VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2017 Bc. Jiří Jonáš

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

MOBILNÍ APLIKACE PRO ŠIFROVANÉ VOLÁNÍ
MOBILE APPLICATION FOR ENCRYPTED CALLS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Jiří Jonáš**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Jan Hajný, Ph.D.**
SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**
Ústav telekomunikací

Student: Bc. Jiří Jonáš ***ID:*** 120073
Ročník: 2 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Mobilní aplikace pro šifrované volání

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je implementace aplikace pro bezpečnou komunikaci na platformě Android. Výstupem práce bude měření rychlosti kryptografických primitiv na telefonu, implementace aplikace pro přenos šifrovaných hlasových dat s šifrováním volitelně v softwaru či na microSD kartě, přičemž implementace bude podporovat obě varianty.

DOPORUČENÁ LITERATURA:

[1] Android Developer [online]. 2016 [cit. 2016-09-12]. Dostupné z: https://developer.android.com/

[2] MENEZES, Alfred, Paul C. VAN OORSCHOT a Scott A. VANSTONE. Handbook of applied cryptography. Boca Raton: CRC Press, c1997. Discrete mathematics and its applications. ISBN 0-8493-8523-7.

Termín zadání: 1.2.2017 ***Termín odevzdání:*** 24.5.2017

Vedoucí práce: doc. Ing. Jan Hajný, Ph.D.
Konzultant:

doc. Ing. Jiří Mišurec, CSc.
předseda oborové rady

Abstrakt
Práce se zabývá implementací aplikace bezpečného přenosu hlasového signálu v digitální podobě přes lokální datovou síť pro operační systém Android. Pro řízení hovoru je použit signalizační protokol SIP a pro přenos hlasových dat je implementován protokol RTP. Pro zajištění bezpečnosti hovoru je nejprve vygenerován oběma komunikujícími stranami symetrický šifrovací klíč pomocí Diffie-Hellmannova protokolu. Po ustanovení šifrovacího klíče, je mezi klienty navázáno telefonické spojení, které je šifrováno pomocí symetrické šifry AES. Šifrování mezi oběma stranami je prováděno přímo v aplikaci nebo na microSD. Součástí řešení práce je provedení měření rychlosti kryptografických primitiv, která jsou použita pro vytvoření šifrovaného hovoru.

Klíčová slova
Kryptografie, šifrování, VOIP, SIP, RTP, AES, Diffie Hellman, Kodeky, Android, Java, Java Card, Secure Element

Abstract
The thesis is focused on implementation of aplication for secure telephone communication on data network. Application is developed for operating system Android. For call management is responsible signaling protocol SIP and for transfer of voice data is used protocol RTP. For security of call is first created cryptografic key for symetric cryptography. After generating key is established call, which is encrypted by symetric cipher AES. Encrypting between communicating sides is provided in application or on microSD card. Part of solution is measurement of speed of cryptographic primitives, which are used for secure call.

Keywords
Cryptography, encryption, VOIP, SIP, RTP, AES, Diffie Hellman, Codecs, Android Java, Java Card, Secure Element

JONÁŠ, J. *Mobilní aplikace pro šifrované volání.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2017. 52 s. Vedoucí diplomové práce doc. Ing. Jan Hajný, Ph.D..

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma "Mobilní aplikace pro šifrované volání" jsem vpracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení §11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

PODĚKOVÁNÍ

Touto cestou bych velice rád poděkoval panu doc. Ing. Janu Hajnému, Ph.D, který připravil velice zajímavou práci a který mi poskytl velmi cenné rady, doporučení a zdroje k vypracování diplomové práce. Dále bych chtěl poděkovat panu Ing. Petru Dzurendovi, který mi ochotně poskytnul pomoc při práci s microSD kartou. V neposlední řadě bych rád poděkoval své rodině a přítelkyni za podporu a trpělivost.

Brno ....

podpis autora

SIX research centre
sensor, information and communication systems

Faculty of Electrical Engineering
and Communication

Brno University of Technology
Technicka 12, CZ-61600 Brno, Czechia

http://www.six.feec.vutbr.cz

Výzkum popsaný v této diplomové práci byl realizovaný v laboratořích podpořených projektem Centrum senzorických, informačních a komunikačních systémů (SIX); registrační číslo CZ.1.05/2.1.00/03.0072, operačního programu Výzkum a vývoj pro inovace.

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

2007–13
OP Výzkum a vývoj pro inovace

OBSAH

1 ÚVOD

V době, kdy se internet stává naším neodmyslitelným společníkem nejen doma u počítače, či ve školní laboratoři, ale je možné na něj "dosáhnout" i na odlehlých místech naší planety, jsou přesouvány téměř všechny služby na toto zázračné médium. Telefonování přes internet sice dnes není žádnou novinkou, ale během vývoje se stále odstraňují překážky, které od počátku brání jeho většímu prosazení. Jednak je to rychlost přenosu, která má obrovský vliv na kvalitu hovoru. Každý rok se tento parametr posunuje mílovými kroky kupředu a dnes je možné dokonce prostřednictvím mobilního telefonu přenášet data rychlostí, která spolehlivě stačí k přenosu hlasových dat.

Jedním z dalších problémů, které vyvstávají při přesunu volání do internetové sítě je jeho bezpečnost. Ne vždy je uživatelům k dispozici zabezpečené připojení k internetu a připojením k veřejné WI-FI síti se člověk může snadno stát obětí odposlechu.

Cílem této práce je připravit aplikaci pro operační systém Android, pomocí které by bylo možné uskutečnit hovor mezi dvěma uživateli připojenými do počítačové sítě a který by byl během přenosu počítačovou sítí šifrován.

Základním krokem je vytvoření aplikace, která dokáže spojit dva klienty v rámci jedné počítačové sítě bez šifrovaného přenosu. Účastníci hovoru nebudou komunikovat prostřednictvím serveru, ale budou komunikovat peer-to-peer. Další část, která vede k vytvoření aplikace se šifrovaným voláním, je vytvoření kryptografických primitiv, které umožní šifrování aplikace. Mezi kryptografická primitiva patří jednak distribuce nebo sestavení šifrovacího klíče pro symetrické šifrování a pak také použití tohoto klíče při šifrování samotných dat. Aby bylo možné provést výběr nejvhodnějších kryptografických algoritmů pro daný účel, je potřeba provést měření rychlosti těchto algoritmů. Pro větší bezpečnost šifrování přenášených dat bude přidána možnost šifrování přenášených data pomocí micro SD karty se secure elementem.

Spojením těchto jednotlivých kroků bude vytvořena aplikace pro šifrované volání, která je schopna sestavit šifrovací klíč přes potenciálně nebezpečný komunikační kanál a pomocí něj šifrovat přenos hlasových dat.

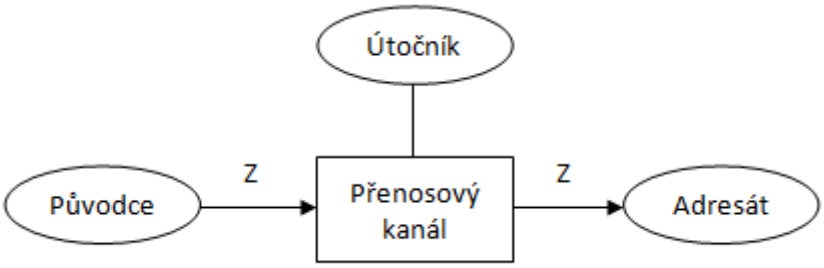
2 KRYPTOGRAFIE

Kryptografie je věda, zabývající se utajováním smyslu zprávy, která je přenášena přes veřejně přístupný komunikační kanál. Pro zjištění přenášené zprávy je zapotřebí speciální znalost, která umožní přenášenou zprávu rozluštit. Kryptografii je možné využít pro dva typy případů. Prvním z nich je zajištění důvěrnosti zprávy, kde je zajištěno, že obsah zprávy obdrží pouze oprávněné osoby. Druhým případem je zajištění autentičnosti, čemuž rozumíme jako zajištění autorství odesílané zprávy.

2.1 Základní pojmy

Mezi základní pojmy patří v kryptografii informace, nosič informace a zpráva, kde informací rozumíme abstraktní zobrazení skutečnosti. Informaci je možné přenášet v čase a prostoru prostřednictvím nosiče informace. Jako nosič informace je možné si představit materiál nebo proces. Zprávu můžeme následně definovat jako posloupnost stavů daného nosiče informace, do kterého byla informace zakódována. Pokud se jedná o posloupnost stavů nosiče v čase, nazýváme ji jako signál (např. časový průběh změn proudu) a posloupnost v prostoru se nazývá záznam (např. sled nabitých nebo vybitých kondenzátorů v dynamické paměti). Nosič informace může nabývat konečného nebo nekonečného počtu stavů, přičemž v současné době se v drtivé většině případů můžeme střetnout s konečným počtem stavů. Stav může být určen pomocí znaku z nějaké konečné abecedy nebo celým číslem. Pro kryptografii není podstatné, v jaké formě se šifrované zprávy nachází.

V kryptografií se informace vyskytuje pouze v podobě zpráv, kterým je zapotřebí zajistit potřebnou ochranu. Problém ochrany zpráv je ilustrován na obrázku 2.1, kde je zobrazeno schéma přenosu zpráv. Zdroj, který vysílá zprávy je označován jako původce, příjemce zprávy se nazývá adresát. Tyto subjekty jsou nazývány jako oprávněné osoby. Může se ovšem jednat i o stroje. Zprávy jsou přenášeny pomocí přenosového kanálu, což je technický systém, umožňující přenos zprávy prostorem nebo časem. K přenosovému kanálu má kromě původce a adresáta přístup i útočník. Jedná se o osobu, která není oprávněna k odposlechu či modifikaci zprávy. Takto může útočník získat důvěrné informace, případně může modifikací zprávy přinutit adresáta k takovému chování, které povede ke ztrátě aktiv. [2]



Obr. 2.1: Schéma přenosu zpráv

2.1.1 Cíle kryptografie

Kryptografie si klade za cíl zajištění důvěrnosti, integrity dat, autentičnosti a nepopiratelnosti.

Důvěrnost je služba, která slouží k udržení informací pouze mezi osobami, které jsou oprávněny jí disponovat. V kryptografii se pro zajištění důvěrnosti využívá matematických algoritmů, které činí přenášené údaje nečitelné.

Integrita dat je službou, která se zaměřuje na neoprávněné změny dat. Umožňuje detekovat, zda bylo s daty manipulováno. Tedy, že ze zprávy mohou být některé části odstraněny, nahrazeny nebo může být do zprávy nějaká informace přidána.

Autentičnost je služba, která se vztahuje k identifikaci jak jednotlivých subjektů, tak samotné informace. Obě komunikující strany by měly být navzájem ověřeny. Informace, která je dodána prostřednictvím přenosového kanálu, by měla být ověřena podle místa a data vzniku, obsahu dat, času odeslání, atd. Z těchto důvodů je autentičnost v kryptografii rozdělena do dvou hlavních tříd, a to autentizace entit a ověření původu dat.

Poslední službou je nepopiratelnost. Tato služba předchází odmítnutí autorství dané zprávy původcem, od kterého byla zpráva údajně zaslána. Původce zprávy tedy nemůže popřít, že danou zprávu vydal. [1]

2.2 Kryptosystémy

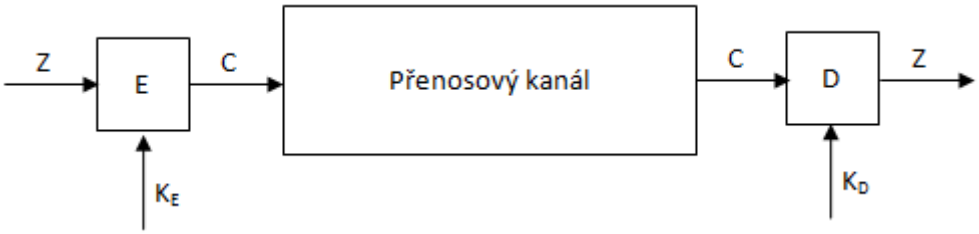
V kryptografii je pro ochranu přenášených informací využívána transformace zpráv. Proces spočívá v transformaci zprávy Z, která je představována nějakou posloupností, na jinou posloupnost C, tzv. kryptogram. Nutnou podmínkou pro vytvoření kryptogramu je jeho zpětná transformace z posloupnosti C zpět na posloupnost Z. Zpětnou transformaci je možné provést pomocí utajované informace K, tzv. klíče, kterou má k dispozici pouze adresát.

Předmětem kryptografie jsou kryptografické systémy, které slouží k zajištění důvěrnosti a autentičnosti zpráv. [2]

2.2.1 Kryptosystém zajišťující důvěrnosti přenášených zpráv

Kyptosystém zajišťující důvěrnost přenesených zpráv je označován jako utajovací kryptosystém nebo také jako šifra. Tento typ systému vyžaduje pro svoji funkci dva parametry. Na straně původce se jedná o číslo K_E a na straně adresáta je vyžadováno číslo K_D.

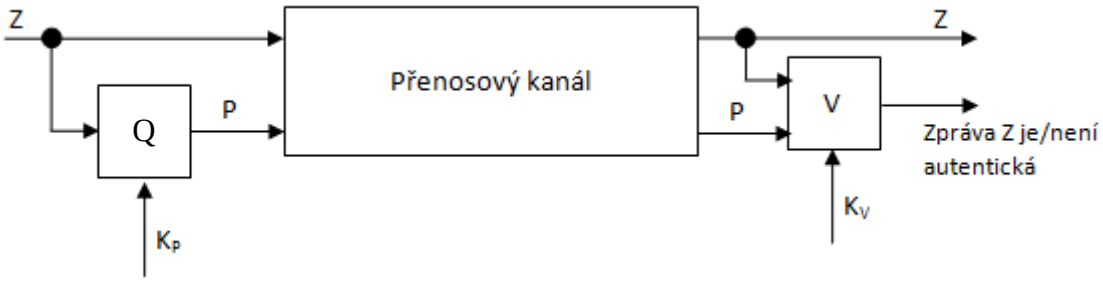
Kryptosystém využívá matematické funkce, které provádějí šifrování na straně původce a dešifrování na straně příjemce. Tyto funkce tvoří jádro celého systému. Na obr. 2.2 je systém pro důvěrný přenos zpráv zobrazen.



Obr. 2.1: Krytosystém pro zajištění důvěrnosti

2.2.2 Kryptosystém zajišťující autentičnosti přenášených zpráv

Autentičností zprávy rozumíme stav, kdy jsou informace o přenášené zprávě pravdivé. Jedná se o místo a datum vzniku zprávy, čas odeslání zprávy, identifikační údaje původce zprávy apod. Pravdivost přenášených zpráv je zajištěna pomocí dodatečných dat, která jsou přenášena společně se zprávou a umožní ověření pravosti informací, které jsou obsaženy ve zprávě.



Obr. 2.2: Kryptosystém pro zajištění autentičnosti

Schéma kryptosystému, na základě kterého jsou autentická data přenášena, je zobrazen na Obr. 2.3. Kryptosystém využívá pro zajištění autentičnosti, dvou parametrů K_P a K_V, které se nazývají pečetící a verifikační klíč. Princip kryptosystému je takový, že je pro odesílanou zprávu vypočítáno číslo, které danou zprávu reprezentuje. Z tohoto čísla je za pomocí pečetící funkce Q a pečetícího klíče K_P vytvořena tzv. pečeť zprávy P. Tato pečeť je přenášena spolu se zprávou adresátovi.

Na straně adresáta je ze získané zprávy vypočítáno číslo, reprezentující danou

zprávu, stejným způsobem, jako na straně původce. Číslo reprezentující zprávu je následně porovnáno s pečetí, která dorazila společně se zprávou. Pečeť je nejprve rozšifrována za pomocí ověřovací funkce V a ověřovacího klíče K_V a pokud se výsledek shoduje s číslem, který vypočítal adresát, jedná se o autentickou zprávu.

2.3 Symetrické šifry

Symetrické kryptosystémy, využívají k šifrování klíč K_E na straně původce a na straně adresáta klíč K_D k dešifrování přenesené zprávy. Klíče K_E a K_D jsou navzájem odvoditelné. Jedná se tedy o tajnou informaci, kterou musejí původce a adresát udržovat v tajnosti. Tyto klíče mohou být různé, ale v praxi se téměř vždy používají klíče stejné. Platí tedy, že $K_E = K_D = K$. Klíčem nemusíme vždy rozumět souvislý blok dat, ale může se rovněž jednat o funkci či algoritmus, který obě komunikující strany sdílejí.

Symetrické šifry je možné rozdělit do dvou tříd. První z nich je tzv. Proudová šifra, která šifruje data po jednotlivých bitech. Druhou třídu představuje Bloková šifra, která provádí šifrování po jednotlivých blocích o pevné délce n-bitů. Použití proudové šifry je rychlejší, ovšem je méně odolná vůči kryptoanlýze [1], či jiným útokům. Blokové šifry jsou robustnější, ovšem s tím souvisí také vyšší náročnost na implementaci i na výkon šifrovacího stroje. Ovšem s postupným vývojem techniky je rozdíl v efektivitě jednotlivých šifer téměř zanedbatelný. Dává se tedy většinou přednost odolnější blokové šifře.

Symetrické šifrování se využívá výhradně k utajování zpráv.

2.3.1 Proudové šifry

Proudové šifry kódují přenášenou zprávu po jednotlivých bitech. Každý i-tý bit ve výsledném kryptogramu tedy odpovídá i-tému bitu šifrované zprávy a klíči K:

$$c_i = E(z_i, K)$$

Pro šifrování se využívá šifrovací posloupnost S, tzv. keystream, který je možné buď generovat pomocí generátoru pseudonáhodné posloupnosti nebo je možné tento klíč číst z paměti. Implementace generátoru pseudonáhodné posloupnosti či data, která jsou uložena v paměti musejí být pro obě komunikující strany shodná. Posloupnost S musí být rovněž na obou stranách synchronizovaná. Jinak by nebylo možné zprávu na straně příjemce rozluštit. Šifrování a dešifrování je tedy prováděno pomocí následujících vztahů:

$$c_i = z_i \oplus s_i, \tag{2.1}$$

$$\mathrm{z_i} = \mathrm{c_i} \oplus \mathrm{s_i}, \tag{2.2}$$

kde symbol $\oplus$ představuje binární operaci XOR, c_i je i-tý bit kryptogramu, z_i je

i-tý bit zprávy a s_i představuje i-tý bit klíče.

Dokonalá šifra

V souvislosti s proudovou šifrou se vyskytuje pojem tzv. Dokonalá šifra. Pokud bychom chtěli dosáhnout takovéto šifry museli bychom splnit tři základní podmínky.

- Klíč musí být stejně dlouhý, jako šifrovaná zpráva. Pokud by klíč nedosahoval délky zprávy, bylo by možné šifru prolomit hrubou silou.
- Klíč musí být náhodný a každý kombinace klíče musí být stejně pravděpodobná.
- Klíč není možné používat opakovaně. Žádné dvě zprávy tedy nemohou být zašifrovány stejným klíčem.

Dokonalá šifra je tedy velmi náročná na správu klíčů. Šifrovací klíč musí být vygenerován nedeterministickým generátorem náhodných čísel a uložen do paměti, která je poté doručena jak původci zprávy tak i jejímu adresátovi. Klíč v tomto případě není možné řešit pomocí generátoru náhodných čísel na obou komunikujících stranách, protože není možné vytvořit dva generátory náhodných čísel, které by generovaly stejnou posloupnost. Pro zašifrování např. 1GB dat tedy potřebujeme klíč o stejné délce.

2.3.2 Blokové šifry

Blokové šifry kódují zprávy, které představuje řetězec bitů o pevné délce. Číslo, které reprezentuje počet bitů je nazýváno délka bloku. Pokud je zpráva, kterou původce potřebuje zašifrovat, delší než daný počet bitů, bude nutné zprávu rozdělit a zašifrovat ji v několika krocích. Příjemce jednotlivé zprávy rozšifruje a následně zřetězí, čímž získá původní zprávu. Pokud je zpráva kratší než délka bloku, dosáhne se délky bloku tzv. výplňovými bity, které mají předem definovanou hodnotu. Často se postupuje tak, že se za poslední bit zprávy doplní hodnota 1 a pokud je třeba, doplní se blok potřebnou sekvencí bitů o hodnotě 0. Adresát zprávy potom zkontroluje poslední bit zprávy. Pokud je jeho hodnota rovna 1, tak ji odstraní. Pokud je posledním bitem hodnota 0, postupuje se od konce rozšifrované zprávy k prvnímu výskytu bitu s hodnotou 1, který je se všemi následujícími bity s hodnotou 0 odstraněn.

2.3.3 Režimy blokové šifry

Pro blokovou šifru by bylo výhodné, pokud by byl každý blok zašifrován vlastním klíčem. Zašifrované zprávě by tak bylo možné zajistit maximální možnou míru utajení. Tento způsob by ale byl velmi náročný na zprávu klíčů a volí se tedy přístup, kdy jsou všechny bloky zašifrovány stejným šifrovacím klíčem. Tento přístup ovšem snižuje stupeň důvěrnosti šifry na minimální. Aby bylo možné lépe čelit útokům, které by měly za úkol narušit důvěrnost šifry, provozuje se bloková šifra podle jejího použití v několika režimech. Použitím správného režimu pro daný případ je možné zvýšit míru důvěrnosti dané šifry

Režim ECB

Nejjednodušší režimem blokové šifry se nazývá Electronic Code Book (ECB). Schéma šifrování a dešifrování v tomto režimu je zobrazeno na obrázku Obr.2.3. Každý blok Z_i šifrované zprávy Z je zakódován pomocí funkce E_K, která představuje šifrování pomocí klíče K, z které je získán zašifrovaný block C_i. Výsledný kryptogram se tedy skládá z kryptogramů $C_1...C_m$. Index *m* představuje počet bloků, na které byla šifrovaná zpráva rozdělena.

Režim ECB je velmi jednoduchý, ovšem jeho nevýhodou je skutečnost, že pro dva stejné bloky Z_i, Z_j získáme kryptogramy C_i, C_j, které budou rovněž stejné. Jeho použití je tedy v aplikacích, kde výskyt stejných bloků nehrozí. [2]

Obr. 2.3 Schéma šifrování v režimu ECB

Obr. 2.4 Schéma dešifrování v režimu ECB

Režim CBC

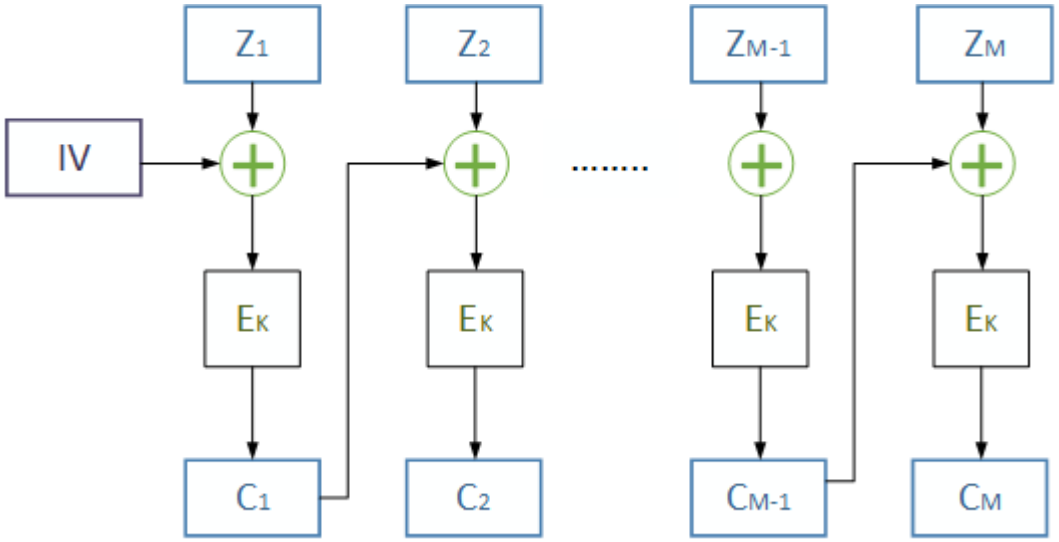
Druhý režim blokové šifry se nazývá Cipher Block Chaining (CBC). V tomto režimu je zajištěno, že pro dva stejné bloky Z_i, Z_j získáme rozdílné kryptogramy C_i, C_j.

Nevýhodou ovšem je, že tento algoritmus šíří chyby. Pokud se například v kryptogramu C_1 vyskytne chyba, která může vzniknout během přenosu, je nutné přenést celou zprávu znovu. Šifrování a dešifrování v tomto režimu probíhá podle následujících vzorců:

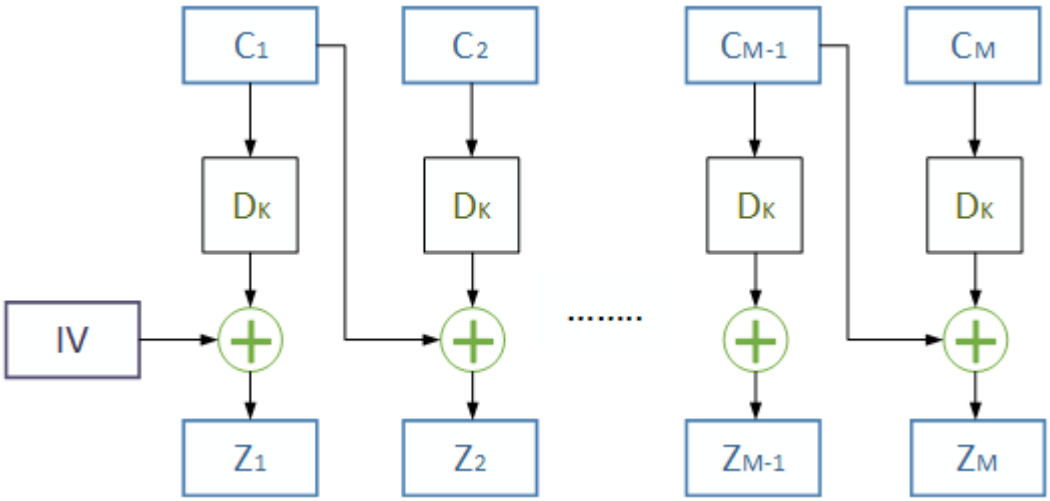
$$C_i = E(Z_i \oplus C_{i-1}, K), \tag{2.3}$$

$$Z_i = E(C_i, K) \oplus C_{i-1}, \tag{2.4}$$

kde i = 1, ..., M, C_0 = IV. Ze vzorce je tedy vidět, že každý blok je šifrován nejen na základě širovací funkce a šifrovacího klíče, ale také na základě kryptogramu z předchozího bloku. Pro šifrování prvního bloku je využíván tzv. Inicializační vektor. Princip režimu je naznačen na obrázcích 2.5 a 2.6.



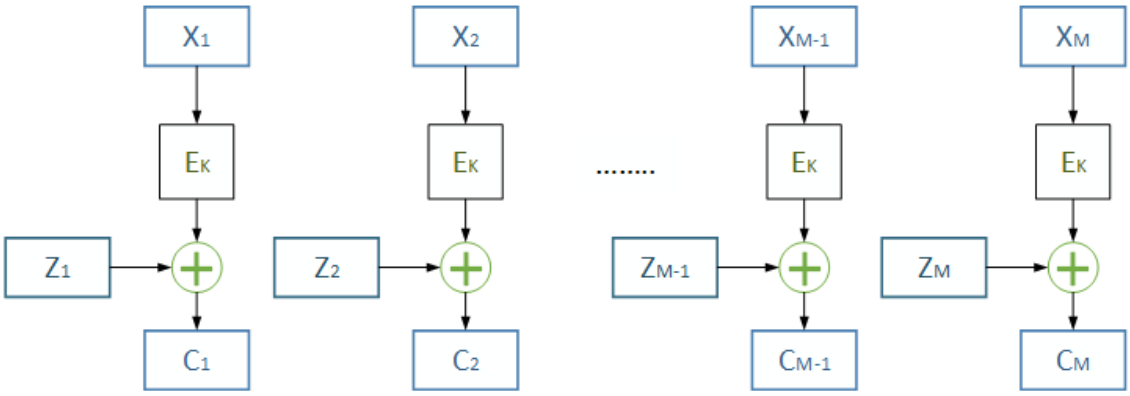
Obr. 2.5 Schéma šifrování v režimu CBC



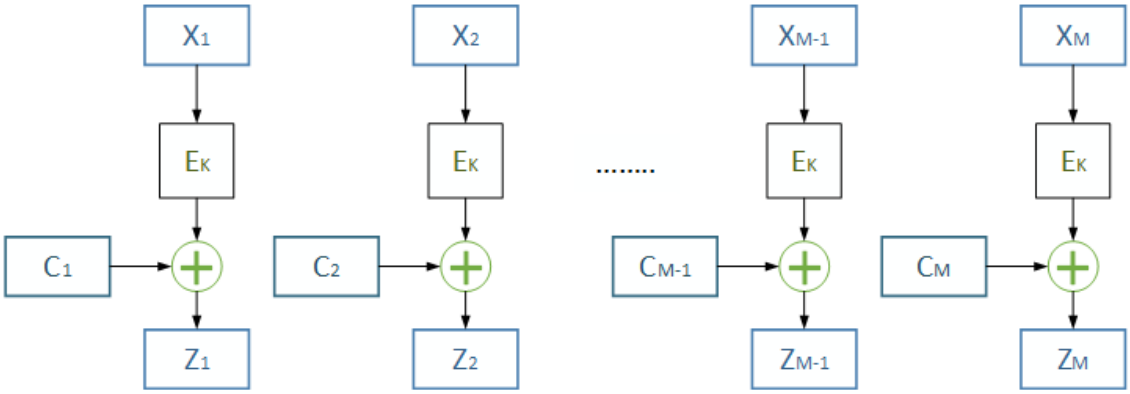
Obr. 2.6 Schéma dešifrování v režimu CBC

Režim CTR

Na rozdíl od předchozích variant režim CTR nešíří chybu, která během přenosu vznikne. Šifrování v tomto režimu probíhá po jednotlivých blocích, které jsou na sobě nezávislé, ale i přesto nejsou dva stejné bloky zašifrovány stejným kryptogramem. Toto je způsobeno čítačem, který je u každého bloku zašifrován a pomocí operace xor připojen k šifrovanému bloku zprávy. Inicializační hodnotu čítače je možné přenést společně s daty v nezašifrované podobě. Útočník nezná klíč a nemůže z něj vypočítat hodnotu, pomocí které by získal přenášenou zprávu. Schéma režimu CTR je zobrazeno na obrázcích Obr 2.7 a Obr 2.8.



Obr. 2.7 Schéma šifrování v režimu CTR



Obr. 2.8 Schéma dešifrování v režimu CTR

2.3.4 Padding

Implementace blokových šifer s sebou nese, oproti šifrám proudovým, jeden problém

a to je požadavek na délku dat. Délka šifrovaných dat musí být rovna nebo násobkem délky bloku, s kterým daná šifra pracuje. Tento problém je řešen operací, která se nazývá padding. Jedná se o doplnění příslušného počtu bajtů nebo bitů do posledního bloku dané šifry. Po rozšifrování dat na straně příjemce je nutné výplňová data odstranit, aby nedošlo ke změně přenášené informace. Existuje několik přístupů, jak s těmito doplňkovými bajty pracovat. Kromě blokových šifer využívají v kryptografii padding i hešovací funkce, které rovněž pracují s celými bloky dat. [1]

ANSI X.923

První typ algoritmu pro přidání výplňových bajtů je ANSI X.923. Data jsou do délky bloku doplněny nulami a poslední bajt určuje délku výplňových bajtů. V tabulce 2.1 je naznačeno doplnění 5 bajtů, kde písmena DD představují data. Hodnoty výplňových bajtů jsou uvedeny v hexadecimální soustavě.

.....	DD DD DD DD DD DD DD DD	DD DD DD **00 00 00 00 05**

Tabulka 2.1 Padding ANSI X.923

ISO 10126

Podle typu paddingu ISO 10126 jsou do plné délky bloku doplněna náhodná data a poslední bajt, stejně jako v předchozím případě, určuje délku výplňových bajtů. Příklad je uveden v tabulce 2.2.

.....	DD DD DD DD DD DD DD DD	DD DD DD **89 AB 1C 33 05**

Tabulka 2.2 Padding ISO 10126

PKCS7

V tomto případě je pro doplnění paddingu nejprve zjištěno, kolik bajtů schází do plné délky posledního bloku dat. Tato hodnota je pak vložena do všech výplňových bajtů. Pro absolutní jistotu, že nebudou při odstranění paddingu odebrány také datové bajty je potřeba doplnit padding i pro data, jejichž délka je násobkem délky bloku. Tento padding bude tvořen jedním celým blokem dat, který bude vyplněn bajty s hodnotou N, kde N představuje délku bloku. Příklad použití je uveden v tabulce 2.3. [12]

.....	DD DD DD DD DD DD DD DD	DD DD DD **05 05 05 05 05**

Tabulka 2.3 Padding PKCS7

ISO/IEC 7816-4

Další typ je bajtová podoba bitového paddingu, kde první doplněný bit má hodnotu logické 1 a ostatní bity jsou již nulové. Padding tedy bude tvořen na prvním výplňovém bajtu hodnotou 80 a na všech ostatních hodnotou 00. Hodnoty jsou uvedeny v hexadecimálním tvaru. Příklad paddingu je naznačen v tabulce 2.4.

.....	DD DD DD DD DD DD DD DD	DD DD DD **80 00 00 00 00**

Tabulka 2.4 ISO/IEC 7816-4

Zero padding

Poslední uvedený typ používá pro doplnění bloku samé nuly. Problém může nastat v okamžiku, kdy jsou data představovány souborem, který je zakončen nulovými bajty. Pro taková data tedy tento padding nelze použít. Jeho nasazení je možné například pro textové řetězce, které jsou představovány binárními daty. Použití je naznačeno v tabulce 2.4.

.....	DD DD DD DD DD DD DD DD	DD DD DD **00 00 00 00 00**

Tabulka 2.5 Zero padding

2.4 Asymetrická šifry

Asymtrické kryptosystémy jsou založeny na problému, který nelze jednoduše vyřešit. Prvním takto definovaným problémem, který můžeme v současnosti považovat za těžko vyřešitelný je problém faktorizace čísel. Šifra, která je na tomto problému založena se nazývá RSA. Na dalším problému, který se nazývá problém diskrétního logaritmu, je založen Diffie-Hellmanův protokol.

2.4.1 Diffie-Hellmanův protokol

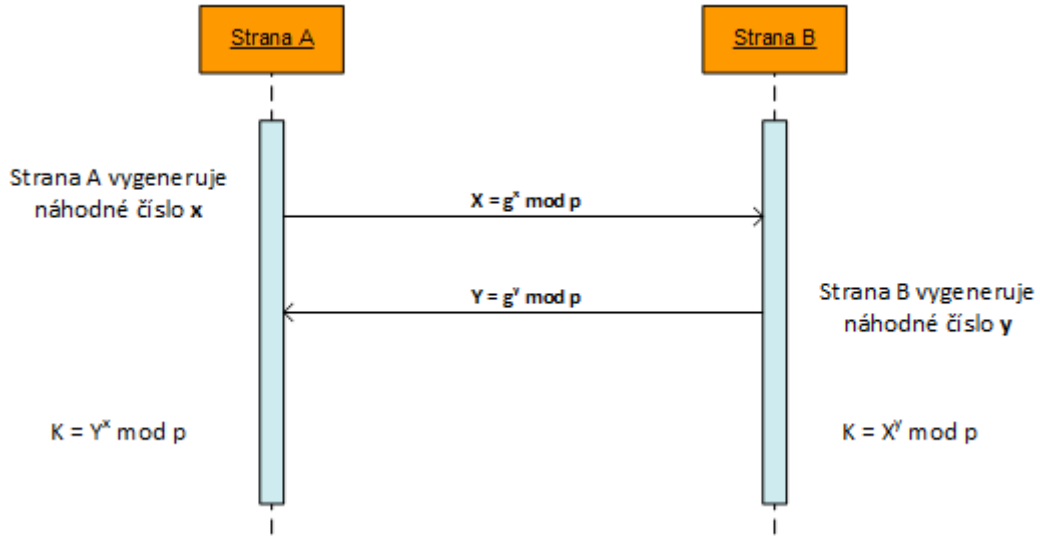
Jedná se o jednoho z nejznámějších zástupců asymetrických kryptosystémů, který je založený na řešení problému diskrétního logaritmu. Známe-li celá čísla x a g a velké prvočíslo p, pak lze snadno vypočítat ze vzorce (2.5) hodnotu y.

$$y = g^x \bmod p \tag{2.5}$$

Pokud bychom ovšem chtěli vypočítat ze vzorce (2.5) hodnotu x, za předpokladu, že známe všechny ostatní parametry, byl by to neřešitelný problém. [2]

Kryptosystémy, které jsou založeny právě na tomto problému a dále definují

podmínku, aby parametr g, měl takovou hodnotu, která by zajistila pro různá x dostatečně velkou množinu hodnot y. U DH algoritmu se využívá tzv. multiplikativní grupa G_{p-1}. Jedná se o algebraickou strukturu, jejíž prvky patří do množiny A = {1, 2, ..., .p-1}. Generátorem této grupy je takové g, pro které lze dosazováním všech hodnot do rovnice (2.1) vygenerovat celou množinu A.



2.9 Sestavení klíče pomocí Diffie-Helmannova protokolu

Průběh toho algoritmu je naznačen na obrázku 2.9. Důkaz, že tento způsob sestavení klíče doopravdy funguje je zobrazen v rovnici (2.6).

$$K = Y^x \bmod p = (g^y)^x \bmod p = (g^x)^y \bmod p = X^y \bmod p = K \qquad (2.6)$$

2.5 Čipové karty

Technologie, která byla dříve podporována na čipových kartách byla obvykle orientována na souborový systém a hlavní role operačního systému, který byl na kartě spuštěn, byla přistupovat k uloženým souborům a zajistit přístupová práva k nim. Novější karty již podporují VM běhové prostředí, které běží nad operačním systémem dané karty, a umožňuje spouštět platformě nezávislé aplikace, které jsou nazývány jako applety. Pro vytváření aplikací, které by bylo možné spouštět na těchto běhových prostředích existují různé implementace tzv. runtime libraries (běhových knihoven). Mezi nejpopulárnější z nich patří Java Card runtime environment (JCRE). Aplikace pro toto prostředí jsou vytvářeny v omezené verzi jazyku Java a používají pouze podmnožinu její runtime library. K dispozici jsou základní třídy pro práci se vstupem a výstupem, parsování zpráv a kryptografické operace. [19]

2.5.1 Secure element

Jedná se o čip, který je odolný proti manipulaci zvenčí a je schopen spouštět applety čipových karet. Toto zařízení představuje malý jednočipový počítač, který je složen z výpočetní jednotky CPU, pamětí ROM a EEPROM, operační paměti RAM a vstupně/výstupních portů. Čip může být také vybavena koprocesorem, který implementuje běžné šifrovací algoritmy, jako je například DES, AES a RSA. Zařízení, které jsou takto zabezpečeny, používají různé techniky, které zabrání získání dat jejím rozebráním či analýzou čipu. Využívají hardwarovou podporu ochrany paměti, aby data, která jsou v paměti uložena, byla dostupná pouze kartě samotné. Pro instalaci aplikace na kartu a v některých případech i pro samotný přístup na ni je vyžadováno použití kryptografického klíče. [17]

Secure element může být integrovaný do mobilních zařízení prostřednictvím SIM karty, SD karty nebo může být implementovaný na NFC čipu.

3 VOIP

VoIP (Voice over Internet Protocol) jedná se o technologii, která umožňuje uskutečňování hovorů prostřednictvím internetové sítě pomocí protokolů UDP/TCP/IP. Základní architektura pro vytvoření hovoru pomocí VoIP se skládá ze dvou koncových zařízení. Tento základní model je v praxi rozšířen o různé služby jako je například hlasová brána, kterou využívá protokol SIP. Dále je možné vést VoIP spojení přes tzv. gatekeeper (používaný u protokolu H.323) nebo v neposlední přes konferenční jednotku MCU (Multipoint Control Unit).

Pro přenos zvukového signálu prostřednictvím digitální technologie VoIP je potřeba provést převod mezi těmito signály. O tuto činnost se starají tzv. kodeky. Kodeky představují jasně definovaný postup, podle kterého je zvukový signál převeden do digitální podoby, respektive digitální data na analogový zvukový signál. Kodeků pro převod zvuku do digitální podoby existuje celá řada (např. G.711, G.722, G.723, G.729, SPEEX, iLBC atd.) a výběr příslušného kodeku, podle kterého je převod prováděn, určuje provozovatel služby, nikoli koncový uživatel. [3]

3.1 Signalizace a řízení hovoru

Signalizace se týká zřizování, vedení a ukončování hovoru, mezi volanými účastníky hovoru. Zahrnuje překlad adres volajícího a volaného účastníka, zjištěni, zda je k dispozici dostatečná přenosová kapacita, vyhledání cesty mezi volaným a volajícím. Funkcí signalizace je rovněž identifikace komunikujících stran. Při ustanovení hovoru je pomocí signalizačních protokolů rovněž dohodnuto, jaké kodeky budou během hovoru používány, jaké budou používány porty a další parametry přenosu. [5]

3.1.1 H.323

H.323 standard pro definici protokolů, které vytvářejí multimediální komunikaci v paketově orientovaných sítích. Používá se pro přenos audio-vizuálních dat v reálném čase. [5]

3.1.2 SIP

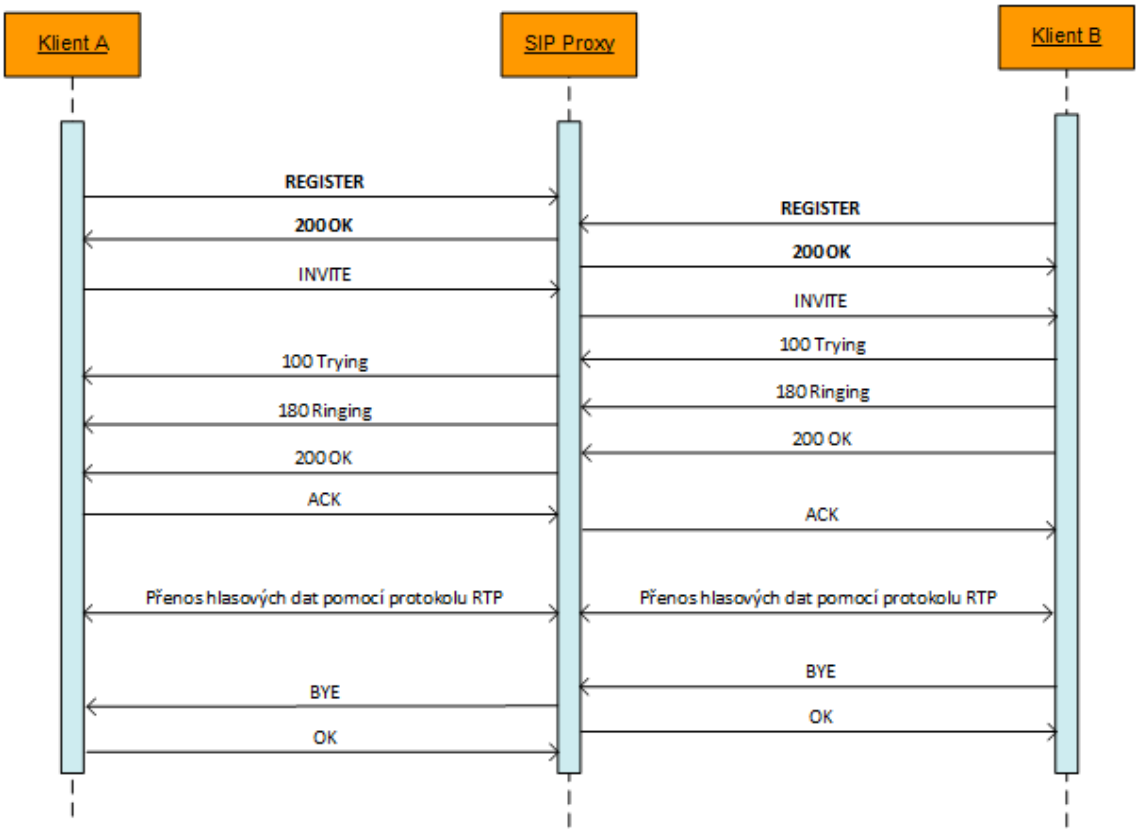
SIP (Session Initiation Protocol) je výhradně signalizační protokol, který se využívá k sestavení spojení, mezi dvěma či více účastníky. Protokol zajišťuje dohled na nad tímto spojením a také jeho ukončení. Protokol se nepoužívá k vlastnímu přenosu dat a neřeší řízení hovoru, které nejčastěji provádí protokol SDP (Session Description Protocol). Protokol SIP je možné využívat také pro textovou komunikaci nebo pro hraní her. [8] SIP je textově orientovaný protokol a příkazy, které jsou posílány prostřednictvím jednotlivých zpráv, jsou zapsány velkými písmeny obdobně jako tomu

je u protokolu HTTP, ze kterého protokol SIP vychází. Základní příkazy protokolu SIP:

- **REGISTER** - registrace účastníka na SIP Proxy serveru
- **INVITE** - požadavek na zahájení nové relace
- **ACK** - potvrzení navázání spojení
- **CANCEL** - přerušení navazování relace
- **BYE** - ukončení relace
- **OPTIONS** - požadavek na informace o možnostech protistrany, při kterém nevzniká spojení

Odpovědi protokolu SIP jsou doplněny o chybové kódy, které jsou obdobně, jako u protokolu HTTP tvořeny třemi čísly. Jsou rozděleny do těchto skupin:

- **1xx - průběh** - požadavek přijat, ale dosud nebyl dokončen
- **2xx - úspěch** - požadavek byl úspěšně proveden
- **3xx - přesměrování** - je vyžadována další akce pro dokončení požadavku
- **4xx - chyba klienta** - požadavek obsahuje syntaktickou chybu, nemůže být serverem zpracován
- **5xx - chyba serveru** - požadavek selhal, ačkoli byl pravděpodobně doručený správný požadavek
- **6xx - globální chyba** - požadavek nemůže být dokončen na žádném serveru



3.1 Komunikace pomocí SIP protokolu

Na obrázku 3.1 je naznačen průběh komunikace pomocí SIP protokolu. Klienti se nejprve musí zaregistrovat pomocí zprávy REGISTER. Registrace probíhá vůči SIP proxy serveru, který plní roli tzv. registrátora. Mezi komunikujícími klienty nemusí být pouze jeden SIP proxy server, ale může jich být několik. Úspěšná registrace je potvrzena zprávou OK s příslušným stavovým kódem. V tomto okamžiku je možné navázat telefonické spojení mezi přihlášenými klienty pomocí SIP zpráv.

3.1.3 SDP

Jedná se o jednoduchý internetový protokol, který pracuje na aplikační vrstvě ISO/OSI modelu. Využívá se k vyjednání parametrů multimediální relace. SDP, obdobně jako SIP, vytváří zprávy v textové podobě. Zpráva SDP je tvořena seznamem položek, které jsou tvořeny dvojicí atribut-hodnota a každá položka je představuje jeden řádek zprávy. Atribut jednotlivých položek je tvořen jednopísmenným identifikátorem, za kterým následuje rovnítko a za ním hodnota dané položky. V tabulce 3.1 jsou zobrazeny položky, které mohou být součástí protokolu SDP. Zároveň je u každé položky naznačeno, zda je povinná či nikoli. [15]

Položka	Název položky	Povinná položka
v=	Verze protokolu	Ano
o=	Původce a identifikátor relace	Ano
s=	Jméno relace	Ano
i=	Informace relace	Ne
u=	URI popisu	Ne
e=	E-mailová adresa	Ne
p=	Telefonní číslo	Ne
c=	Informace o spojení	Ano
b=	Šířka pásma	Ne
t=	Doba, po kterou je relace aktivní	Ano
r=	Počet opakování	Ne
z=	Úprava časového pásma	Ne
k=	Šifrovací klíč	Ne
a=	Atribut relace	Ne
m=	Název média a transportní adresa	Ne

Tabulka 3.2 Seznam položek protokolu SDP [15]

3.1.4 RTP

RTP (Real Time Protocol) je protokol, který standardizuje doručování dat mezi dvěma koncovými stanicemi, které je třeba přenášet v reálném čase. Řadí se mezi ně především audio a video přenosy. Protokol RTP "zabalí" jednotlivé části multimediálních dat do samostatných bloků, které jsou následně vkládány do UDP paketů. Protokol dále připojuje informace o typu přenášeného multimediálního obsahu, zajišťuje číslování paketů, přidává časové razítko vzniku jednotlivých datových částí, což usnadňuje práci na straně příjemce. RTP přenáší rovněž informaci o konkrétním streamu, neboť může být přenášeno několik samostatných proudů. [4]

3.1.5 Vzorkování zvukového signálu

Pro přenos hlasu prostřednictvím internetové sítě, je nutné zvukový signál, který je analogový, převést do digitální podoby. Digitalizaci je prováděna tak, že spojitý signál, který je představován vlnící se křivkou, je převáděn do podoby sekvence číselných hodnot, které jsou vyjádřeny v binární podobě. Tato transformace je provedena tvz. vzorkováním. Prvním parametrem, který je u vzorkování nutné nastavit, je správné množství vzorků, za jednotku času, aby bylo možné ze získaných dat zvukový signál opět sestavit. Pokud by byl počet vzorků příliš nízký, mohlo by dojít k tzv. aliasingu neboli podvzorkování. Pokud bychom ovšem signál vzorkovali příliš často, bude docházet k větší spotřebě šířky pásma. Druhým parametrem, který určuje kvalitu vzorkování je počet bitů, které budou pro jednotlivé vzorky použity.

Optimální množství vzorků pro digitální přenos zvukového signálu, aby bylo možné opětovně sestavit původní signál, udal v roce 1933 Harry Nyquist. Podle Nyquistovy věty je nutné odebírat a vzorky na dvakrát tak vysoké frekvenci, než je nejvyšší vzorkovaná frekvence. Pro VoIP byla určena maximální přenášená frekvence na 4KHz, která stačí pro věrný přenos hlasu. Pro VoIP se tedy používá zpravidla frekvence 8KHz. Ve VoIP jsou data vzorkována pomocí 8 bitového slova. Přes datovou síť je tedy nutné přenést datový tok 64Kbit/s.

3.1.6 Kodeky

Zvukový signál, který je získán pomocí vzorkování je nutné přenést v co nejkratším čase mezi komunikujícími stranami. Z tohoto důvodu je výsledný zvukový signál v binární podobě zakódován podle určitého algoritmu, aby bylo nutné přenášet přes síť co nejmenší objem dat. Metody, které se starají o převod signálu do digitální podoby a o kódování a dekódování dat se nazývají kodeky. V praxi se používá nepřeberné množství jak volně dostupných tak i proprietárních kodeků.

G.711

Jedná se o základní kodek veřejné telefonní sítě. Využívá pulzně kódovou modulaci (PCM) pro převod analogového signálu na digitální. Nepoužívá žádný kompresní algoritmus. Kodek převádí zvukovou informaci na datový proud od rychlosti 64Kbit/s. Jedná se o vzorkování na frekvenci 8KHz, kde každý vzorek je zakódován na 8 bitech. Např. kvalita zvukového záznamu na CD je určena frekvencí 44,1KHz a 16 bitovými vzorky. Z tohoto důvodu se dá u kodeku G.711 hovořit o ztrátové kompresi.

G.**726**

Jedná se o kodek založený na adaptivní diferenciální pulzně šířkové modulaci. Je možné ho používat v režimu 16Kbit/s až 32Kbit/s. Jedná se o jednoho z prvních kompresních kodeků. Kvalita přenášeného zvuku je stejná, jako u G.711, ovšem za využití poloviční šířky pásma. Kodek G.726 totiž nepřenáší jednotlivé zakódované vzorky, ale pouze informaci o rozdílu mezi současným a předcházejícím vzorkem.

G.**729A**

Jedná se o kodek, který využívá pro přenos dat šířku pásma o hodnotě 8Kbit/s. I přes tuto nízkou hodnotu dosahuje dobré kvality přeneseného zvuku. Využívá se totiž kompresní metody CELP, která je velmi populární pro kompresi hlasu. V rámci této metody je vytvářen tzv. codebook zvuků různých lidských hlasů. Mezi komunikujícími stranami se tedy neposílá zakódovaný vzorek, ale posílá se kód odpovídajícího vzorku hlasu. Tento kodek není volně dostupný a je možné jej používat pouze po zaplacení licenčního poplatku

GSM

Představuje kodek, který se vyznačuje podobnými parametry jako G.729A. Jeho

hlavní výhodou je fakt, že se jedná o kodek zdarma dostupný. Rovněž dokáže snížit šířku přenášeného pásma z 64Kbit/s na podstatně nižší hodnotu. Zde se jedná o 13Kbit/s.

iLBC

iLBC je ztrátový kodek. Stejně jako GSM je volně dostupný. Pracuje v režimu 15,2Kbit/s nebo 13,3Kbit/s. Dosahuje velkého stupně komprese, čímž podstatně zatěžuje procesory koncových zařízení.

Speex

Jedná se o kodek, který se umí změnit svoji vzorkovací frekvenci podle aktuální datové propustnosti v síti. Kodek pracuje s rychlostmi 2,15Kbit/s až 22,4Kbit/s. Je volně použitelný a je šířen pod licencí GNU.

3.2 API pro volání

3.2.1 Android SIP API

Operační systém Android poskytuje API, které umožňuje implementovat ve vlastních aplikacích protokol SIP. Android API poskytuje kompletní rozhraní pro SIP komunikaci, které zahrnuje správu hovoru, sestavení příchozích. V aplikaci pak není nutné implementovat řízení stavu hovoru nebo nahrávání či přehrávání přenášeného zvukového signálu. Vše je již vyřešeno v rámci této knihovny. API je možné použít kromě volání rovněž pro videokonference nebo pro posílání textových zpráv. Pro použití SIP API je nutné, aby koncové zařízení, na kterém bude výsledná aplikace nainstalována, umožňovalo bezdrátové připojení k internetu. Buď prostřednictvím mobilních dat nebo WiFi. Při vývoji je tedy nutné aplikace testovat na fyzických zařízeních. Nenižší podporovaná verze systému Android je 2.3. Aplikaci, která implementuje knihovnu SIP API je možné používat pouze za předpokladu, že má uživatel založený SIP účet u poskytovatele třetí strany. [11]

3.2.2 PJSIP

PJSIP je volně dostupná knihovna pro multimediální komunikaci. Implementuje protokoly SIP, SDP, RTP, STUM, TURN a ICE. Knihovna kombinuje signalizační protokol SIP s multimediálními funkcemi. Podporuje rovněž překlad adres. Pomocí této knihovny je možné implementovat aplikace pro různé operační systémy a také pro různá zařízení. Od desktopových počítačů přes mobilní telefony až k vestavěným systémům. PJSIP stojí na třech pilířích, které jsou pro komunikaci přes internet zásadní. Jedná se o multimediální komunikaci v reálném čase, implementace signalizačních protokolů a překlad IP adres pomocí techniky NAT.

Knihovna disponuje kvalitní dokumentací, která je dostupná na webu.

3.2.3 MjSip

Další knihovna, kterou lze využít k implementaci VoIP komunikace pro operační systém Andriod se jmenuje MjSip. Implementace je, jak již název napovídá, provedena podle standardu protokolu SIP. Knihovna je implementována v jazyku Java a je distribuována pod licencí GNU GPL. MjSip dodržuje architekturu protokolu SIP podle specifikace RFC 3261. Rozhraní ještě nad rámec RFC umožňuje kontrolovat průběh hovoru. Knihovna tedy zahrnuje všechny vrstvy a komponenty architektury SIP, rozšíření protokolu SIP definované IETF (např. RFC 3428 rozšíření pro zasílání zpráv, několik užitežných API pro kontrolu hovoru a referenční implementaci některých SIP systému, jako například Proxy server nebo Session Border Controller. [8]

3.2.4 Sipdroid

Sipdroid je open-sourceová aplikace, která implementuje SIP klienta pro operační systém Android. Sipdroid je svobodný software a jeho kód je možné volně distribuovat či upravovat. Licenční podmínky jsou stanoven podle licence GNU General Public License.

GNU General Public license

Jedná se o licenci pro svobodný software. Licence vyžaduje, aby byla odvozená díla distribuovaná pod toutéž licencí. Obecná veřejná licence GNU tedy zajišťuje, aby svobodu sdílení a úprav svobodného softwaru pro zajištění svobodného přístupu k tomuto softwaru pro všechny jeho uživatele.

3.2.5 Android RTP API

Android poskytuje API, které implementuje RTP protokol, prostřednictvím kterého je možné mezi dvěma a více účastníky po posílat proud dat prostřednictvím sítě. Pomocí této knihovny je aplikacím umožněno vytvářet VoIP aplikace, telekonference, streaming zvukových dat. Komunikaci je možno provádět přes jakoukoli dostupnou síť. Rozdíl mezi tímto API a SIP API je především v tom, že si programátor může naimplementovat vlastní spojení mezi volanými účastníky a není tak vázán na poskytovatele, přes kterého v aplikacích implementujících SIP API probíhá. Výsledná aplikace tudíž může komunikovat přímo se svým protějškem. [11]

4 IMPLEMENTACE

Implementace aplikace je tvořena dvěma částmi. První představuje aplikaci pro chytrá zařízení s operačním systémem Android a je vytvořena v programovacím jazyce Java. Druhou částí je aplikace pro čipovou SD kartu, na které je implementován symetrický šifrovací algoritmus AES. Implementace je provedena v jazyce Java Card.

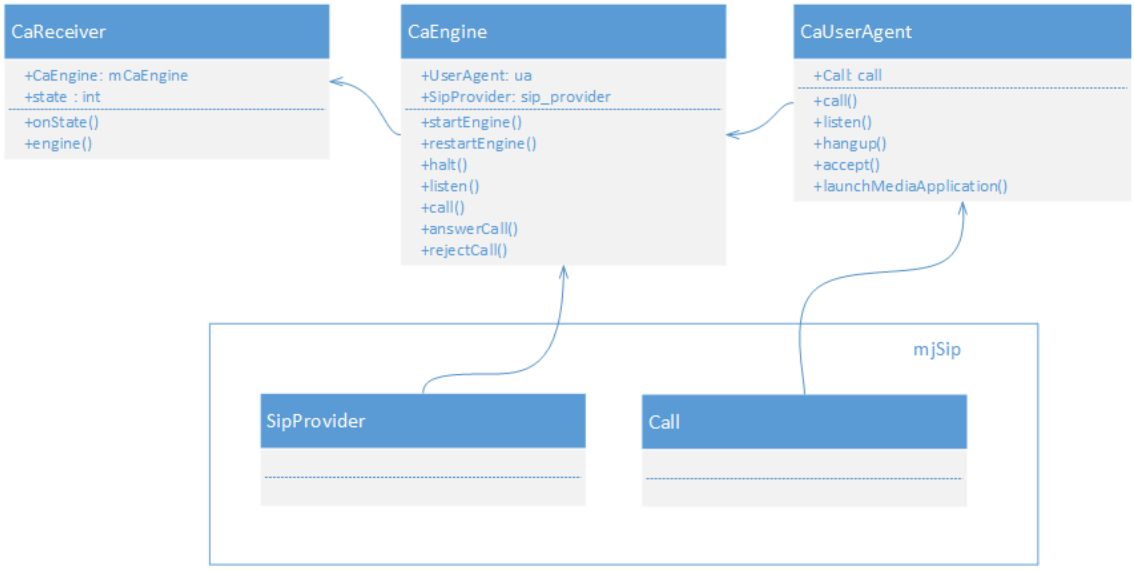
V aplikaci pro Android byla postupně vytvořena část pro přenos hlasových dat mezi dvěma účastníky prostřednictvím protokolu SIP. Tento protokol přenáší signalizační data mezi komunikujícími stranami a umožňuje ustanovit přenos hlasu mezi oběma účastníky. Další část představuje samotný přenos hlasových dat, který je implementován prostřednictvím protokolu RTP. Další část implementace představuje sestavení šifrovacího klíče, pomocí kterého bude přenos šifrován symetrickou šifrou. Pro sestavení klíče byl rozšířen protokol SIP o nový typ zprávy, který tuto operaci umožní provést. Pro zabezpečení přenášených dat protokolů SIP a RTP je implementován symetrický šifrovací algoritmus AES. Implementace algoritmu AES byla provedena jednak přímo v aplikaci pro prostředí Android tak také pomocí micro SD karty.

V průběhu vytváření této aplikace byla prováděna měření rychlosti jednotlivých kryptografických primitiv, ať se již jedná o symetrické šifry nebo o algoritmy pro sestavení šifrovacího klíč.

4.1 Implementace přenosu signalizačních dat pomocí protokolu SIP

Aplikace, která je v rámci tohoto projektu vytvořena je určena pro komunikaci v lokální síti a je používána v režimu peer-to-peer. Jedná se tedy o přímou komunikaci mezi klienty, mezi kterými není SIP server, který by komunikaci zastřešil. Toto řešení je zvoleno z toho důvodu, že hovor mezi účastníky bude šifrovaný a klasický SIP server, jehož prostřednictvím by protokol SIP spojoval klienty, by nebyl schopen tuto komunikaci zprostředkovat. Bylo by nutné rovněž implementovat tento server, na kterém by bylo možné provádět stejné šifrování, jako v klientských aplikacích. Z tohoto důvodu má implementace SIP protokolu v některých ohledech specifické chování.

Pro vytvoření signalizačního protokolu SIP byla využita knihovna mjSIP, která má implementovány knihovny pro vytvoření SIP volání. Jedná se o rozsáhlou knihovnu, která je využita k vytvoření spojení mezi komunikujícími stranami a k vytváření SIP zpráv. Nad touto knihovnou je vytvořena nadstavba, která umožňuje sestavit hovor. Jedná se o třídy, které jsou implementovány v balíčku cz.vutbr.xjonas04.callingapp.sip. Na obrázku 4.1 je naznačena komunikace aplikace s knihovnou mjSip.



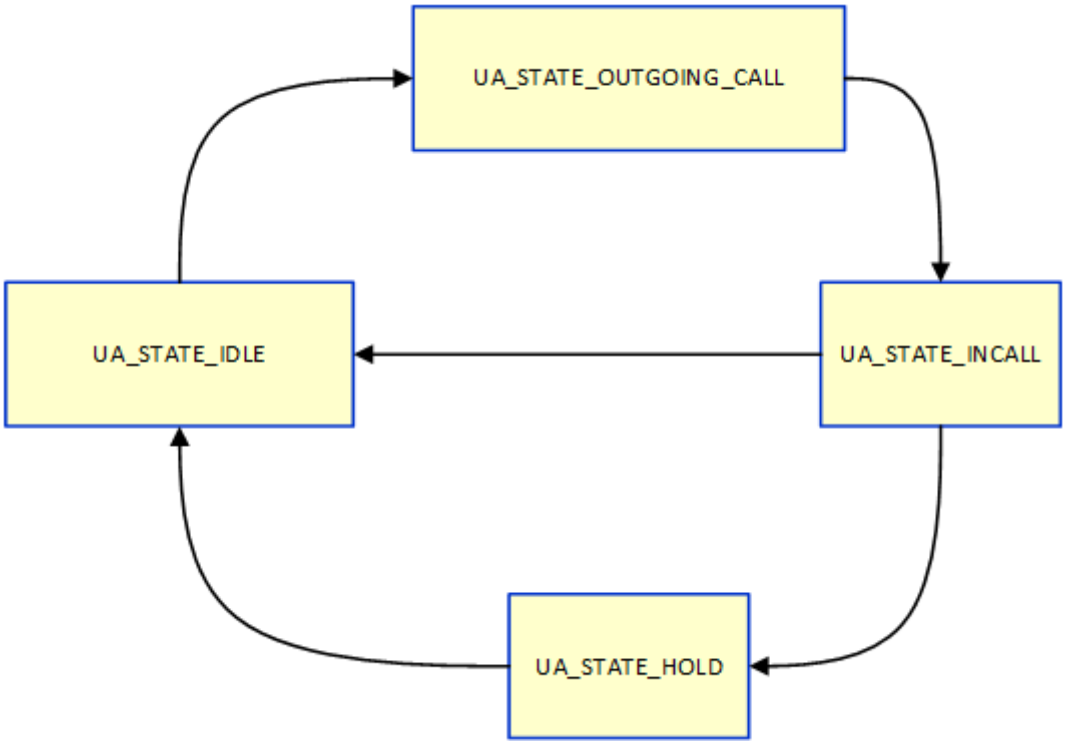
4.1 Diagram tříd - napojení aplikace na knihovnu mjSip

CaReceiver

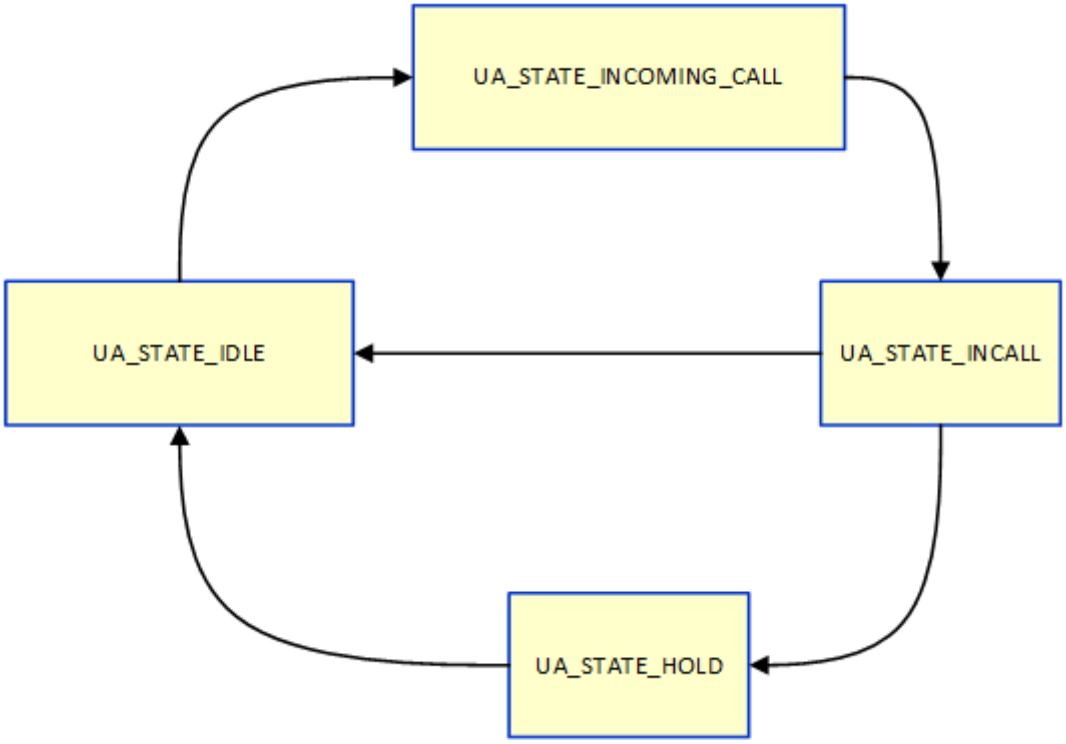
První z nich je třída CaReceiver. Jedná se o třídu, jejíž hlavním úkolem je vytvářet prostředníka mezi aktivitou aplikace CallingApp, která tvoří hlavní okno uživatelského rozhraní, a mezi implementací SIP protokolu. Třída CaReceiver uchovává ve statické proměnné odkaz aktivitu, prostřednictvím které byl SIP hovor inicializován, čímž je možné z této třídy provádět změny uživatelského rozhraní aplikace. Další funkcí této třídy je uchování stavu hovoru ve statické proměnné on_call. Rovněž je zde implementována metoda onState, která reaguje na změnu stavu protokolu SIP. Pořadí stavů, ve kterých se může aplikace nacházet je naznačena na obrázcích 4.2 a 4.3. Výhozí stav aplikace je vždy UA_STATE_IDLE. Seznam jednotlivých stavů, které mohou v aplikaci nastat:

- UA_STATE_INCOMING_CALL - aplikace je přepnuta do tohoto stavu v okamžiku, kdy je protokolem SIP doručena zprávu INVITE. Tato zpráva oznamuje, že protistrana chce vytvořit telefonické spojení. V tento okamžik je na uživatelském rozhraní zobrazeno tlačítko pro příjem hovoru a čeká se na reakci uživatele, který by hovor přijal.
- UA_STATE_OUTGOING_CALL - tento stav signalizuje, že uživatel daného zařízení kontaktoval žádostí o hovor protistranu. Tentokrát je tedy aplikace iniciátorem hovoru a tudíž odesílatelem zprávy INVITE.
- UA_STATE_INCALL - aplikace změní stav na tuto hodnotu v okamžiku, kdy je doručena zpráva o přijetí hovoru. V tomto okamžiku je vytvořeno RTP spojení, které umožní uskutečnit telefonický hovor mezi klienty.
- UA_STATE_HOLD - přepnutí do tohoto stavu nastane v okamžiku, kdy aplikace přijme zprávu INVITE, ale již se nachází ve stavu UA_STATE_INCALL. Pro aplikaci nenastane žádná změna.

- UA_STATE_IDLE - aplikace je přepnuta do tohoto stavu v okamžiku, kdy je hovor ukončen. Tento stav může nastat buď tak, že uživatel dané aplikace hovor ukončil nebo že přijal zprávu o ukončení hovoru od protistrany.



Obr 4.2 Diagram stavů - volání



4.3 Diagram stavů - příjem hovoru

Ve třídě je dále implementována metoda *engine()*, která vytváří instanci třídy CaEngine a následně si ji uloží do statické proměnné. Při dalším volání této funkce se již využívá dříve vytvořený objekt.

CaEngine

Třída CaEngine je třída, která zajišťuje volání funkcí knihovny mjSip. Prostřednictvím této třídy je umožněno aktivitě, která definuje uživatelské rozhraní, volat funkce knihovny mjSip. Činnost této třídy bude demonstrována popisem několika funkcí. Hlavní metodou, která je zde implementována se nazývá *startEngine()*. Tato metoda zajistí inicializaci socketu pro komunikaci prostřednictvím UDP. Při startu aplikace je socket inicializován s portem 5060, který slouží k naslouchání, zda byla přijata některá ze SIP zpráv. Při spuštění aplikace jsou tedy obě strany nastaveny tak, že se chovají jako server, který naslouchá na portu 5060. Pokud se uživatel dané aplikace rozhodne iniciovat hovor, je socket, naslouchající na portu 5060 zrušen a vytvořen socket nový, který bude komunikovat prostřednictvím náhodně vygenerovaného portu. Metoda, která je volána při vytváření hovoru a která zajistí reinicializaci socketu se nazývá *restartEngine()*. Při inicializaci aplikace je kromě nastavení socketu vytvořen objekt třídy CaUserAgent, který je hlavním prostředníkem pro odesílání a přijímání zpráv od knihovny mjSip.

CaUserAgent

Jedná se o třídu, která byla použita z projektu Sipdroid a byla modifikována pro účely této aplikace. Mezi hlavní úpravy patří volání metody *call()*, která byla upravena, aby bylo možné volat vzdáleného klienta prostřednictvím IP adresy a nikoli pomocí přihlašovacího jména. Další změny, které bylo potřeba v této třídě provést, se týkají především sestavení asymetrického šifrovacího klíče, které bylo nutné nejen do této třídy, ale do celé knihovny mjSip dopracovat.

4.2 Implementace přenosu hlasových dat pomocí protokolu RTP

Přenos hlasových dat prostřednictvím RTP protokolu byl implementován za pomoci knihovny Sipdroid. RTP komunikace je incializována pomocí třídy *JAudioLauncher*. V okamžiku, kdy je pomocí SIP protokolu ustanoven hovor, se v objektu *CaUserAgent* vytvoří instance třídy *JAudioLauncher*, dojde k vytvoření socketů pro RTP hovor a začnou se odesílat hlasová data vzdálenému uživateli. V rámci ustanovení hovoru jsou také nastaveny parametry hovoru, které byly ustanoveny prostřednictvím SIP protokolu. Nastaví se příslušný kodek, který si komunikující strany navzájem dohodly. Dále pak port a ip adresu protistrany.

4.3 Implementace měření času

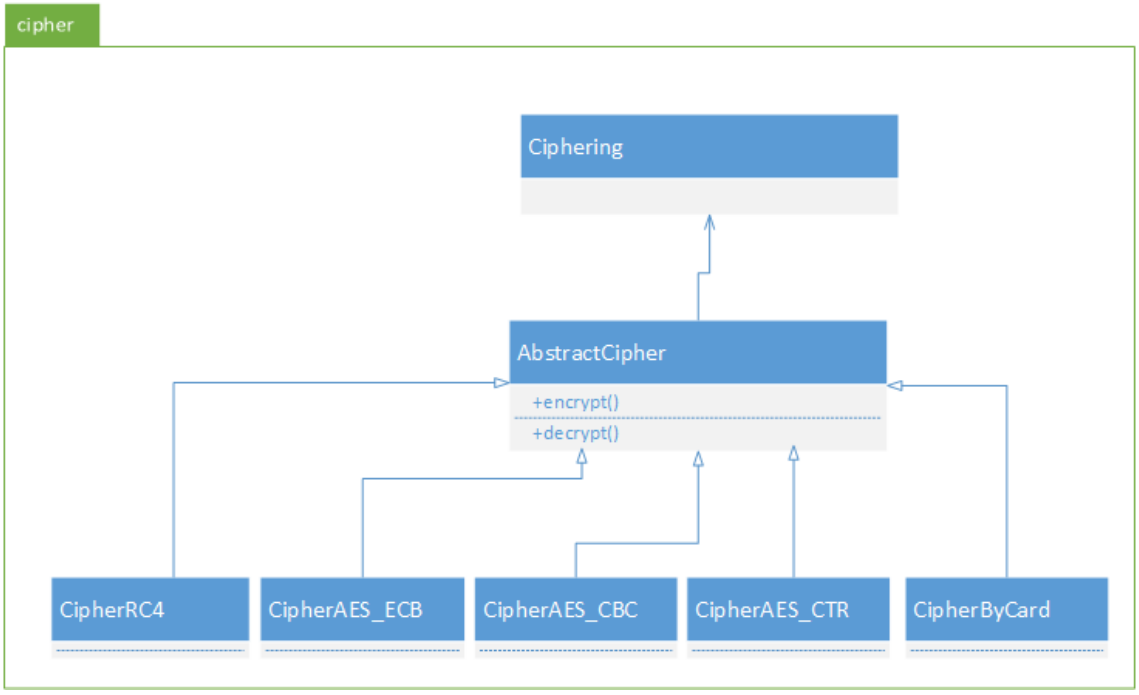
V aplikaci je vytvořena třída *TimeMeasurment*, kterou se používá k měření času algoritmů. Pro její použití je potřeba vytvořit instanci této třídy na začátku měřeného algoritmu. Při vytvoření této třídy se vytvoří časové razítko, které je uchováváno v objektu. Po ukončení algoritmu je zavolána metoda *stop()*, která odečte čas, kdy byl vytvořen daný objekt od času aktuálního a tím zjistí dobu trvání algoritmu. Výsledek je zapsán do konzole.

4.4 Implementace symetrických šifrovacích algoritmů

Pro šifrování dat pomocí symetrické šifry je možné použít režim blokové nebo proudové šifry. Použití proudové šifry je sice rychlejší, ale má nižší odolnost vůči kryptoanalýze oproti šifrám blokovým. Pro blokovou šifru tedy hovoří otázka bezpečnosti. S postupným zlepšováním výkonu počítačů se i rozdíl v rychlosti šifrování mezi těmito dvěma režimy snížil na minimum. Pro ověření tohoto faktu je tato skutečnost ověřena i na mobilních zařízeních, které mají oproti osobním počítačům stále nižší výkon. Blokové šifry mohou být implementovány v režimech ECB, CBC a CTR, přičemž režim ECB se nepovažuje jako bezpečný, protože pro stejná data a získáme

v rámci jednoho šifrování vždy stejný kryptogram. Hlavním algoritmem, který je pro implementaci blokových šifer použit je AES.

Implementace symetrických šifer má v jazyce Java oporu v knihovně *javax.crypto*. Ze všech možných variant byla vybrána symetrická šifra RC4 a bloková šifra AES v režimech ECB, CBC a CTR. Na obrázku 4.4 je zobrazen diagram tříd jednotlivých blokových šifer. Všechny objekt, představující jednotlivé šifry, jsou implementovány podle abstraktní třídy *AbstractCipher*, aby byla umožněna jednodušší práce s různými typy šifer a bylo možné k nim přistupovat jednotně. Kromě výše zmíněných čtyř šifer, které jsou pro šifrování použity je v diagramu tříd uveden objekt pro další s názvem *CipherByCard*. jedná se o třídu, ve které je implementována komunikace s micro SD kartou, na které je naimplementovaný šifrovací algoritmus.



Obr 4.4 Diagram tříd části aplikace, která implementuje symetrické šifry

4.4.1 Výsledky měření

Pro symetrické šifrovací algoritmy bylo provedeno měření rychlosti jejich provedení. Měření bylo prováděno na testovacím vzoru o délce 240 bajtů. Jak je vidět z výsledků v tabulkách níže, tak se u většiny algoritmů se pohybuje rychlost šifrování na hodnotách okolo 1ms. Výjimku tvoří akorát šifrování pomocí micro SD karty, kde je šifrování i dešifrování výrazně delší.

LG G3						
	RC4		AES ECB		AES CBC	
číslo měření	šifrování [ms]	dešifrování [ms]	šifrování [ms]	dešifrování [ms]	šifrování [ms]	dešifrování [ms]
1	0,62	1,32	0,30	0,42	1,14	0,88
2	0,51	1,11	0,17	0,29	0,82	0,73
3	0,60	0,95	0,15	0,24	1,02	0,75
4	0,61	0,95	0,17	0,24	0,96	0,74
5	0,55	1,18	0,15	0,24	0,86	0,75
6	0,59	0,87	0,15	0,24	1,32	0,71
7	0,51	0,84	0,15	0,41	0,79	1,42
8	0,51	0,81	0,15	0,38	0,68	1,71
9	0,70	1,20	0,20	0,29	0,77	0,84
10	0,49	0,84	0,18	0,37	1,03	0,76
Průměr	**0,57**	**1,01**	**0,18**	**0,31**	**0,94**	**0,93**

Tabulka 4.5 Měření rychlosti symetrických šifer na zařízení LG G3 (část 1)

LG G3				
	AES CTR		AES CBC prostřednictvím micro SD karty	
číslo měření	šifrování [ms]	dešifrování [ms]	šifrování [ms]	dešifrování [ms]
1	0,58	0,32	66,82	49,47
2	1,17	0,15	65,08	49,41
3	0,34	0,23	71,63	44,35
4	0,32	0,15	67,57	45,44
5	0,40	0,14	60,18	45,97
6	0,32	0,16	74,83	41,55
7	0,67	0,15	67,18	41,74
8	0,30	0,15	68,73	49,98
9	0,30	0,15	65,39	45,01
10	0,42	0,15	60,19	43,83
Průměr	**0,48**	**0,17**	**66,76**	**45,67**

Tabulka 4.6 Měření rychlosti symetrických šifer na zařízení LG G3 (část 2)

ASUS MeMO Pad 7						
	RC4		AES ECB		AES CBC	
číslo měření	šifrování [ms]	dešifrování [ms]	šifrování [ms]	dešifrování [ms]	šifrování [ms]	dešifrování [ms]
1	0,19	0,19	0,18	0,09	6,78	0,53
2	0,20	0,12	0,23	0,14	6,66	1,07
3	0,11	0,13	0,31	0,10	6,40	0,67
4	0,17	0,13	0,29	0,15	6,40	0,33
5	0,13	0,21	0,19	0,10	6,49	0,32
6	0,11	0,12	0,31	0,14	6,50	0,28
7	0,14	0,13	0,20	0,09	6,30	0,33
8	0,13	0,16	0,25	0,11	6,47	0,27
9	0,14	0,14	0,23	0,08	6,33	0,28
10	0,13	0,18	0,26	0,11	6,28	0,33
Průměr	**0,14**	**0,15**	**0,24**	**0,11**	**6,46**	**0,44**

Tabulka 4.7 Měření rychlosti symetrických šifer na zařízení ASUS MeMO Pad 7 (část 1)

ASUS MeMO Pad 7				
	AES CTR		AES CBC prostřednictvím micro SD karty	
číslo měření	šifrování [ms]	dešifrování [ms]	šifrování [ms]	dešifrování [ms]
1	7,27	0,16	64,97	45,41
2	8,80	1,24	61,41	47,48
3	6,33	0,20	63,89	39,10
4	6,22	0,10	61,75	37,58
5	7,21	0,26	65,45	37,09
6	6,20	0,09	62,91	53,62
7	6,42	0,11	57,66	47,07
8	6,11	0,08	76,91	44,19
9	6,12	0,11	61,01	47,80
10	6,16	0,38	56,56	37,28
Průměr	**6,68**	**0,27**	**63,25**	**43,66**

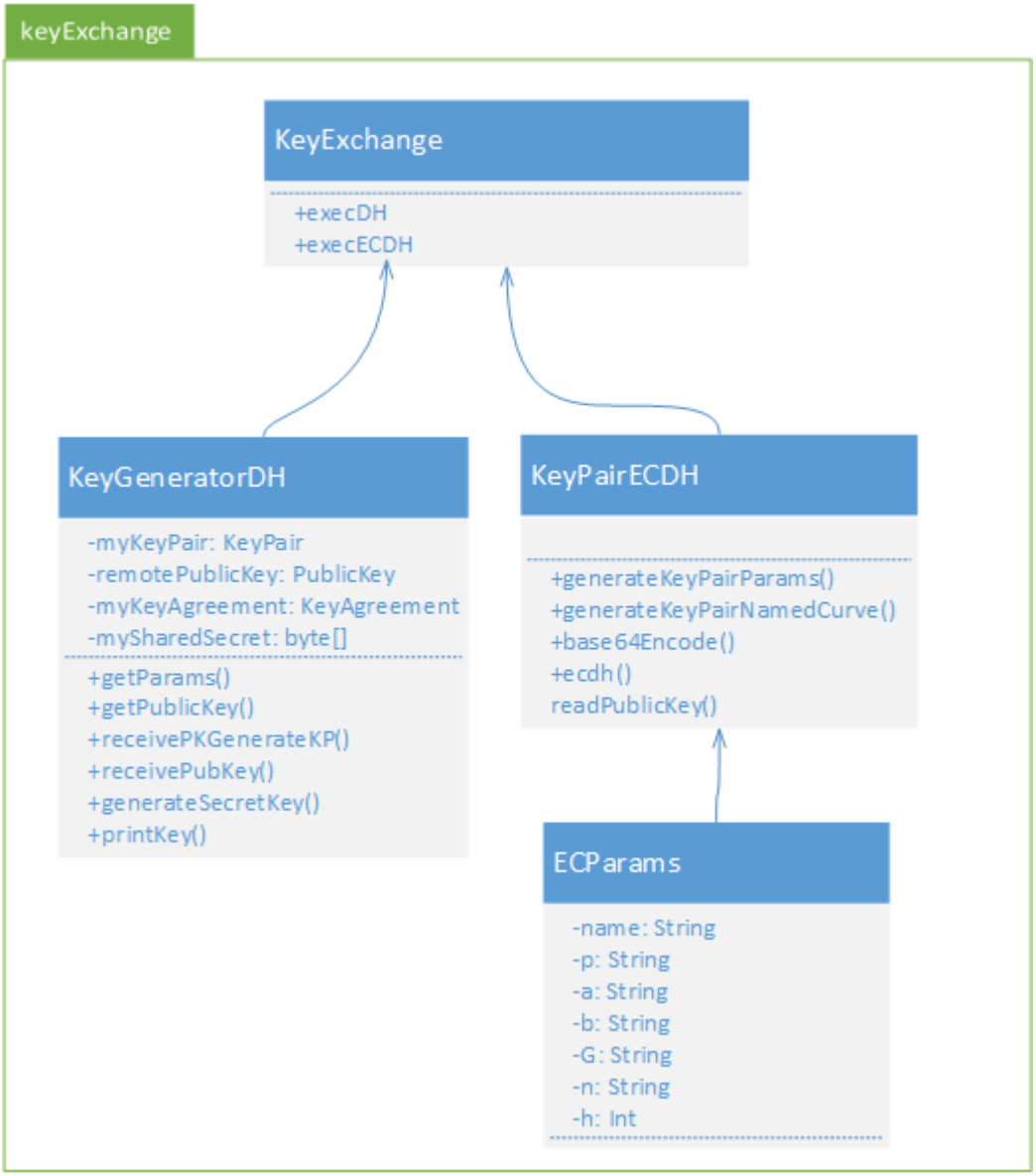
Tabulka 4.8 Měření rychlosti symetrických šifer na zařízení ASUS MeMO Pad 7 (část 2)

4.5 Implementace algoritmů pro sestavení klíčů

Pro sestavení klíčů pro symetrické šifry jsou použity dvě variace algoritmu Diffie-Helmann. První řešení je založeno na klasické variantě tohoto protokolu. Je implementován v třídě *KeyGeneratorDH* a využívá knihovny implementované v Javě. Konkrétně se jedná o java.security a javax.crypto. Třída *KeyGeneratorDH* implementuje v metodě *getParams* generátor náhodných parametrů pro vytvoření protokolu pro sestavení klíče.

Pro variantu Diffie-Helmanova protokolu není v jazyce Java podpora bez knihoven třetích stran. Pro tento protokol byla vybrána knihovna SpongyCastle, která je distribuována pod licencí MIT, která umožňuje použití knihovny zdarma. Implementace tohoto algoritmu za pomoci knihovny SpongyCastle je vytvořena ve třídě *KeyGeneratorECDH*. Postup vytvoření klíče je následující. Uživatel si nejprve pomocí metody *generateKeyParams* vygeneruje vlastní soukromý a veřejný klíč z parametrů, které jsou v aplikaci přednastaveny. Veřejný klíč je zakódován pomocí kódování base64 a odeslán protistraně. Protistrana veřejný klíč rozkóduje a předá jej se svým soukromým klíčem metodě *ecdh*. Výstupem této metody je tajný klíč. Tímto si způsobem je vygenerován tajný klíč, který mají obě strany stejný.

Diagram tříd, použitých pro implementaci Diffie-Hellmanova protokolu, je zobrazen na obrázku Obr 4.5.



Obr 4.9 Diagram tříd části aplikace, která implementuje symetrické šifry

4.5.1 Měření doby sestavení šifrovacího klíče

Jelikož je u algoritmu DH implementováno i vygenerování parametrů *p* a *q*, pomocí kterých je sestaven bezpečný klíč, bude čas náhodného generování od celkového času odečten, abychom mohli porovnat naměřené hodnoty s protokolem ECDH, kterému byly parametry nastaveny staticky. Z výsledků měření vyplývá, že je sestavení klíče pomocí algoritmu DH výrazně rychlejší.

LG G*3*			
Číslo měření	Doba generování parametrů [ms]	Celkový čas [ms]	**Čas sestavení klíče [ms]**
1.	2867	2870	**3**
2.	673	678	**5**

3.	218	222	**4**
4.	1629	1634	**5**
5.	2155	2158	**3**
6.	1291	1295	**4**
7.	1302	1306	**4**
8.	2489	2490	**1**
9.	46	53	**7**
10.	4127	4129	**2**
Průměr		1683,5	**3,8**

Tabulka 4.1 Výsledky měření DH algoritmu na přístroji LG G3

ASUS MeMO Pad 7			
Číslo měření	Doba generování parametrů [ms]	Celkový čas [ms]	**Čas sestavení klíče [ms]**
1.	1021	1085	**64**
2.	370	372	**2**
3.	26	29	**3**
4.	69	72	**3**
5.	273	276	**3**
6.	78	81	**3**
7.	72	74	**2**
8.	578	580	**2**
9.	24	26	**2**
10.	51	53	**2**
Průměr		264,8	**8,6**

Tabulka 4.2 Výsledky měření DH algoritmu na přístroji ASUS MeMO Pad 7

HTC Wildfier S			
Číslo měření	Doba generování parametrů [ms]	Celkový čas [ms]	**Čas sestavení klíče [ms]**
1.	4446	4459	**13**
2.	1508	1521	**13**
3.	5973	5982	**9**
4.	8810	8820	**10**
5.	1002	1013	**11**
6.	2149	2160	**11**
7.	457	470	**13**
8.	1353	1365	**12**
9.	512	522	**10**
10.	1017	1029	**12**

Průměr		2734,1	**11,4**

Tabulka 4.3 Výsledky měření DH algoritmu na přístroji HTC Wildfier S

LG G3	
Číslo měření	Čas sestavení klíče [ms]
1.	951
2.	875
3.	992
4.	978
5.	924
6.	984
7.	946
8.	921
9.	995
10.	950
Průměr	**951,6**

Tabulka 4.4 Výsledky měření ECDH algoritmu na přístroji LG G3

ASUS MeMO Pad 7	
Číslo měření	Čas sestavení klíče [ms]
1.	884
2.	878
3.	904
4.	914
5.	773
6.	745
7.	788
8.	774
9.	784
10.	772
Průměr	**821,6**

Tabulka 4.5 Výsledky měření ECDH algoritmu na přístroji ASUS MeMO Pad 7

HTC Wildfire S	
Číslo měření	Čas sestavení klíče [ms]
1.	7402

2.	6302
3.	6163
4.	6012
5.	5985
6.	5961
7.	5958
8.	5913
9.	5928
10.	5918
Průměr	**6154,2**

Tabulka 4.6 Výsledky měření ECDH algoritmu na přístroji HTC Wildfier

4.6 Zabezpečení aplikace pro přenos hlasových dat

Samotná aplikace pro přenos hlasových dat, jak byla do tohoto okamžiku popsána, umožňuje komunikaci mezi dvěma klienty. Přenos dat ovšem probíhá přes potenciálně nebezpečné prostředí, což v našem případě představuje lokální počítačová síť. Pokud by byl do stejné sítě připojen útočník, který by dokázal odposlouchávat komunikaci mezi účastníky hovoru, mohl by si následně zrekonstruovat celou komunikaci, která mezi nimi proběhla.

Současná kryptografie proto umožňuje použít dva způsoby, jak zabránit útočníkům přístup k citlivým datům, která jsou přenášena přes nechráněné médium. Jedná se o symetrické a asymetrické šifry. Každý z těchto dvou přístupů má ovšem problém, který stěžuje jeho použití. U symetrických šifer existuje problém v podobě distribuce šifrovacích klíčů u asymetrických šifer představuje problém rychlost šifrování. [2]

Řešením je tedy spojit přednosti symetrického a asymetrického šifrování. V praxi to vypadá tak, že je pomocí asymetrického šifrovacího algoritmu sestaven nebo doručen protistraně šifrovací klíč, pomocí kterého je pak celá komunikace šifrována.

Pro telefonní aplikaci, která je tvořena v rámci tohoto projektu bude tedy provedeno zabezpečení pomocí asymetrického Diffie-Hellmanova protokolu pro sestavení šifrovacího klíče. Samotný přenos bude následně využívat šifrování AES za použití sestaveného klíče.

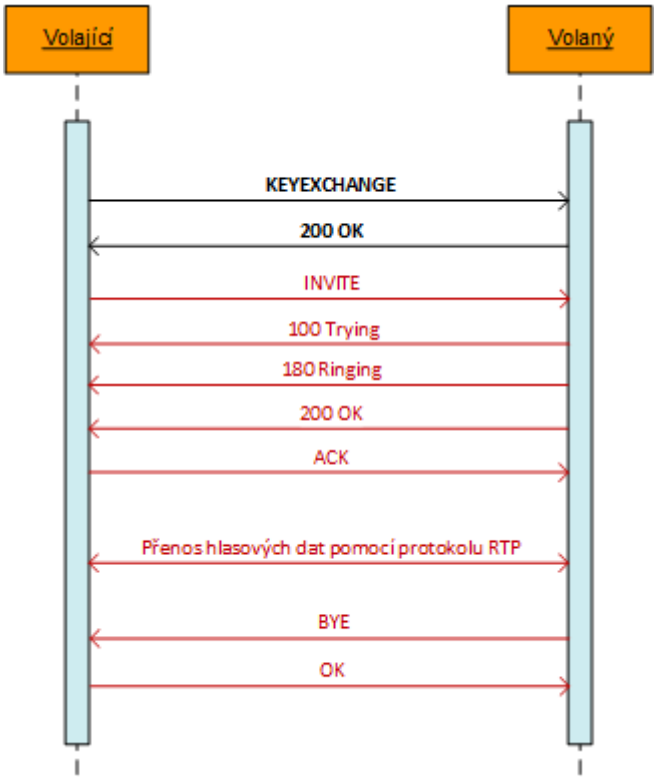
4.6.1 Implementace sestavení šifrovacího klíče

Jak již bylo uveden výše, pro šifrování komunikace mezi klienty bude nejprve nutné sestavit šifrovací klíč pomocí Diffie-Hellmanova protokolu. Pro tento účel je protokol SIP rozšířen o zprávu KEYEXCHANGE, která umožní oběma stranám sestavit symetrický šifrovací klíč. V těle této zprávy, kterou odesílá volající volanému, je zašifrované náhodné číslo *X*, pomocí kterého je volaná strana schopná sestavit

šifrovací. Následně volaný odesílá zprávu OK, která v těle přenáší zašifrované číslo *Y*, z kterého je volající schopen sestavit stejný klíč, který je následně použit pro šifrování přenosu.

Průběh zpráv je zobrazen na obrázku 4.6, na kterém jsou barevně odlišeny zprávy, KEYEXCHANGE a OK, které jsou přenášeny v nezašifrované podobě. Ostatní zprávy SIP protokolu, včetně přenosu hlasových dat pomocí RTP protokolu jsou šifrovány pomocí symetrickou šifrou AES pomocí klíče, který byl sestaven pomocí prvních dvou zpráv. Z diagramu je tedy patrné, že každý hovor bude mít vytvořený vlastní šifrovací klíč.

K přenosu parametrů DH protokolu, ze kterých je generován šifrovací klíč, se váže ještě jeden problém, který vyplývá ze skutečnosti, že pomocí protokolu SIP jsou přenášeny textové zprávy. Parametry pro sestavení klíče jsou ale generovány jako pole bajtů. Aby bylo možné posílat i obsah zprávy KEYEXCHANGE v textové podobě, je pole bajtů zakódováno pomocí algoritmu base64 do textové podoby.



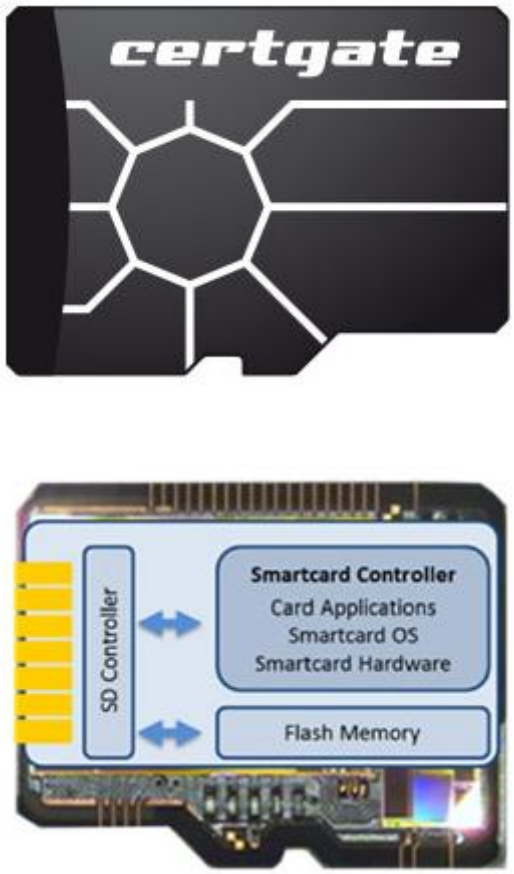
Obr 4.10 Průběh protokolu SIP rozšířeného o zprávu KEYEXCHANGE

4.6.2 Šifrování prostřednictvím SD karty

Pro implementaci šifrování na čipové kartě je použita micro SD karta od společnosti

Cretgate, která je zobrazena na obrázku 4.7. Aby bylo možné applet, vytvořený v jazyce JavaCard, je potřeba nainstalovat do počítače, na kterém bude probíhat instalace appletu na kartu, ovladač "certgate SmartCard microSD 1.0" pro 32 nebo 64 bitovou verzi operačního systému Windows.

Pro používání karty v mobilní zařízení s operačním systémem Android je rovněž potřeba doplnit ovladače, které umožní komunikaci s SD kartou. Jedná se o aplikce, cgCard-v1-Addon-for-cgSmartcardService-1.0.33-24.apk a cgSmartcardService-1.0.137-101.apk. Applet je možné nainstalovat na SD kartu pomocí konzole spuštěním skriptu *gppro_upload_jc222.bat,* který se nachází přímo ve složce *JavaCard.*



Obrázek 4.11 micro SD karta od společnosti Certgate [20]

Implementace appletu pro čipovou kartu využívá knihovny *javacard* pro základní operace a javacardx pro implementaci šifry AES v režimu CBC. Aplikace je vytvořena v objektu MainApp a skládá se z konstruktoru, ve kterém je volána funkce *register(),* která zajistí registraci appletu v běhovém prostředí JCRE. Další částí aplikace je implementace funkce *install()*, která je volána prostřednictvím JCRE a vytvoří novou instanci třídy MainApp. Poslední povinnou částí aplikace je funkce *process()*, která má jako vstupní parametr objekt APDU. APDU (Application Protocol Data Unit) je zpráva, prostřednictvím kterých je prováděna komunikace s aplikací na kartě.

Obrázek 4.12 APDU zpráva - požadavek

Na obrázku 4.8 je naznačeno tělo zprávy APDU. Délka každé části je 1 bajt, kromě pole Data, které nabývá velikosti L_c Význam jednotlivých částí APDU zprávy:

- **CLA** - Třída instrukce - určuje, zda se jedná o instrukci vlastní nebo vestavěnou
- **INS** - Kód instrukce, která specifikuje konkrétní příkaz
- **P1, P2** - Parametry příkazu
- **L_c** - Určuje délku dat, která jsou ve zprávě přenášena
- **Data** - Data přenášená ve zprávě, maximální délka je 255 která koresponduje s maximální hodnotou pole L_c, pro které je vyhrazen 1 bajt
- **L_e** - Určuje očekávanou délku dat v odpovědi

Obrázek 4.13 APDU zpráva - odpověď

Význam polí APDU zprávy, která je posílána jako odpověď:

- **Data** - Data, která jsou přenášena jako odpověď
- **SW1, SW2** - tzv. Command processing status, který určuje, zda příkaz na kartě proběhl úspěšně, hodnota pro správně provedený příkaz je 90 00 (hexadecimálně)

Pro komunikaci aplikace na platformě Android je potřeba použít knihovnu cgSmartcardServiceAPI-1.2.jar. Aby bylo možné se ke kartě připojit, je nutné vytvořit třídu, která implementuje rozhraní *SEService.Callback* z výše zmíněné knihovny. Dále je nutné v této třídě naimplementovat metodu *serviceConnected()*. Pro inicializaci komunikace mezi aplikací a micro SD kartou je nyní vytvořit instanci třídy *SEService*. Pokud je vytvoření tohoto objektu úspěšné bude zavolána dříve zmíněná metoda *serviceConnected()*.

4.6.3 Implementace šifrování protokolu SIP a RTP

Pro šifrování přenášených dat byl vybrán algoritmus AES v režimu CBC. Důvodem je dobrá odolnost proti kryptoanalýze [1] a poměrně slušné výsledky při měření rychlosti šifrování. Dalším důvodem, proč byl tento algoritmus vybrán je ten, že je možné jej implementovat na micro SD kartě. Aplikace tak budou navzájem komunikovat, i když

bude jedna strana šifrovat přeno na micro SD kartě a druhá strana přímo v aplikaci.

Další omezení, které přineslo použití čipové karty, byla maximální velikost přenášených dat mezi kartou a aplikací v telefonu. Maximální velikost je stanovena na 255 bajtů. Další omezení je, že je nutné posílat na kartu data již zarovnaná na násobek délky šifrovacího bloku. Z tohoto důvodu jsou šifrovaná data rozdělena na části o velikosti maximálně 240 bajtů, což je největší povolená délka, která je dělitelná délkou bloku, která je pro tento algoritmus nastavena na 16 bajtů.

Pro implementaci šifrování hlasového přenosu byla vytvořena třída *CipheringData*. Tato třída obsahuje statickou proměnnou typu *AbstractCipher*, která umožňuje nastavit šifrování pomocí libovolného algoritmu. Šifrovací algoritmus je nastavován pomocí metody *initCipherData()*. Ve třídě *CipheringData* jsou dále implementovány statické metody *encrypt()* a *decrypt()*. Tyto metody pak volají nad příslušným objektem, který byl nastaven metodou *initCipherData()* metody pro šifrování a dešifrování dat.

Důvodem, proč jsou ve třídě *CipheringData* statické metody, je skutečnost, že jsou v aplikaci šifrována data protokolu SIP i RTP. Každý z těchto dvou protokolů používá vlastní socket a bylo by tedy nutné provádět inicializaci algoritmu na dvou místech. Inicializace objektu *CipheringData* je používána v okamžiku, kdy se aplikace rozhoduje, zda bude šifrovat data na micro SD kartě nebo v telefonu. Nyní je aplikace nastavena tak, že používá vždy šifrování na kartě, pokud je karta dostupná.

Výsledek šifrování je možné vidět na obrázcích 4.14 a 1.15. První obrázek zobrazuje ustanovení hovoru pomocí protokolu SIP. Na prvních dvou řádcích v obou obrázcích je možné vidět zprávy, pomocí kterých dojde k sjednání šifrovacího klíče. Na druhém obrázku je pak možné pozorovat, že od 3. přenášeného paketu je tělo datagramu UDP zašifrováno.

```
10.42.0.31      SIP/SDP    592 Unknown request: KEYEXCHANGE sip:jirijonas@sip.antisip.com |
10.42.0.13      SIP/SDP    481 Status: 200 OK |
10.42.0.31      SIP/SDP    708 Request: INVITE sip:jirijonas@sip.antisip.com |
10.42.0.13      SIP        325 Status: 100 Trying |
10.42.0.13      SIP/SDP    585 Status: 180 Ringing |
10.42.0.13      SIP/SDP    636 Status: 200 OK |
10.42.0.31      SIP        471 Request: ACK sip:jirijonas@10.42.0.31:5060;transport=udp |
10.42.0.31      SIP        399 Request: BYE sip:jirijonas@10.42.0.31:5060;transport=udp |
```

4.14 Výpis komunikace SIP protokolu z programu Wireshark - nešifrovaná komunikace

```
10.42.0.31      SIP/SDP    592 Unknown request: KEYEXCHANGE sip:jirijonas@sip.antisip.com |
10.42.0.13      SIP/SDP    481 Status: 200 OK |
10.42.0.31      UDP        762 Source port: 42175  Destination port: 5060
10.42.0.13      UDP        362 Source port: 5060  Destination port: 42175
10.42.0.13      UDP        634 Source port: 5060  Destination port: 42175
10.42.0.13      UDP        362 Source port: 5060  Destination port: 42175
10.42.0.31      UDP        410 Source port: 42175  Destination port: 5060
```

4.15 Výpis komunikace SIP protokolu z programu Wireshark - šifrovaná komunikace

5 ZÁVĚR

V rámci této práce byly zkoumány možnosti vytvoření aplikace pro operační systém android, která by umožňovala vytvořit šifrovaný přenos hlasových dat.

Úvodní část se věnuje kryptografickým algoritmům, které by bylo možno použít streamovaný přenos dat. Je zde zkoumána, bezpečnost a také rychlost jednotlivých typů kryptosystémů. Dále jsou analyzovány symetrické šifry, které mohou být dále rozděleny na blokové a proudové. Blokové šifry mohou dále pracovat v různých režimech. Na konci kapitoly jsou rozebrány principy asymetrických šifer a možnosti jejich uplatnění a také je zde popsán tzv. Security element.

Druhá část této práce se zaměřuje na možnosti volání přes internet. Nejprve je popsán VoIP protokol obecně a dále je kladen důraz na signalizační protokoly a protokol RTP pro přenos dat v reálném čase. Nakonec jsou zmíněny konkrétní knihovny, pomocí kterých je možné implementovat telefonní přenos na systému Android.

Další část je zaměřena na implementaci jednotlivých částí. Nejprve je popsána implementace aplikace, která umožňuje uskutečnit nešifrovaný hovor. Implementace je popsána jak z pohledu protokolu SIP, ke kterému není potřeba SIP proxy server a uživatelé spolu komunikují přímo peer-to-peer, tak také z pohledu RTP protokolu. Následně je popsán způsob implementace měření rychlosti kryptografických algoritmů na mobilním telefonu. Dále je popsána implementace asymetrických protokolů pro sestavení šifrovacího klíče Jedná se o Diffie-Hellmanův protokol a Diffie-Hellmanův protokol s eliptickými křivkami, na kterých bylo provedeno měření rychlosti sestavení šifrovacího klíče. Z měření bylo zjištěno, že Diffie-Hellmanův protokol je podstatně rychlejší a byl proto použit ve výsledné aplikaci. Kapitola se zbývá rovněž implementací symetrických šifer a měření jejich rychlostí. Nejzajímavějším výsledkem měření byly výsledné hodnoty šifrování na microSD kartě. Oproti šifrování na telefonu trvaly jednotlivé operace přibližně o desetkrát déle. Na závěr kapitoly je uveden popis implementace šifrovacího algoritmu AES na micro SD kartě a komunikace s mobilní aplikací. Na závěr kapitoly je popsán způsob šifrování protokolů SIP a RTP, které uskutečňují přenos hlasu

Výsledkem práce je tedy aplikace pro šifrované volání, která umožňuje sestavit šifrovací klíč s protistranou pomocí Diffie-Helmannova protokolu a následně jej využít pro symetrické šifrování pomocí protokolu AES. Sestavení hovoru pomocí protokolu SIP a následný přenos hlasových dat je již uskutečňován v zašifrované podobě. Symetrické šifrování je implementováno jednak v samotné aplikace pro systém Android a také na micro SD kartě. Jak již výsledky měření napovídaly, tak komunikace šifrovaná pomocí algoritmu implementovaného na SD kartě je velmi sekaná a případný vzorek hlasu je přenášen velmi zpožděný.

Ačkoli byla vytvořena aplikace, která umožňuje šifrované volání, bezpochyby zde zůstal prostor pro další vývoj. Jednou z věcí, která by zvýšila bezpečnost volání by bylo podepisování parametrů Diffie-Hellmanova protokolu pomocí certifikátu, aby bylo možné ověřit identitu komunikujících stran. Další možností, jak aplikaci vylepšit by mohlo být rozšíření volání v rámci celého internetu. Pro tuto možnost by bylo zapotřebí implementovat SIP proxy server, který by dokázal přenášet šifrované zprávy.

LITERATURA

[1] MENEZES, Alfred, Paul C. VAN OORSCHOT a Scott A. VANSTONE. *Handbook of applied cryptography*. Boca Raton: CRC Press, c1997. Discrete mathematics and its applications. ISBN 0-8493-8523-7.

[2] BURDA, Karel. *Úvod do kryptografie. Úvod do kryptografie*. Brno: Akademické nakladatelství CERM, 2015. ISBN: 978-80-7204-925- 7.

[3] BAZALA, David. *Telekomunikace a VoIP telefonie 1.díl*. BEN - technická literatura, 2006. ISBN: 80-7300-201-9.

[4] RFC3550 [online]. 2003 [cit. 2016-10-13]. Dostupné z: https://www.ietf.org/rfc/rfc3550.txt

[5] Rodina protokolů TCP/IP, verze 2.7 [online]. 2011 [cit. 2016-10-13]. Dostupné z: http://www.earchiv.cz/l223/nahled.php3?l=11&me=1

[6] About PJSIP [online]. 2014 [cit 2016-10-14]. Dostupné z: http://www.pjsip.org/about.htm

[7] Getting Started: Building for Android [online]. 2014 [cit 2016-10-16]. Dostupné z: https://trac.pjsip.org/repos/wiki/Getting-Started/Android

[8] Welcome to MjSip [online]. 2014 [cit 2016-10-16]. Dostupné z: http://www.mjsip.org/

[9] LÓRENZ, R. *Bezpečnost - Proudové šifry, blokové šifry, DES, 3DES, AES, operacní módy* [online]. 2011 [cit. 2016-11-13]. Dostupné z: https://edux.fit.cvut.cz/oppa/BI-BEZ/prednasky/bez7.pdf

[10] RAMZAN, Zulfikar. Amin. *A Study of Luby-Rackoff Ciphers*. 2001 [cit. 2016-11-19]. Dostupné z: http://groups.csail.mit.edu/cis/theses/ramzan-phd.pdf

[11] Android Developer [online]. 2016 [cit. 2016-11-15]. Dostupné z: https://developer.android.com/

[12] RFC5652 - Cryptographic Message Syntax (CMS) [online]. 2003 [cit. 2016-10-13]. Dostupné z: https://tools.ietf.org/html/rfc5652

[13] KELLY, Timothy. *VOIP for dummies*. Hoboken: Wiley Publishing, Inc, 2005. ISBN: 0-7645-8843-5

[14] MILLER, Lawrence, GREGORY, Peter H. *SIP Communications for dummies*. Hoboken: Wiley Publishing, Inc, 2009. ISBN: 978-0-470-38114-4

[15] JOHNSTON, Alan B. *SIP: Understanding the Session Initiation Protocol*. Norwood: Artech House, 2009. ISBN: 978-1-60783-995-8

[16] HODSON, Ryan. *SIP: Android Programming Succinctly*. Morrisville: Syncfucion Inc, 2014.

[17] Accessing the embedded secure element in Android 4.x [online]. [cit. 2017-05-10]. Dostupné z: http://nelenkov.blogspot.cz/2012/08/accessing-embeddedsecure-element-in.html

[18] Secure Element Evaluation Kit for the Android [online]. [cit. 2017-05-10]. Dostupné z: http://seek-for-android.github.io/.

[19] Android secure element execution environment [online]. [cit. 2017-05-10]. Dostupné z: https://nelenkov.blogspot.cz/2012/08/android-secure-element-execution.html.

[20] cgCard - Integrated Security - certgate [online]. [cit. 2017-05-10]. Dostupné z: https://www.certgate.com/cgcard

SEZNAM POUŽITÝCH ZKRATEK

AES Advanced Encryption standard

API Application Programming Interface

CBC Cipher Block Chaining

CELP Code Excited Linear Predictions

CTR Counter mode

DH Diffie Hellman key exchange

ECB Electronic Codebook

ECDH Elliptic Curve Diffie Hellman key exchange

iLBC Internet Low Bitrate Codec

JCRE Java Card Runtime Environment

PCM Pulse-code modulation

QoE Quality of Experience

RTP Real-time Transport Protocol

SE Secure Element

SIP Session Initiation Protocol

URI Uniform Resource Identifier

VoIP Voice over Internet Protocol

WiFi Wireless Fidelity

SEZNAM PŘÍLOH

A OBSAH ELEKTRONICKÉ PŘÍLOHY

- callingapp.zip - zdrovojvé kódy aplikace pro šifrované volání pro operační systém android
- callingapp.apk - přeložená aplikace pro šifrované volání
- JavaCard.zip - zdrojové kódy appletu pro micro SD kartu
- diplomova_prace.pdf - elektronická verze diplomové práce