај

1 Увод.....	6
1.1 Историја криптографије.....	7
1.1.1 Древна криптографија.....	7
1.2.1 Средњовековна криптографија.....	10
1.3.1 Модерна криптографија.....	11
1.2 Симетрична криптографија.....	13
1.3 Асиметрична криптографија.....	13
2 Коришћене технологије.....	15
2.1 Андроид.....	15
2.2 Јава.....	16
2.1.1 Аутоматско управљање меморијом.....	16
2.2.1 Синтакса.....	16
2.3 <i>Google Cloud Messaging</i>	18
3 Преглед области истраживања.....	19
3.1 Сателитски Сценарио.....	19
3.2 Системи са нултим знањем.....	19
Софтверско решење.....	22
3.3 Циљ пројекта.....	22
3.4 Архитектура целокупног софтверског решења.....	22
3.5 Имплементација.....	22
3.1.1 <i>readBytes</i>	23
3.2.1 <i>getKeys</i>	25
3.3.1 <i>Coding</i>	26
3.4.1 <i>informationReconciliation</i>	27
3.5.1 <i>privacyAmplification</i>	27
4 Комуникација.....	27
4.1 <i>Google Cloud Messaging</i>	27
4.2 Архитектура <i>Google Cloud Messaging</i> сервиса.....	28
4.3 Подешавање <i>Google Cloud Messaging</i> сервиса.....	28

4.4 <i>Smack CCS (XMPP) Server</i>	30
4.1.1 Имплементација CCS/XMPP локалног сервера	31
4.2.1 Пример функција локалног сервера	32
4.3.1 Слање порука.....	34
План за даљи развој	39
Закључак	40
Литература.....	41

1 УВОД

Савремене информационе технологије драстично су промениле данашњи свет рачунарских мрежа, с обзиром на свакодневни раст броја нових корисника Интернета и самих интернет сервиса, повећава се како количина тако и вредност информација па слободно можемо рећи да је информација један од најбитнијих ресурса данашњице. У данашње време Интернет улази у разне аспекте свакодневног живота. Он се користи за комуникацију, трговину, информисање, електронско банкарство, итд. Услед овога Интернет нам олакшава доста тога али исто тако нападачима олакшава да нелегалним нападима дођу до чуваних информација. Обезбеђивањем канала кроз који пролазе наше информације умањујемо не само приватну већ и све чешће економску штету.

За циљну групу која ће користити овај модул изабрани су корисници који користе уређаје на Андроид платформи и који користе апликације за рамену порука преко интернета. Определили смо се за ову циљну групу корисника јер је све већи број корисника који користи апликације за размену порука преко Интернета, како за приватно или необавезно такозвано „ћаскање“, тако и оних који користе ову могућност у пословне сврхе.

У складу са претходно наведеним предностима и манама комуникације преко Интернета за тему овог рада определили смо се за израду крипто модула за размену кључева на Андроид платформи. Апликација пружа корисницима могућност да се региструју како би могли безбедно да комуницирају са другим регистрованим корисницима тако што прво израчунају кључ који ће касније користити за шифровање жељених порука.

Шифровањем информација онемогућавамо нежељеној страни да има увид у садржај наше информације. Шифровање је процес трансформације оригиналне информације (отворени текст) у шифровани податак (шифрат). Обрнут процес шифровању је, дешифровање које на основну шифрата реконструира првобитну нешифровану поруку. У оба процеса поред отвореног текста и шифрата користи се вредност која се назива кључ шифровања. Број симбола који представља кључ тј. његову дужину зависи од шифарског система који се користи и од величине отвореног текста. Постоје два основна типа криптографије симетрична и асиметрична криптографија о којима ћемо детаљније причати у деловима 1.2 и 1.3.

Као што је претходно напоменуто за циљну групу су изабрани корисници који користе уређаје на Андроид платформи. Постоје разни начини и уређаји преко којих можемо да размењујемо поруке али смо се определили за паметне мобилне уређаје због њихове распрострањености и доступности у скоро свакој ситуацији. За Андроид платформу смо се определили јер по неким истраживањима Андроид корисници чине чак 85% укупног броја корисника свих оперативних система на глобалном нивоу.

За израду саме Андроид апликације данас постоји неколико интегрисаних развојних окружења од којих су најпопуларнија Eclipse, Android Studio, IntelliJ IDEA. На слоју комуникације могу се користити разне технологије неке од њих су Java са разним библиотекама које омогућавају лаку комуникацију између сервера и корисника, Python,

PHP итд. За складиштење порука често се користе MySQL, Oracle. Али у овом пројекту за складиштење порука и комуникацију са уређајем користили смо Google Cloud Messaging сервис.

Структура апликације се може поделити на две целине: кориснички део тј. Андроид апликација и комуникациони део тј. локални сервер и *Google Cloud Messaging* сервис. Андроид апликација садржи информације о кориснику које омогућавају комуникацију између два корисника и приступ локалном серверу. Локални сервер садржи кључ који гарантује да се Андроид апликација повеже баш са жељеним сервером и касније повезује Андроид апликацију са *Google Cloud Messaging* сервисом. *Google Cloud Messaging* сервис преузима податке са локалног сервера, складишти их и касније прослеђује између корисника.

Користећи наведене технологије, апликација је реализована у складу са наведеним захтевима а упутство за коришћење апликације дато је заједно са детаљним прегледом њене структуре и сликама свих важнијих делова ове апликације.

1.1 Историја криптографије

Криптографија је наука шифровања која је присутна хиљадама година уназад. Ова наука бави се методама за очување тајности информација. Људи су одувек имали потребу да обезбеде тајност информација тако што су на разне начине шифровали своје информације. Са друге стране постојали су људи који су желели да сазнају или дешифрирају поруке које нису биле намењене њима, често су то били непријатељи у рату, конкуренти у послу итд. У наредна три поглавља проћи ћемо кроз кратак преглед основних историјских епоха развоја и ширења криптографије у људском друштву.

1.1.1 Древна криптографија

Хијероглифи (пиктограми) су коришћени у старом Египту, они су се уклесавали у камене површине и сматрају се најстаријим примером шифровања. Дуго се сматрало да је немогуће дешифровати Хијероглифе, али је откриће и разумевање камена из Розете (*Rosetta Stone*) у деветнаестом веку омогућило дешифровања овог писма. Камен из Розете је заправо камена плоча на којој је био исписан закон на три различита писма: хјероглифима, демотским писмом (упрошћени хијероглифи) и старогрчким писмом на основу кога су дешифровани Хијероглифи.



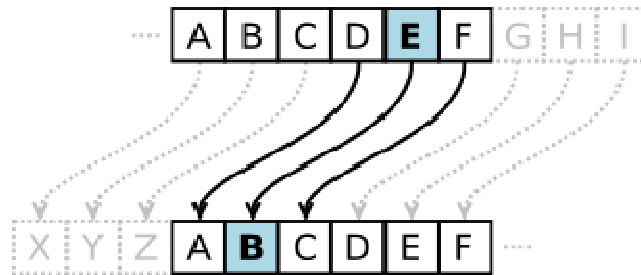
Слика 1: Камен из Розете

Скитал је облик шифровања који се користио у Спарти, старој Грчкој око шестог века п.н.е. Скитал се састојао од цилиндра одређеног пречника око којег је била обмотана пергамент трака. Текст је био исписан на пергаменту обмотаном дуж цилиндра. Метод шифровања био је дизајниран тако да прималац поруке може да прочита поруку тако што би обмотао траку на којој је исписана порука око цилиндра истог пречника. Овакав тип шифара називамо шифре замењивања.



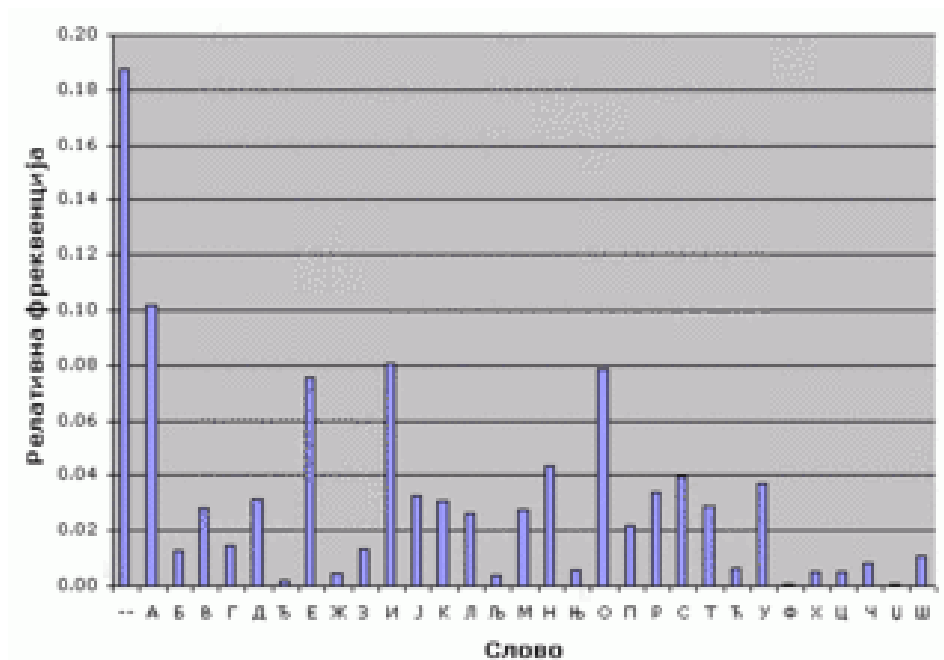
Слика 2: Скитал

Цезаров шифрат се јавља у првом веку н.е, носи овакво име јер га је често користио Јулије Цезар. Овај шифрат ради тако што се свако слово отвореног текста мења одговарајућим словом тог писма, померним за одређени број места. Пошиљаоц и примаоц поруке се унапред договарају за колики број места ће померати слова текста, нпр. три слова уназад.



Слика 3: Цезаров шифрат

Први начин разбијања овог шифрата лежи у томе да се свако слово може заменити само једним истим словом. Одавде долазимо до фреквенције слова, ако фреквенцију слова представимо графички као на слици испод, лако можемо уочити колика је вероватноћа за појављивање одређеног слова. У српском језику најзанимљивија слова су *a, u, o, e* као најзаступљенија и *u, f, h* као најмање заступљена слова.



Слика 4: Дијаграм фреквенције слова у српском језику

Пошто постоји ограничен број помака у сваком писму (нпр. у српском језику је 30) други начин разбијања овог шифрата је коришћење **brutte-force** напада. Ако за исечак шифрата испробамо све могуће помаке доћи ћемо до отвореног текста.

Помак	Могући отворени текст
0	Е Њ Д З Љ П Љ У
1	Ђ Н Г Ж Л О Љ Ђ
2	Д М В Е К Њ К Т
3	Г Љ Б Ђ Ј Н Ј С
4	В Л А Д И М И Р
5	Б К Ш Г З Љ З П
6	А Ј Џ В Ж Л Ж О
...	
27	И Р Ж К Њ Т Њ Ц
28	З П Е Ј Н С Н Х
29	Ж О Ђ И М Р М Ф

Слика 5: Начин разбијања Цезаровог шифрата

1.2.1 Средњовековна криптографија

Како су технике дешифровања постале софистицираније услед сазнања до којих се дошло дешифровањем древних шифрата и проналажењем нових, криптографија је постајала све популарнија у средњем веку. Нагли раст дипломатије доводио је до потребе за преносом поверљивих информација, а самим тим и до употребе криптографије.

Вижернова шифра (*Vigenère cipher*)

Принципи овог шифрата заснивају се на шифрирању текста коришћењем серије Цезарових шифара. Пошто су поруке шифроване Вижерновом шифром потуно другачије у зависности од кључа, чак и ако је трећа страна познавала табелу писма било је изузетно тешко разбити ову шифру. За овај шифрат било је потребно сто година да се развије од почетне замисли до практичне примене. Овоме је додатно отежавала чињеница да су се у то време и даље користиле једноставне шифре замене, а шифровање и дешифровање Вижерновом шифром је било занатно компликованије него једноставним шифрама замене. За шифровање овим шифратом користи се табела писма, позната као *tabula recta*, Вижернов квадрат или Вижернова таблица. Састоји се од алфабета написаног 26 пута у новом реду, а сваки ред је ротиран у лево у односу на претходни, тако да одговара свим могућим комбинацијама Цезарове шифре. У појединачној тачки шифровања, шифрат користи различити алфабет из једног од редова. Ред који ће бити коришћен зависи од понављајућег кључа.

Пример:

Рецимо да је отворени текст који треба да се шифрује следећи:

ATTACKATDAWN

Особа која шаље кључ, бира кључ и понавља га онолико пута док се дужина кључа не изједначи са дужином отвореног текста нпр. особа је изабрала кључ LEMON.

Пример:

LEMONLEMONLE

Сада прво слово отвореног текста (А) шифрујемо користећи алфабет из реда (L) јер је оно прво слово кључа. То се ради тако што се тражи слово на коме се секу ред (L) и колона (А) Вижерновог квадрата, из овога добијамо да је тражено слово, слово (L). Аналогно овоме, за следеће слово користи се следеће слово кључа, слово које се налази у пресеку реда (Т) и колоне (Е) је слово (Х). По овом систему пролазимо кроз цео отворени текст и добијамо следеће резултате.

Пример:

Отворени текст: ATTACKATDAWN

Кључ: LEMONLEMONLE

Шифрат: LXFOPVEFRNHR

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Слика 6: Вижернова таблица

1.3.1 Модерна криптографија

Енигма

Дешифровање шифара помоћу оловке и папира постало је много теже појавом машина за шифровање почетком двадесетог века.

Енигма је име машине за шифровање коју је осмислио Немачки инжењер Артур Шерибус (*eng. Arthur Scherbius*) 1918. године. Главне предности ове машине које су је представљале на тржишту и биле заслужне за њен успех јесу преносивост и поверљивост. Пошто немачка војска још увек није знала да је шифра коју су користили у Првом светском рату била дешифрована чим је била продата прва Енигма, није била свесна да мора да побољша шифру коју користи. Због велике цене Енигме, немачка војска је одустала од њене употребе. Тек када је Немачка открила да је изгубила овај рат управо зато што су Британци успели да дешифрују њихове шифре, схватили су важност шифровања и одлучили да прихвате Енигму.

На први поглед Енигма изгледа као обична писаћа машина. Са предње стране има тастатуру од 26 слова, изнад које се налази исто толики број сијалица, свака од тих сијалица представља једно слово.

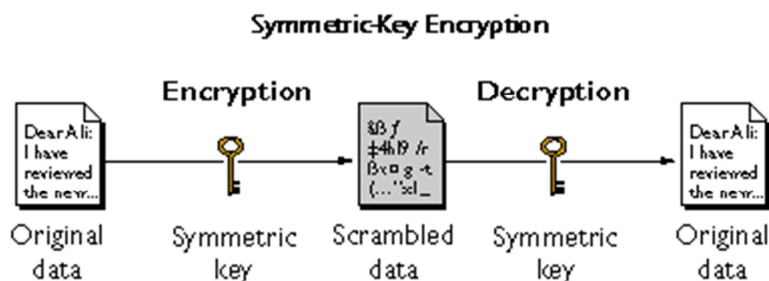
Метод шифровања коришћен на Енигми познат је као полиалфабетски шифрат замене, „кључ“ се састоји од комбинације зупчаника (ротора) познатих као „скремблер“, а на сваком од њих је уписано 26 слова абецеде, и механизам за појединачну замену карактера. Енигма се користи тако што се прво подеси скремблер и након тога се куца отворени (не шифровани) текст на тастатури Енигма машине. Шифрована слова се приказују на табли са лампама. При сваком притиску на тастер ваљци се даље окрећу, што значи да се користи други кључ за сваки карактер. Енигма дешифрује поруке истим кључем којим их шифрује, тако да је дешифровање подједнако комплексно као и шифровање. Немачке снаге унапредиле су Енигму тако што су користили три од пет ротора како би сакрили скремблере и додајући већег броја ротора.



Слика 7: Енигма машина

1.2 Симетрична криптографија

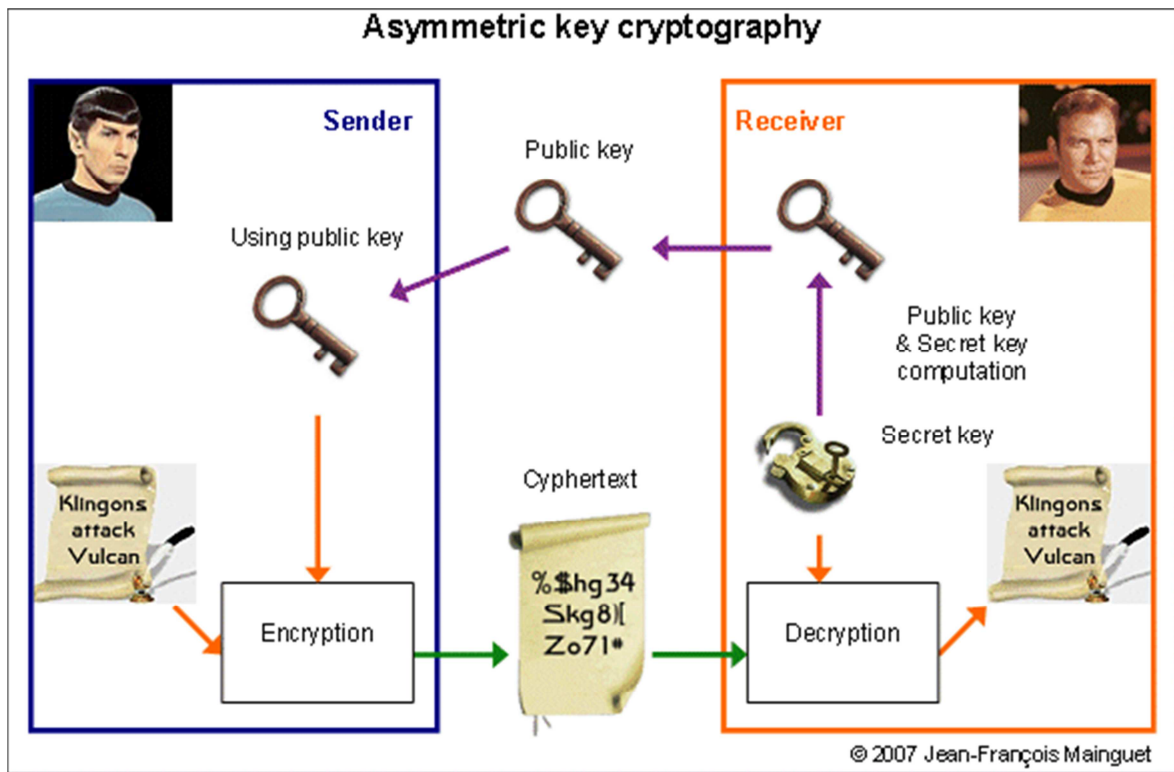
Код симетричне криптографије тј. шифровања користи се исти кључ за шифровање и дешифровање, па је због тога број могућих комбинација веома велик, а самим тим и сигурност оваквог алгоритма. Друга предност симетричне размене кључева је и брзина. Једина недоследност овог протокола је како пренети кључ, јер ако се тајни кључ пресретне, порука се лако може дешифровати. Зато се овај тип шифровања користи у приликама где кључ не делимо и не шаљемо кроз комуникациони канал.



Слика 8: Симетрична криптографија

1.3 Асиметрична криптографија

За разлику од симетричне криптографије, асиметрична користи два кључа који могу бити јавни и приватни. Овај протокол функционише по следећем принципу: у исто време се генеришу оба кључа, приватни и њему одговарајући јавни. Јавни кључ се даје особама које шаљу шифроване податке и помоћу њега шифрују податке које желе да пошаљу. Помоћу свог приватног кључа прималац дешифрује добијене податке. На овај начин сваки прималац има јавни кључ који може дати било коме јер се он користи само за шифровање, а за дешифровање користи свој приватни кључ. Предност овог начина шифровања је у томе што се не мора бринути о томе да ли ће јавни кључ бити компромитован, јер је он већ јавно доступан и помоћу њега се само шифрују подаци.



Слика 9: Асиметрична криптографија

2 КОРИШЋЕНЕ ТЕХНОЛОГИЈЕ

Апликација као целина може се поделити на три основна дела: интерфејс тј. Андроид апликација, локални сервер и Google Cloud Messaging сервис. За израду сваког од ових делова коришћене су различите технологије. Технологије које су коришћене у овом пројекту су следеће:

1. Андроид
2. Java
3. Google Cloud Messaging

2.1 Андроид

Андроид је оперативни систем који је заснован на Линукс језгру, дизајниран је најпре за уређаје са екраном осетљивим на додир као што су паметни телефони и таблет уређаји. Андроид је развила компанија под истим именом „*Android, inc*“. Компанија *Google* је подржавала Андроид, а касније их је и купила 2005. године. Кориснички интерфејс је заснован на директној манипулацији објеката који се налазе на екрану, померање прста по екрану систем читава као улаз и на основу тога се омогућава управљање садржајем на екрану. Додатне функционалности омогућује додатни хардвер као што су жирокоп, акцелерометар, сензор близине. Ови додаци користе се у разним захтевима корисника, један од њих је промена оријентације екрана у односу на положај уређаја.

Андроид је оперативни систем отвореног кода који поседује Апач лиценцу (*Apache License*). Она допушта да се софтвер слободно мења и дистрибуира од стране произвођача уређаја, телекомуникационих оператера и програмера. Андроид поседује велику заједницу програмера који стално проширују функционалности уређаја и развијају нове апликације. Већина апликација које се развију за Андроид налазе се на *Google Play*, која је раније носила име Андроид продавница.

Google Play је дигитална дистрибутивна платформа за Андроид апликације и Интернет продавница за мултимедијални садржај. *Google Play* је у власништву компаније *Google*. Овај сервис корисницима омогућава да лако пронађу и преузму Андроид апликације као и преузимање музике, књига, филмова итд. Сервис на произвођачима мултимедијалног садржаја оставља да ли ће се њихов садржај преузимати бесплатно или ће се наплаћивати. Такође, корисници могу извршити куповину уређаја као што су *Google Nexus* мобилни телефони и таблет уређаји.

Андроид је тренутно најраспрострањенији оперативни систем за паметне мобилне уређаје. До септембра 2013. године активирано је укупно милијарду Андроид уређаја. Дневно се са *Google Play* сервиса преузме 1,5 милијарди апликација а укупан удео Андроид оперативног система на светском трижишту паметних телефона износи око 64%.

Развијање софтвера за Андроид је процес у коме се програмирају нове апликације за Андроид оперативни систем. Апликације се углавном развијају у програмским језицима који су базирани на Јава програмском језику, користећи Андроид алате за програмере.

2.2 Јава

Јава је програмски језик који је конкурентан, базиран на класама и објектно оријентисан. Јаву је развила компанија *Sun Microsystems*. Главни концепти Јаве засновани су на језику Обреон чији је творац Никлаус Емил Вирт (нем. *Niklaus Emil Wirth*) који је, такође, написао Паскал, Модуле итд. Из Јаве је избачен концепт модула који је замењен пакетима какве и данас користимо. Идеја Јаве је да програмерима омогући да код који ради на једној платформи не мора да се прекомпајлира како би радио на другој платформи. Јава апликације се компајлирају у бајткод (*bytecode*) који може да се покрене на било којој Јава виртуелној машини без обзира на архитектуру рачунара. Јава је најпопуларнији програмски језик који се користи, процењује се да тренутно постоји 9 милиона Јава програмера. Јаву је осмислио Џејмс Гослинг (енг. *James Gosling*) и званично је објавио 1995. Овај програмски језик је доста своје синтаксе наследио од програмских језика C и C++, али за разлику од њих није толико оријентисан ка програмирању на ниском нивоу.

2.1.1 Аутоматско управљање меморијом

Јава користи аутоматски сакупљач смећа (*automatic garbage collector*) за уклањање сувишних објеката из меморије. Овај процес ради независно од покренутих програма и самостално одлучује које објекте ће отколони из меморије. Програмер одређује када се објекат креира, а *Java runtime* је задужен да ослободи меморију када објекат престане да се користи. Када више не постоје референце ка датом објекту, приступ недоступној меморији је одобрен и аутоматски је ослобађа *garbage collector*. Идеја аутоматског руковођења меморијом је та да програмер треба да буде поштеђен одговорности да ручно руководи коришћењем меморије.

2.2.1 Синтакса

Иако је синтакса Јаве у највећој мери наслеђена из програмског језика C++, на супрот програмском језику C++ који комбинује синтаксу за структурно, генеричко и објектно оријентисано програмирање, Јава је намењена само објектно оријентисаном програмирању. Сав код се пише унутар класа и сваки податак је објекат, изузев примитивних типова података као што су *integer*, *floating-point*, *boolean*, и карактери који нису објекти због бољег резултата тј. учинка написаног кода. Јава користи коментаре сличне коментарима програмском језику C++. Постоје три типа коментара: коментар једне линије означен са две косе црте (*//*), коментар који обухвата више линија одједном, он се састоји од почетног пара знакова који се састоји од косе линије праћеном звездицом (*/**) и завршава се обрнутим паром знакова тј. звездицом праћеном косом линијом (**/*). Трећи је *Javadoc* тип коментара који почиње са косом цртом и две звезде (*/***) а завршава се звездом и косом цртом (**/*). Овакав начин коментарисања омогућава кориснику да покрене *Javadoc* и формира документацију програма. Најчешће коришћене речи у Јава синтакси за дефинисање типа објеката су *public*, *static* и *void*.

Public указује да је метод јаван и да може бити позван из кода друге класе или класу могу користити друге класе изван хијерархије класе. Хијерархија класа односи се на име директоријума у коме се налази *.java* фајл.

Static означава да је метод статичан тј. да је повезан само са класом у којој је дефинисан, не са било којом инстанцом те класе. Једино статичке методе се могу позивати без референцирања објекта. Статичке методе не могу да приступе ниједном члану класе који није статичан.

Void показује да метод са овим префиксом не враћа никакву вредност.

Примери:

```
// Ово је пример коментарисања једне линије коришћењем две косе црте

/* Ово је пример коментарисања више линија одједном коришћењем комбинације коса црта и звезда на почетку и звезда коса црта на крају . Овакавим типом коментара лакше можемо да напишемо већи текст а такође можемо и лако да деактивирамо део кода */

package fibsandlies;
import java.util.HashMap;

/**
 * Ово је пример Javadoc коментара; Javadoc може да компајлира документацију из овог текста
 * . Javadoc коментар мора да буде праћен класом или методом..
 */
public class FibCalculator extends Fibonacci implements Calculator {
    private static Map<Integer, Integer> memoized = new HashMap<Integer, Integer>();

    /**
     * Main методу написану овако Јава виртуелна машина користи као почетну тачк програма.
     */
    public static void main(String[] args) {
        memoized.put(1, 1);
        memoized.put(2, 1);
        System.out.println(fibonacci(12)); //Get the 12th Fibonacci number and print to console
    }

    public static int fibonacci(int fibIndex) {
        if (memoized.containsKey(fibIndex)) {
            return memoized.get(fibIndex);
        } else {
            int answer = fibonacci(fibIndex - 1) + fibonacci(fibIndex - 2);
            memoized.put(fibIndex, answer);
            return answer;
        }
    }
}
```

2.3 Google Cloud Messaging

Омогућавање практичног имплементирања овог пројекта се налази у комуникацији и омогућавању корисницима да размењују кључеве. Додавање Интернет комуникације између два уређаја (корисника), или корисника и сервера, преко *Google Cloud Messaging* сервиса, који омогућава размену датотека у реалном времену. Преко овог сервиса корисници могу преко Интернета, у реалном времену да усклађују локалне низове и на тај начин обезбеде идентичан кључ на обе стране. Детаљније о овој технологији причаћемо у четвртом делу рада који се односи на комуникацију.

3 ПРЕГЛЕД ОБЛАСТИ ИСТРАЖИВАЊА

Протоколи од значаја који су утицали на стварање идеје и имплементацију овог система су:

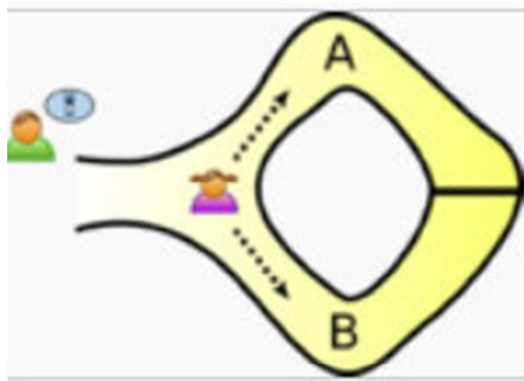
- „Сателитски Сценарио“
- Системи са нултим знањем

3.1 Сателитски Сценарио

Уели Маурер је увидео потенцијал „ББ84“ протокола и теоријски осмислио нови протокол познат под именом „Сателитски Сценарио“. Он је једносмерни квантни канал протокола „ББ84“ заменио сателитом који константно емитује случајне сигнале слабом снагом, тако да разни фактори могу да узрокују промене шума у комуникационом каналу, сваки емитовани сигнал садржи грешку на различитим местима у односу на првобитно емотиовани сигнал. Предности овог протокола су могућност размене симетричног кључа без унапред одређених параметара али и могућност генерисања кључева великих дужина. Недостатак овог протокола је његова имплементација и ако сателит делује као иделано решење овог протокола, имати приступ сателиту делује као нешто немогуће, али ако би се успешно имплементирао, решио би се проблем дистрибуције кључева код *OneTime pad* протокола.

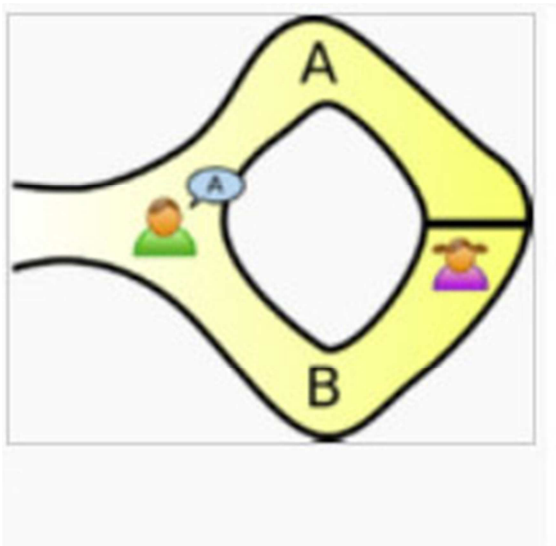
3.2 Системи са нултим знањем

Овај систем се користи када особа (Алиса) жели да докаже другој особи (Боб) да поседује нешто или зна неку тајну, а притом не жели да ода исту. Да би боље разумели овај проблем, помоћи ће нам следећи сценарио који називамо „Бобова пећина“.



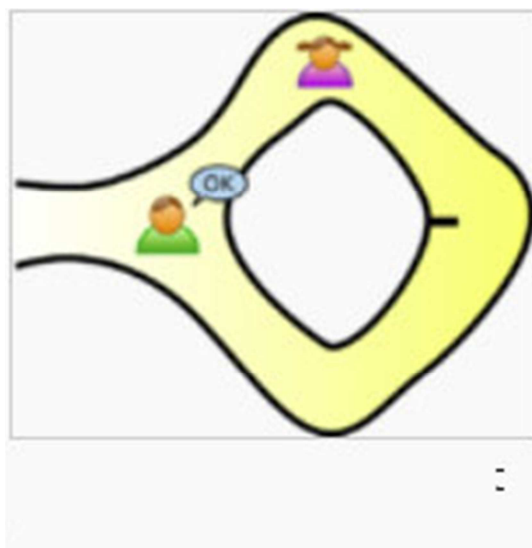
Слика 11: Прва фаза система са нултим знањем

На првој слици Алиса улази у Бобову пећину, баца новчић и на случајан начин бира да ли да крене путем А или В, док је Боб чека испред улаза у пећину.



Слика 12: Друга фаза система са нултим знањем

На другој слици се види да је Алиса изабрала пут В. Након тога Боб улази у пећину, баца новчић и на случајан начин бира да Алиса изађе путем А. Изабрана страна представља доказ о поседовању неке информације.



Слика 13: Трећа фаза система са нултим знањем

По сценарију на слици број три, Алиса и ако је изабрала пут В излази на пут А, јер поседује кључ којим може да пролази путем А и путем В. Ако Алиса не зна тајну може да превари Боба са вероватноћом $\frac{1}{2}$. Ако овај процес понове итеративно n пута, Алиса може да превари Боба само ако сваки пут погоди тачан одговор.

СОФТВЕРСКО РЕШЕЊЕ

3.3 Циљ пројекта

На основу претходних примера дошли смо на идеју да осмислимо протокол који би по угледу на „Сателитски Сценарио“ омогућио да из генератора насумичних вредности, који се налази на интернету, након неколико фаза усаглашавања својих локалних копија оригиналног низа, два корисника (Алиса и Боб) добију идентични симетрични кључ без слања кључа кроз комуникациони канал и без познавања локалне копије кључа другог корисника. За формирање локалних копија инспирацију смо нашли у игри „Потапање бродова“, коју играју два играча са по две координатне мреже (матрице) чије су димензије унапред познате. На својој локалној матрици играч распоређује бродове и бележи поготке противника, док на другој матрици бележи своје поготке. Аналогно овој игри и наши корисници праве насумичну локалу копију заједничке матрице са сервера, након чега насумично „гађају“ један другог како би установили на којим пољима се поклапају и тако добили заједнички кључ.

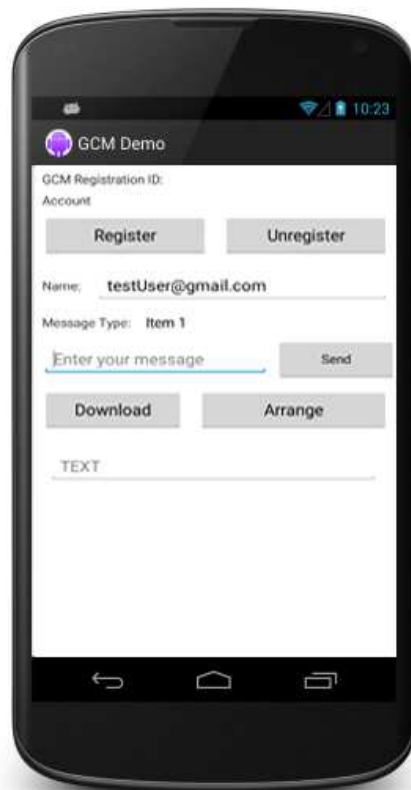
3.4 Архитектура целокупног софтверског решења

Архитектура овог софтверског решења се састоји из серверске и клијентске стране. На серверу је постављена дводимензионална матрица коју клијент преузима са сервера. Ова матрица је насумичним редоследом попуњена вредностима које могу да износе „0“ или „1“. Након преузимања, клијент од заједничке матрице прави локалну копију. Након тога, по принципу „Сателитског Сценарија“, врши се исправљање грешака у корелисаном низу. Прва фаза је фаза дестилације, друга фаза је усаглашавање информација. Након тога, формира се нови низ у који се уписује који парови локалне матрице сада транспоноване у једнодимензионални низ се задражавају а који одбацују, тај низ шаљу оба корисника и на основу примљеног низа кључеви се усклађују и оба клијента формирају идентични кључ. О овом делу протокола причаћемо детаљније мало касније.

3.5 Имплементација

За имплементацију овог протокола у апликацију коришћен је *Android Studio*. Ово је интегрисано развојно окружење (*IDE*) за Андроид платформу. Главна логика апликације смештена је у класи *MainActivity*. Притиском на дугме *Download*, чије је право име идентитета *downloadBT*, позивамо функцију *DownloadFile* која покреће прегледач (*browserIntent*) коме се прослеђује адреса датотеке која је смештена на серверу и тако покреће преузимање и складиштење датотеке на меморију телефона. Затим притиском на

дугме *Arrange*, чије је право име идентитета *arrangeBT* позивамо следеће функције редом: *readBytes*, *getKeys*, *Coding*, *informationReconciliation*, *privacyAmplification*.



Слика 14: Графички изглед апликације

3.1.1 readBytes

```
public void readBytes() {  
    System.out.println("readKeys");  
  
    final File file = new  
File(Environment.getExternalStorageDirectory().getAbsolutePath(),  
"Download/testByteKeyEX.txt");  
  
    try {  
        FileInputStream fstream = new FileInputStream(file);  
        DataInputStream in = new DataInputStream(fstream);  
        BufferedReader br = new BufferedReader(new InputStreamReader(in));
```

```

String strLine = "";
int i=0;
while ((strLine = br.readLine()) != null) {
    //save first 10 chars to name
    //System.out.println(strLine.length());
    //if (strLine.length()==0) break;
    for (int j = 0; j<50; j++) {
        zajednicka[i][j]=strLine.charAt(j);
        //System.out.print(name[i][j]);
    }
    //print the current data in name
    //System.out.println();
    // for(int k=0; k<50; System.out.println(name[i][k++]))
    //System.out.println(name);
    i++;
}
in.close();
} catch (Exception e) {
    System.err.println("Error: " + e.getMessage());
}

for (int i = 0; i<50; i++) {
    for (int j = 0; j < 50; j++) {
        System.out.print(zajednicka[i][j]);
    }
    System.out.println();
}

```

У овој функцији врши се читање матрице са сервера и уписивање у дводимензионални низ zajednicka[i][j]

3.2.1 *getKeys*

```
public void getKeys() {  
    Random rn = new Random();  
  
    int local_x;  
    int local_y;  
  
    for (int i = 0; i < 2500; i++) {  
        local_x = rn.nextInt(50);  
        local_y = rn.nextInt(50);  
  
        local[local_x][local_y] =  
Character.getNumericValue(zajednicka[local_x][local_y]);  
    }  
    System.out.println("getKeys");  
    for (int i = 0; i < 50; i++) {  
        for (int j = 0; j < 50; j++) {  
            System.out.print(local[i][j]);  
        }  
        System.out.println();  
    }  
}
```

У овој функцији се прво формира инцијална променљива (*seed*) m која се уводи ради веће вероватноће случајних вредности координата које се насумично бирају. Оне могу да садрже вредност од 0 до 50. Након што је изабран насумични пар координата, на тој позицији се у локалној матрици преписује вредност заједничке матрице која може да садржи „0“ или „1“. Тако се насумично попуњава половина локалне матрице, док се у преосталу половину уписује вредност „0“. Овим путем се обезбеђује да удео заједничког знања буде $\frac{1}{2}$ тј. да количина информације не прелази 50%.

3.3.1 Coding

```
public void Coding() {  
    System.out.println("Coding");  
    String y = "";  
    for(int i = 0; i<50; i++) {  
        for(int j=0; j<49; j+=2) {  
            y = "" + zajednicka[i][j] + zajednicka[i][j+1];  
            //i=i+1;  
            if (y.equals("10")) {  
                kodovano[i][j] = 1;  
            }  
            else /*if(y.equals("01"))*/ {  
                kodovano[i][j] = 0;  
            }  
        }  
    }  
  
    for (int i = 0; i<50; i++) {  
        for (int j = 0; j < 50; j++) {  
            System.out.print(kodovano[i][j]);  
        }  
        System.out.println();  
    }  
}
```

Овом функцијом се пролази кроз низ *for* петљом и одеређује парност пакета који се састоје од два узастопна бита. Ако је парност изабраног пара „1“ онда се задржава први бит, и том методом се уређује комплетан низ.

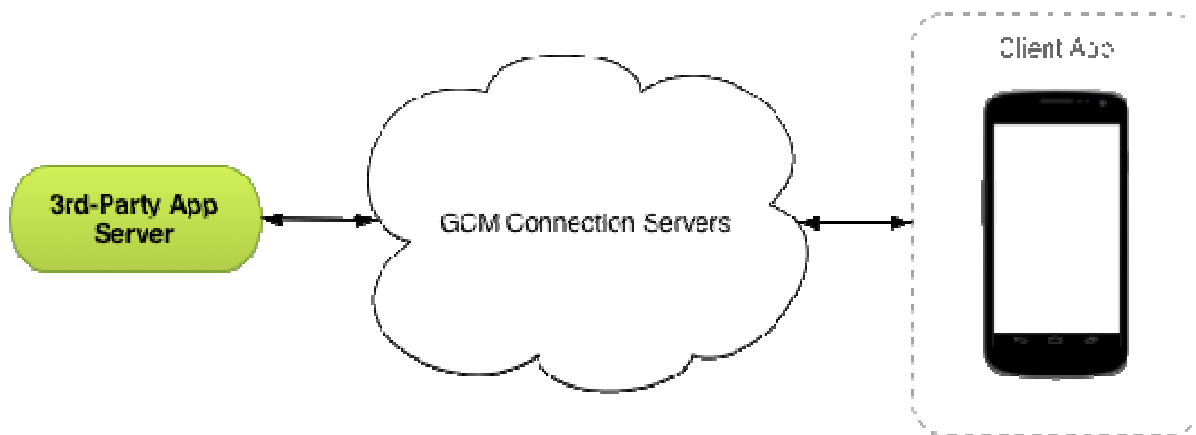
3.4.1 *informationReconciliation*

Помоћу ове функције претходно уређени низови се потуно изједначавају. Овај део протокола почиње тако што се претходно уређени низови поделе у блокове унапред одређене дужине за које се касније израчунава бит парности. Ако се на одређеном блоку бит парности разликује, то значи да постоји разлика између низова у том блоку. Ова функција се извршава у више рунди тј. док се не исправе све грешке. Низови који су преостали након извршавања ове функције представљају материјал за генерисање заједничког кључа на обе стране.

3.5.1 *privacyAmplification*

Фаза протокола о којој сада причамо се са разлогом последња извршава. Ова функција служи за потпуно избацивање нападачеве узајамне информације са низовима корисника који желе да формирају заједнички кључ. Ово постижемо тако што једна страна бира комбинацију функција које ће се применити и након тога шаље комбинацију бројева другом кориснику. Вредност једног броја представља једну криптографску функцију, такође, вредност ових бројева је унапред договорен.

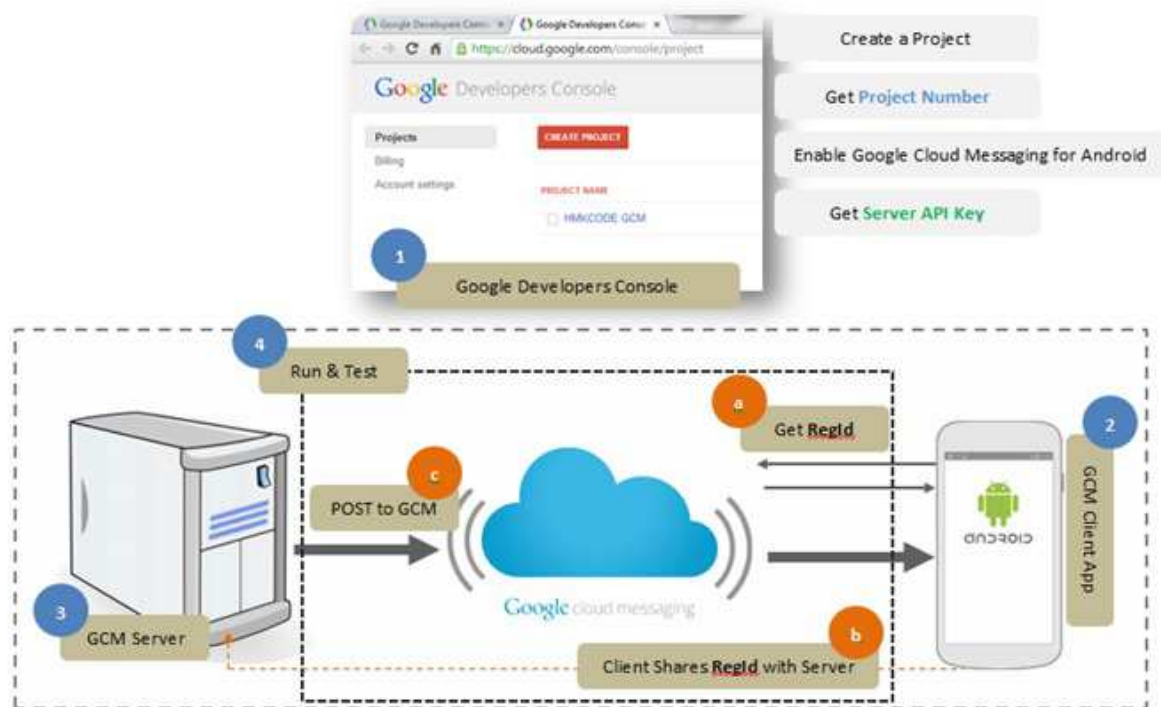
4 КОМУНИКАЦИЈА



Слика 15: Архитектура Google Cloud Messaging система

4.1 *Google Cloud Messaging*

Google Cloud Messaging за Андроид је бесплатан сервис који омогућава програмерима да шаљу податке са сервера до Андроид апликација на Андроид уређајима и да шаљу податке са апликације до сервера. Преко овог сервиса могуће је обострано послати обавештење о промени неког стања или послати податак величине до 4 kb. Овај сервис је задужен и за чување порука и њихово касније прослеђивање Андроид апликацији покренутој на одређеном уређају.



аутентификацији, чланови развојног тима, *email* адресе и сличне категорије које се користе у пројекту. У конзоли можемо да креирамо пројекат или да будемо позвани да надгледамо или уређујемо већ постојећи пројекат започет од стране програмера.

На слици испод видимо како изгледа почетни екран конзоле. Кликом на дугме „*Create project...*“ правимо нови пројекат који касније можемо да подешавамо по потреби наше апликације.



Слика 17: Креирање Google Cloud Messaging пројекта

2. Омогућавање GCM сервиса за Андроид

У овом кораку омогућујемо *Google Cloud Messaging* сервис за Андроид платформу. Без овог корака не би било могуће да имплементирамо овај сервис у нашу апликацију.

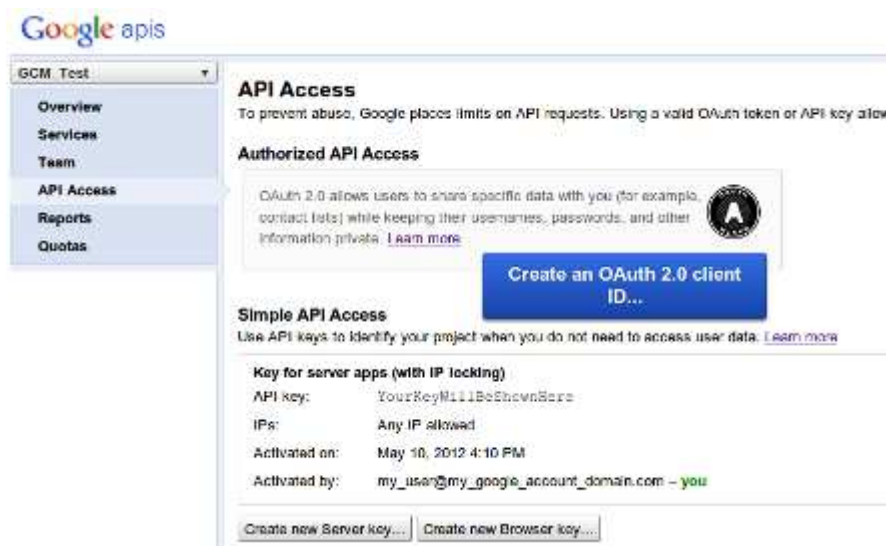
 Google Cloud Datastore API 	☐ OFF	Courtesy limit: 10,000,000 requests/day
 Google Cloud Messaging for Android 	☒ ON	
 Google Cloud Messaging for Chrome 	☐ OFF	Courtesy limit: 10,000 requests/day

Слика 18: Омогућавање Google Cloud Messaging-a за Андроид

3. Креирање серверског кључа

Сада морамо да креирамо серверски кључ како би могли да шаљемо податке са нашег сервера. На наредној слици видимо како изгледа форма за формирање кључа. Кликом на

думе „Create new Server key...” формирамо нови кључ. У напредним подешавањима постоји могућност да ручно подешавамо ИП адресу сервера.



Слика 19: Креирање серверског кључа

4.4 Smack CCS (XMPP) Server

Google Cloud Messaging (GCM) Cloud Connection Server (CCS) је крајња тачка XMPP(Extensible Messaging and Presence Protocol) која омогућава константну, асинхрону, двосмерну везу са Google сервером. Иста конекција се користи за слање и примање порука између севера и Андроид уређаја. Неке од прености коришћења овог протокола су:

- Асинхрони приступ XMPP протокола омогуће да се са мање утрошених ресурса шаље више података.
- Комуникација је двосмерна што значи да север може да шаље поруке Андроид уређају, али исто тако и Андроид уређај може да шаље поруке назад серверу.
- Уређај шаље поруке истом везом преко које је и примио поруку што увелико смањује утрошак батерије при слању порука.

У овом примеру видимо да податке које смо претходно дефинисали чувамо у две променљиве које су типа *String*. У променљивој *GCM_SENDER_ID* чувамо *ID* пројекта, док у променљивој *GCM_SERVER_KEY* чувамо кључ сервера. Након што смо додали *GCM_SENDER_ID* и *GCM_SERVER_KEY* можемо кренути да користимо локални сервер. Сада смо омогућили комуникацију са *Google Cloud Messaging* серверима ради слања *downstream* порука Андроид уређајима и примања *upstream* порука послате са уређаја.

4.2.1 Пример функција локалног сервера

```
public void handleIncomingDataMessage(Map<String, Object> jsonObject)
{
    @SuppressWarnings("unchecked")
    Map<String, String> payload = (Map<String, String>) jsonObject.get("data");

    //APA91bEVWVY Alice telefon
    //APA91bFb75o Bob telefon

    String from = jsonObject.get("from").toString();

    String to;
    if(from.equals("APA91bEVWVYmCeq2aALX2BbtxsVxt3eHBeBHBSpTU9r9yYbDUCzcCMfPhiDxccHSpYh8dR
3BRjjSHeJ6OsXcOgyc6m3I2HCFbJ0oYVtiMjO4_gcE4UXu9x7wNNpH4ZzH2ZfTDGtksGBiEMmWc5V5vDlM5i56
1WjJySUjqhnaqqYQO7ulJK4ml60"))
    {
        to
        ="APA91bFb75oRbSowe4mPa0_lq5bFziYTkbcd6U_NZXnWPwaXEkSdSxMIQQCNmgkfHVjBoLkMjPBXq9w_n8Og
NhKUEF10x0aKCP-A0xxCRWbOqQnx-
rsVeGkuEb4xtfk4pa_uaaWcsFh0ttL0U9f1ZtqrqbweJWN8kCF9YH5t78fUgI304HIPvOw";
    }
    else
    {
        to
        ="APA91bEVWVYmCeq2aALX2BbtxsVxt3eHBeBHBSpTU9r9yYbDUCzcCMfPhiDxccHSpYh8dR3BRjjSHeJ6OsXc
Ogyc6m3I2HCFbJ0oYVtiMjO4_gcE4UXu9x7wNNpH4ZzH2ZfTDGtksGBiEMmWc5V5vDlM5i561WjJySUjqhnaqq
YQO7ulJK4ml60";
    }

    // PackageName of the application that sent this message.
    String category = jsonObject.get("category").toString();
    logger.log(Level.INFO, "Application: " + category);

    String action = payload.get("action");
    if(action.equalsIgnoreCase("com. cryptomodul.gcmdemo.REGISTER"))
    {
        String name = payload.get("name").toString();
    }
}
```

```

// Store username and registrationID in DB
    addUser(name, from);

    // Send an REGISTER response back
    payload.put("message", "Registration successful");

    String echo = createJsonMessage(from, getRandomMessageId(), payload, null,
null, false);

    send(echo);

    logger.info("Adding new user: " + name + ":" + from);
}
else if(action.equalsIgnoreCase("com. cryptomodul.gcmdemo.UNREGISTER"))
{
    removeUser(from);

    logger.info("Removing ID: " + from);
}
else if(action.equalsIgnoreCase("com. cryptomodul.gcmdemo.ECHO"))
{
    // Send an ECHO response back

    //zamenio from sa to

    String echo = createJsonMessage(to, getRandomMessageId(), payload, null,
null, false);

    send(echo);
}
else if(action.equalsIgnoreCase("com. cryptomodul.gcmdemo.BROADCAST"))
{
    // Send an Broadcast response back

    broadcastMesage(payload);
}
else if(action.equalsIgnoreCase("com. cryptomodul.gcmdemo.NOTIFICATION"))
{
    // Send a Notification response back

    try
    {
        User user = userDao.queryForId(from);

        NotificationKey notificationKey =
notificationDao.queryForId(user.getName());

        notificationMessage(notificationKey, payload);
    }
    catch (Exception e)
    {
        logger.error("Error in Notification response back", e);
    }
}
}

```

```

    }

    catch (SQLException e)
    {
        e.printStackTrace();
    }

}

else
{
    logger.warning("Unkown action sent: " + action);
}

}

```

У овој функцији се одређује шта се дешава са поруком коју је сервер примио. На почетку се проверава из *String*-а под називом „*from*“ који је заправо *JSON* објекат, који проверава од кога је порука доспела до сервера, и да ли је то корисник који је регистровани корисник наше апликације. Сервер, такође, проверава да ли је корисник регистрован, ако јесте порука се прослеђује, ако корисник није регистрован од њега се захтева да унесе податке потребне за регистрацију и на крају ако корисник поднесе захтев да се његов налог избрише, сервер уклања његове податке. Након овога, проверава се тип слања поруке који је корисник изабрао. Детаљније о самој стуктури поруке причаћемо у следећем поглављу.

4.3.1 Слање порука

Архитектура поруке

GCM порука је у суштини *JSON* објекат који садржи неке од опција које су обавезне или могу бити препуштене слободном избору. Ове опције омогућавају различите облике понашања при испоруци поруке.

- *to*: APA91bEVWVY
- *message_id*: *јединствени ID поруке*
- *data*: јединствени *JSON* објекат у коме се налазе подаци које садржи порука
- *time_to_live*: (необавезна опција) време трајања поруке, подразумевана вредност је две недеље
- *delay_while_idle*: (необавезна опција) *boolean* чија вредност одређује да ли да се порука достави уређају који није активан

Изглед *GCM* поруке

```

{
  "to": "DEVICE_REGISTRATION_ID",
  "message_id": "m-1366082849205",
  "data":

```

```
{  
  "message":"hello world",  
  "action":"com.cryptomodul.gcmdemo.ECHO",  
}  
"time_to_live":"600",  
"delay_while_idle": false  
}
```

Корисник може изабрати три начина на која ће послати поруку а то су: *Broadcast*, *Echo*, *User Notification*.



Слика 21: Приказ могућих типова порука

Echo Message

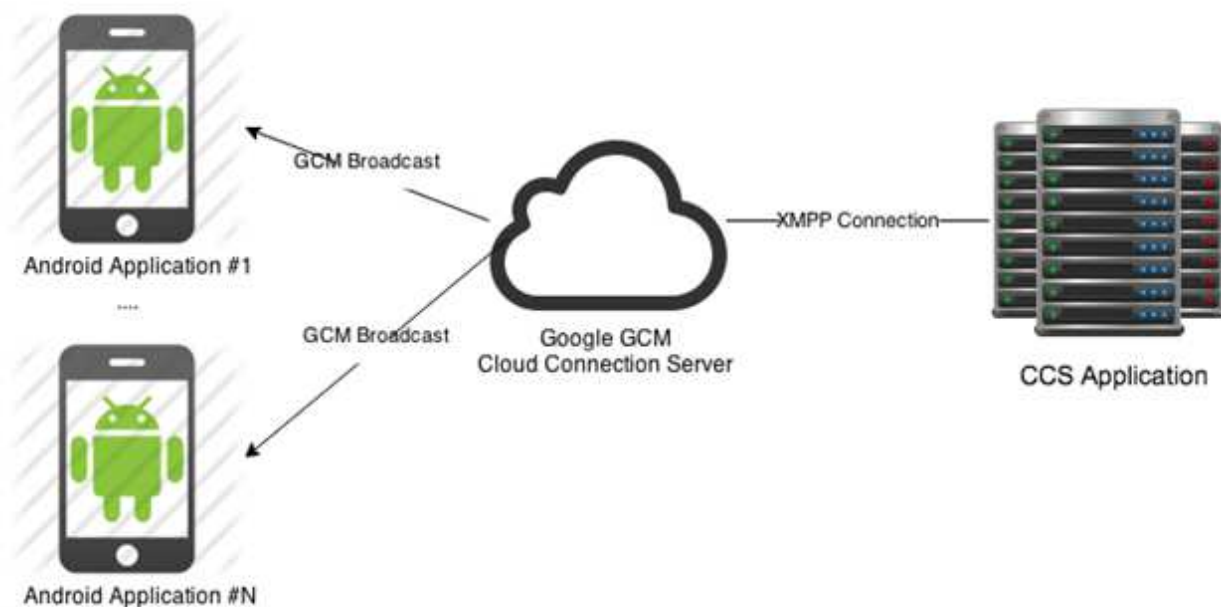
Овај тип порука функционише по принципу еха у пећини. Уређај шаље *upstream* поруку *GCM* сервису а локални сервер исту поруку враћа до уређаја *downstream* конекцијом. У овом случају порука се чува на локалној меморији уређаја док не стигне успешно, што повећава сигурност при губљењу везе са Интернетом.



Слика 22: Графички приказ поруке типа *Echo*

Broadcast Message

Наша апликација може да пошаље и поруку типа *Broadcast*. Овај тип поруке сличан је претходном типу али на овај начин порука се шаље свим корисницима укључујући и пошиљаоца поруке. Овакав тип поруке користан је и може се користити у разне сврхе, на пример на овај начин лако можемо обавестити кориснике да је могуће ажурирати верзију апликације.

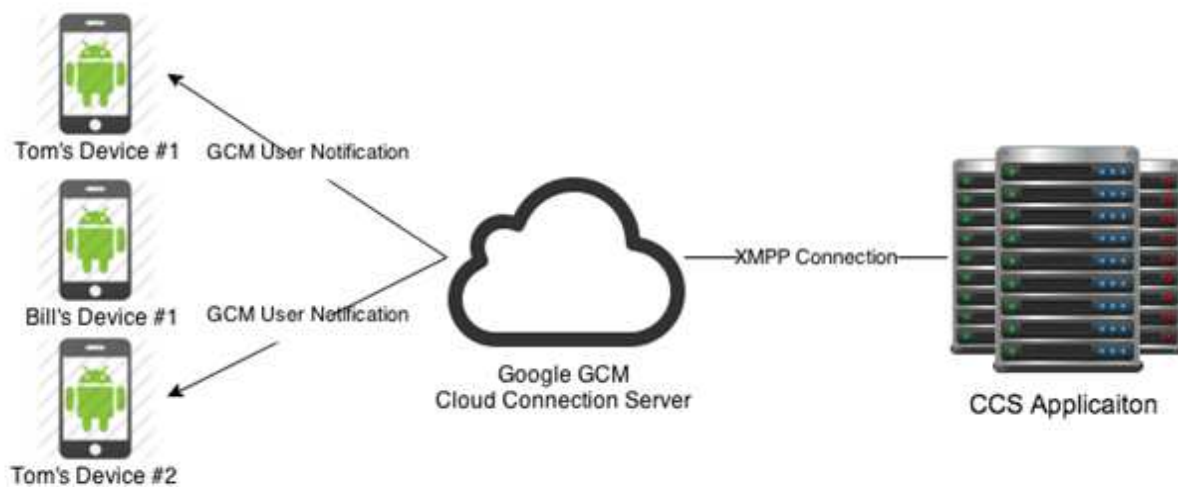


Слика 23: Графички приказ поруке типа *Broadcast*

User Notification Message

Последњи, али свакако не и најмање важан тип порука је *User Notification Message*. Овај тип порука дозвољава да се уређаји корисника групишу тако да једна порука буде прослеђена свим уређајима корисника. За овај тип слања потребно је при регистрацији корисника направити кључ помоћу којег би се сви уређаји груписали. Овакав тип слања

порука користан је у следећем сценарију. Ако из неког разлога Алиса у неком делу протокола размене кључева мора да користи други уређај, овим приступом би било могуће наставити протокол са тог уређаја.



Слика 24: Графички приказ поруке тина User Notification

ПЛАН ЗА ДАЉИ РАЗВОЈ

One-time-pad

Као побољшање крипто модула планира се спајање новонастале апликације са већ постојећим пројектом „шифровање смс порука *OneTime pad*-ом на Андроид платформи. Спајањем ове две апликације проширили би могућности наше апликације и на смс комуникацију. Још једно од планираног побољшања нашег модула је то да наше дводимензоналне матрице проширимо са још једном или више димензија, заједнички кључ би тада био вишедимензионални облик до кога би нападач још теже дошао.

ЗАКЉУЧАК

У овом раду смо прошли кроз нека од питања везаних за размену кључева симетричне и асиметричне криптографије и мало боље се упознали са основним начелима размене кључева. Развојем крипто модула за размену порука на Андроид платформи са планираним побољшањима добијамо комплетно, самостално решење размене шифрованих података које има примену у скоро свим сферама нашег друштва.

У веома кратком временском периоду наш модул може генерисати кључеве великих дужина. У овом раду размену кључева смо применили у апликацији за размену порука али овај модул се може применити при размени било ког типа информације на разним системима и платформама. Због велике дужине кључа могу се применити разни криптолошки алгоритми који би одговарали потребама корисника. Са успешно размењеним кључевима и шифрованим садржајем наше поруке, може се рећи да је наш модул за размену порука потпуно сигуран.

Међутим, флексибилност и отвореност Андроид система оставља велику могућност да сами уређаји или код написан од стране програмера садржи елементе који би нарушили безбедност нашег система. Претпоставља се да је компанија *Samsung*, која је једна од водећих произвођача Андроид паметних уређаја, на неким моделима паметних уређаја остављала такозвани бекдор (енг. *backdoor*) преко којег је могуће активрати микрофон, камеру, имати увид у геолокацију или приступити фајловима на уређају. У овом случају трећа страна би лако могла доћи до израчунатог кључа или имати увид у карактере поруке које смо уносили додиривањем екрана нашег уређаја.

Из свега горе наведеног можемо закључити да само ако смо у потпуности самостално развили и хардверски и софтверски део нашег модула можемо бити сигурни да је наш модул апсолутно безбедан.

ЛИТЕРАТУРА

- [1] *Милан Милосављевић, Саша Адамовић, Криптологија 2*, прво издање Универзитет Сингидунум, Београд 2014
- [2] *Милан Милосављевић, Саша Адамовић, Криптологија 1*, прво издање Универзитет Сингидунум, Београд 2013
- [3] *Милан Милосављевић, Саша Адамовић, Основи теорије информација и кодовања*, прво издање Универзитет Сингидунум, Београд 2013
- [4] *Thawte, History of cryptography*, 2013
- [5] Званична интернет документација *Google Cloud Messaging* сервиса
<https://developer.android.com/google/gcm/index.html>
- [6] Званична интернет документација *Smack* библиотеке
<https://community.igniterealtime.org/welcome>