VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

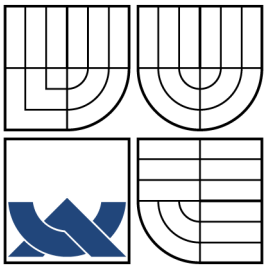
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

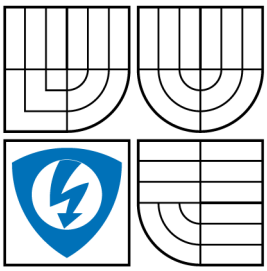
PLATEBNÍ SYSTÉM PRO OPERAČNÍ SYSTÉM ANDROID

BAKALÁRSKA PRÁCA
BACHELOR'S THESIS

AUTOR PRÁCE PAVOL IĽKO
AUTHOR



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND
COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

PLATEBNÍ SYSTÉM PRO OPERAČNÍ SYSTÉM ANDROID

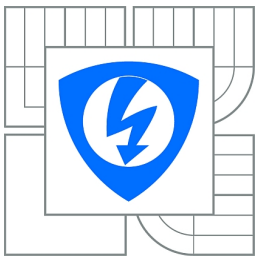
PAYMENT SYSTEM FOR OPERATING SYSTEM ANDROID

BAKALÁRSKA PRÁCA
BACHELOR'S THESIS

AUTOR PRÁCE — PAVOL IĽKO
AUTHOR

VEDÚCI PRÁCE — doc. Ing. KAROL BURDA, CSc.
SUPERVISOR

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Pavol Iľko	***ID:***	134505
Ročník:	3	***Akademický rok:***	2012/2013

NÁZEV TÉMATU:

Platební systém pro operační systém Android

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte a popište protokol ACP. Na jeho základě navrhněte platební systém pro mobilní zařízení s operačním systémem Android. Svůj návrh zdůvodněte jak z bezpečnostního tak i z provozního hlediska. Návrh prakticky zrealizujte a otestujte.

DOPORUČENÁ LITERATURA:

[1] Burda, K. aj.: Access Control Protocol (ACP). [Internet Draft] Internet Engineering Task Force, Fremont, 2011.
[2] Menezes A., Oorschot P., Vanstone S.: Handbook of Applied Cryptography. CRC Press, New York, 1997.

Termín zadání:	11.2.2013	***Termín odevzdání:***	5.6.2013

Vedoucí práce: doc. Ing. Karel Burda, CSc.
Konzultanti bakalářské práce:

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Začiatok práce sa zaoberá zhromažďovaním všeobecných informácií (súčasné platobné systémy, zabezpečenie, možné útoky), ktoré vytvárajú teoretický podklad nutný k ďalšej praktickej činnosti. Práca následne detailne popisuje protokol ACP. Praktický návrh platobného systému pomocou tohto protokolu je jej hlavnou súčasťou. V záverečnej časti je načrtnutý náhľad do problémov, ktoré sa vyskytujú pri implementácii danej aplikácie.

KĽÚČOVÉ SLOVÁ

ACP protokol, platobný systém, kryptografické metódy, Android, Rhino, Java, JavaScript

ABSTRACT

The introductory part of this thesis deals with the information (current payment systems, data security, computer attacks) that provide the theoretical framework needed for further practical task. Further on, the detail description of ACP protocol is offered. The work primarily focuses on the proposal of payment system based on ACP protocol. Lastly, some of the difficulties that occur when implementing such application are outlined.

KEYWORDS

ACP protocol, payment system, cryptographic methods, Android, Rhino, Java, JavaScript

IĽKO, Pavol *Platební systém pro operační systém Android*: bakalárska práca. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2013. 42 s. Vedúci práce bol doc. Ing. Karol Burda, CSc.

PREHLÁSENIE

Prehlasujem, že som svoju bakalársku prácu na tému „Platební systém pro operační systém Android“ vypracoval samostatne pod vedením vedúceho bakalárskej práce, využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej bakalárskej práce ďalej prehlasujem, že v súvislosti s vytvorením tejto bakalárskej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/nebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona č. 121/2000 Sb., o právu autorském, o právoch súvisejúcich s právom autorským a o zmeně niektorých zákonov (autorský zákon), vo znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhé, hlavy VI. diel 4 Trestného zákoníka č. 40/2009 Sb.

Brno ....

(podpis autora)

POĎAKOVANIE

Rád by som poďakoval vedúcemu semestrálnej práce pánovi doc. Ing. Karlovi Burdovi, CSc. za odborné vedenie, konzultácie, trpezlivosť a podnetné návrhy k práci. Zároveň chcem poďakovať Ing. Petrovi Ležákovi za odbornú pomoc pri vytváraní aplikácie. Taktiež by som rád poďakoval slečne Lýdii Rezničákovej za trpezlivosť a korekciu textu.

Brno . . . . . .

. . . . . .

(podpis autora)

OBSAH

ZOZNAM OBRÁZKOV

ZOZNAM TABULIEK

ÚVOD

V kapitole 1 sú načrtnuté súčasné mobilné platobné systémy. Podkapitoly 1.1 a 1.2 sa venujú súčasným online mobilným platbám podrobnejšie. Kapitola 2.1 popisuje možnosti autentizácie a kapitola 2.2 pojednáva o kryptografických metódach, ktoré je možné využiť pri platobnom systéme. Kapitola 3 charakterizuje útoky, ktoré by mohli byť použité proti navrhovanému systému. Protokol ACP je priblížený v kapitole 4. V podkapitole 4.2 je vysvetlený formát AVP správ, ktoré obsahuje ACP protokol a v podkapitole 4.3 a jej sekciách sú vysvetlené jednotlivé typy AVP správ a aké hodnoty obsahujú.

V ďalších kapitolách (5 a 6) sa práca zaoberá praktickým návrhom platobného systému. Kapitola 6 sprístupňuje popis znalostí, ktoré sú nutné na vývoj tejto aplikácie. V podkapitole 6.1 je priblížený operačný systém Android. Vývoj a dostupné nástroje využívané pri tvorbe aplikácií sú rozobrané v podkapitole 6.2. Podkapitoly 6.3 a 6.4 pojednávajú o hierarchii serveru a štruktúre aplikácie. Problémy, s ktorými sa môže vývojár stretnúť pri vytváraní danej aplikácie, sú načrtnuté v podkapitole 6.5.

1 SÚČASNÉ PLATOBNÉ SYSTÉMY

V dnešnej dobe existuje viacero možností ako platiť priamo pomocou mobilného telefónu. Podľa [4] môžeme spôsoby mobilných platieb rozdeliť na:

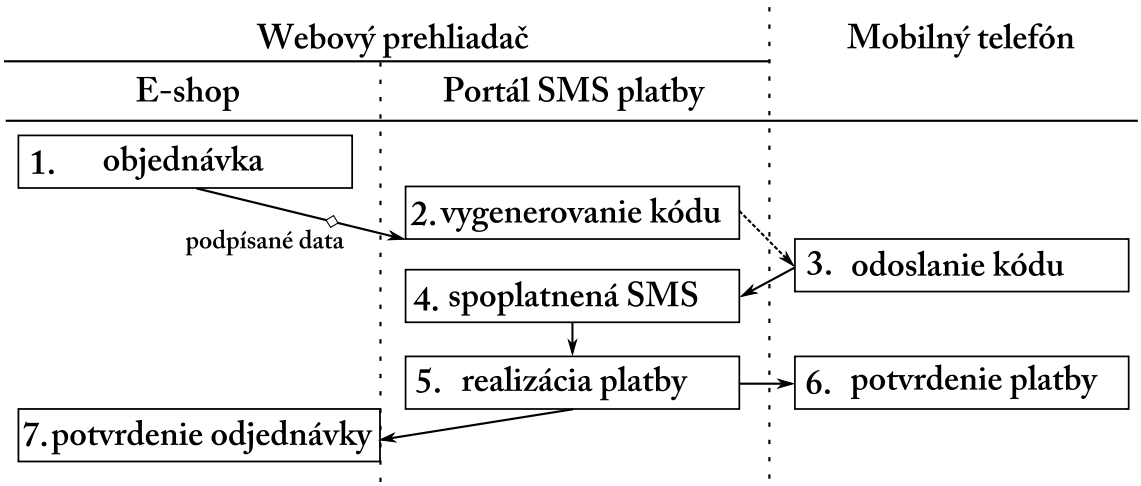
a) automatické – klient zavolá na špeciálne číslo a následne si vďaka hlasovému menu tlačítkami na mobilnom telefóne zvolí príslušné operácie. Vyúčtovanie platby riadi mobilný operátor.
b) SMS – klient napíše prázdnu alebo textom špecifikovanú SMS správu na určené číslo, kde kontrolér vyhodnotí platbu priamo k telefónnemu číslu.
c) GSM SIM Toolkit – klient komunikuje pomocou SMS správ, ale zadávanie príkazov je oveľa pohodlnejšie, lebo môže používať metapríkazy (makrá) naprogramované v SIM karte telefónu.
d) WAP (Wireless Application Protocol) – klient komunikuje pomocou textového prehliadača, ktorý má nainštalovaný na svojom mobilnom telefóne. Na prenos dát sa používajú SMS správy alebo operátorom podporované dátové protokoly.

1.1 SMS platba online

V súčasnosti je možnosť platiť malé platby (do cca. 500 kč) pomocou mobilného zariadenia online. Základný postup platby spočíva v tom, že po objednaní tovaru z E-shopu je zákazník presmerovaný z web serveru obchodníka na portál poskytovateľa služby mobilných SMS platieb a taktiež sú predané podpísané autentizačné parametre. Portál zabezpečí overenie správnosti údajov a identifikáciu zákazníka. Portál vygeneruje z podpísaných údajov kód, ktorý zákazník odošle so svojho mobilného telefónu pomocou SMS správy na telefónne číslo platobného portálu. Následne príde zákazníkovi spoplatnená SMS správa v cene objednaného tovaru. Overenie kreditu na mobilnom zariadení a aj samotnú realizáciu platby zariadi platobný portál. Po ukončení platby je zákazník presmerovaný na web server obchodníka spoločne s informáciou o výsledku tranzakcie. Postupnosť SMS platby online môžeme vidieť na Obr. 1.1.

1.2 Mobito

Služba Mobito umožňuje zákazníkovi platiť mobilným zariadením pomocou jednoduchého prepojenia s bankovým účtom (Obr. 1.2) alebo s platobnou kartou. Služba Mobito je zákazníkom poskytovaná prostredníctvom Mobilného Platobného Systému, ktorý prevádzkuje spoločnosť MOPET CZ a.s.



Obr. 1.1: Postupnosť SMS platby online.

Podľa [14] táto spoločnosť figuruje na finančnom trhu ako vydavateľ elektronických peňazí. Ku jej kompetenciám patrí aj poskytovanie platobných služieb, ktoré bolo schválené na základe právo mocného rozhodnutia o povolení k činnosti inštitúcie elektronických peňazí vydaného Českou národnou bankou.

Služba Mobito je určená pre platby elektronickými peniazmi medzi zákazníkmi navzájom a zároveň pre platby zákazníkov elektronickými peniazmi za tovar a služby poskytované obchodníkmi, ktorí sú užívateľmi Mobilného Platobného Systému.

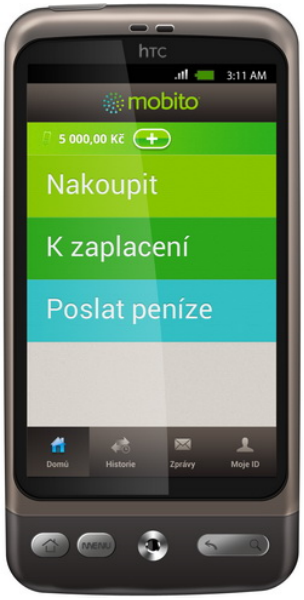
Platby môžu slúžiť aj k zasielaniu a prijímaniu darov alebo na iné finančné vysporiadanie medzi zákazníkmi navzájom. Každý užívateľ, ktorý chce využívať služby Mobito musí byť registrovaný na Mobito Portále `www.mobitoplatito.cz`, kde ako identifikátor slúži reálne telefónne číslo, alebo respektíve číslo vygenerované na Mobito Portále.

Nabitie služby Mobito a vybitie služby Mobito a všetky platby v Mobilnom Platobnom Systéme sú prevádzané v pomere 1:1 medzi elektronickými peniazmi a českou korunou.

Službu je možné používať na všetkých mobilných zariadeniach pomocou „správ na displej“ (USSD kódov, Unstructured Supplementary Service Data) zavolaním na číslo **135#* cez prehľadné menu s číselnými voľbami, ako ukazuje Obr. 1.4. Výnimkou sú iba telefóny od kanadskej spoločnosti RIM, ktoré nepodporujú USSD správy. Na ovládanie nepotrebuje zákazník ani pripojenie k internetu, postačí GSM signál a celá komunikácia je zadarmo.

Pre mobilné zariadenia s operačným systémom Android a iOS si firma pripravila aplikácie, pomocou ktorých sa dá Mobito ovládať ešte jednoduchšie. Aplikácie sú dostupné pre Android na GooglePlay a pre iOS na AppleStore.

Obr. 1.2: Prepojenie účtu MOBITO s Českou sporiteľnou.



Obr. 1.3: Mobito na Androide.

Obr. 1.4: Mobito na rôznych zariadeniach.

2 ZABEZPEČENIE

2.1 Autentizácia strán

Identifikáciu žiadateľa môžeme overiť rôznymi spôsobmi. Podľa [3] môžeme rozdeliť triedy autentizácie na:

a) autentizácia znalosťou – klient svoju identitu preukáže znalosťou tajnej informácie (hesla, PIN kódu, odpoveďou na dopredu dohodnutú otázku atď.)
b) autentizácia biometriou – klient svoju identitu preukáže svojimi biometrickými charakteristikami (napríklad otlačkami prstov, hlasom, rohovkou a podobne)
c) autentizácia predmetom – klient svoju identitu dokazuje predmetom (napríklad identifikačným preukazom, platobnou kartou, čipovou kartou a podobne).

Každá trieda autentizácie má svoje výhody a nevýhody a preto sa často kombinujú. Napríklad pri platbe kartou je potrebný PIN kód (autentizácia znalosťou) a platobná karta (autentizácia predmetom).

Ďalej popíšem autentizácie, ktoré by mohli byť využité pri platobnom systéme s protokolom ACP.

2.1.1 Jednorazové heslo

Jednorazové heslo, ako z názvu vyplýva, je možné použiť na autentizáciu iba raz. Klient dostane zoznam jednorazových hesiel a rovnaký zoznam je uložený v prístupovom zozname. Po použití hesla ho kontorlér vymaže z prístupového zoznamu. Z bezpečnostného hľadiska je znížené riziko zneužitia zachyteného hesla útočníkom. Nevýhodou je distribúcia a bezpečné ukladanie zoznamov hesiel, kvôli možnosti úniku hesiel predtým ako budú vydané klientovi alebo odcudzeniu s nedostatočne zabezpečeného úložiska.

2.1.2 Opakované heslo

Náhodne opakované heslo sa používa opakovane, ale nepravidelne. V súčasnosti sa jedná o takzvané grid karty. Plastové karty s veľkosťou platobnej karty. Na grid karte je vytlačená tabuľka, najčastejšie rozdelenie je do 8 stĺpcov a 6 riadkov. V každej bunke tabuľky je uvedený jeden, prípadne viacero znakov. Túto tabuľku má k dispozícii klient aj kontrolér. Pri autentifikácii zašle kontrolér súradnice buniek tabuľky a klient potom musí zaslať kód, ktorý je v daných bunkách.

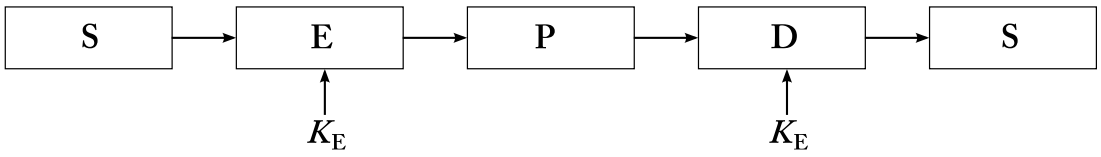
2.1.3 Časovo premenné heslo

Premenlivosť kódu pri časovo premennom hesle sa dosahuje zavedením závislosti vypočítaného kódu na čase. V zariadení je uložený tajný individuálny kľúč K a hodiny, ktoré zariadeniu poskytujú aktuálny čas t. V prípade potreby autentizačného kódu je požadovaný kód vypočítaný pomocou hashovacej funkcie $h_t = \mathrm{H}(K, t)$. Hashovacia funkcia H zaistí, že útočník nedokáže zo zaslaného hesla vypočítať tajný kľúč K. Heslo h_t je zaslané kontroléru, ktorý pozná kľúč K a aktuálny čas t a urobí vlastný výpočet. Keď sa hesla zhodujú, tak kontrolér prijme autentizáciu.

2.2 Kryptografické metódy

2.2.1 Symetrická šifra

Klient zašifruje správu (S) s kľúčom K_E na kryptogram $\mathrm{C} = \mathrm{E}(S, K_E)$ a odošle kontroléru, cez prenosový kanál (P). Klient aj kontrolér majú k dispozícii rovnaký tajný kľúč K_E. Kontrolér kryptogram dešifruje a ak platí $\mathrm{D}(C, K_E) = \mathrm{S}$, tak bude správa čitateľná. Princíp symetrického šifrovania ukazuje Obr. 2.1. Nevýhodou použitia symetrickej šifry je nutnosť mať rovnaký šifrovací kľúč na oboch stranách.



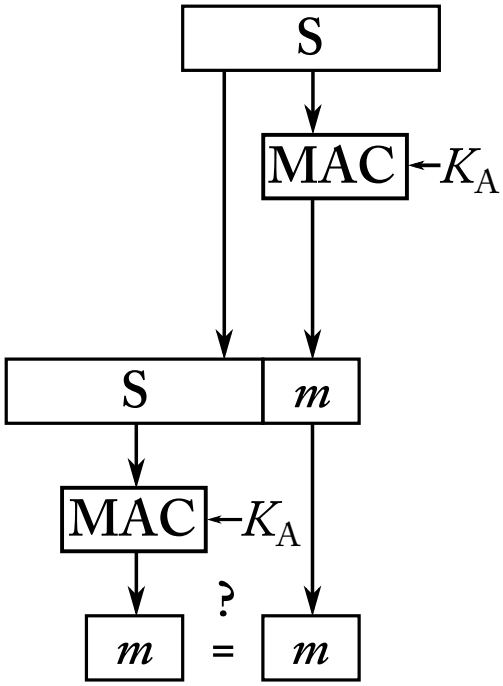
Obr. 2.1: Princíp symetrického šifrovania.

Na autentizáciu sa pri symetrickej šifre využíva MAC (Massage Authentication Code) a HMAC (Keyed-Hashing for Message Authentication Code). Sú to jednocestné funkcie so správou S s tajným parametrom K_A (Autentizačný kľúč), ktoré vytvoria autentizačný kód. HMAC na výpočet využíva hashovaciu funkciu H (SHA-1, MD5, RIPEMD-160, PANAMA, SHA256 a iné.) a dve rôzne nemenné hodnoty ipad (bit 0x36 opakovaný X krát) a opad (bit 0x5C opakovaný X krát). MAC využíva šifrovanie DES (Data Encryption Standard). Zápis funkcii:

- Funkcia MAC: $m = \mathrm{MAC}(S, K_A)$.
- Funkcia HMAC: $m = \mathrm{H}(K_A \text{ XOR opad} \mid\mid \mathrm{H}(K_A \text{ XOR ipad} \mid\mid S))$

Vytvorený kód sa pripojí za správu S. Kontrolér pomocou kľúča K_A vypočíta s prijatej správy $m' = \mathrm{MAC}(S, K_A)$, prípadne HMAC a porovná ho s prijatým kódom

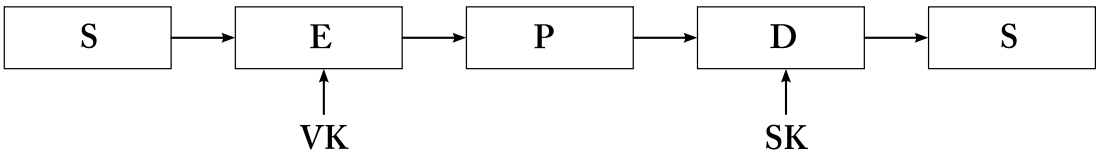
m. Ak sa tieto kódy zhodujú správa je autorizovaná. Priebeh autentizácie ukazuje Obr. 2.2.



Obr. 2.2: Autentizácia pri symetrickom šifrovani.

2.2.2 Asymetrická šifra

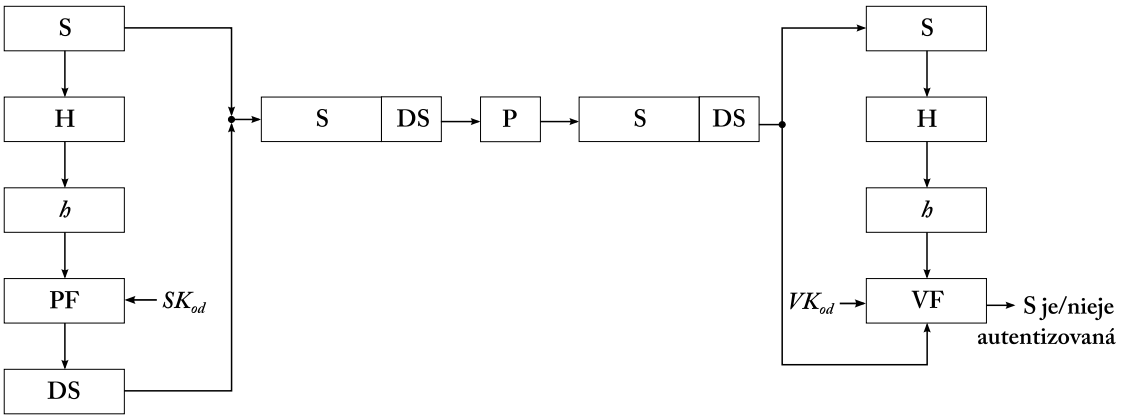
Asymetrická šifra odstraňuje nevýhody symetrickej šifry. Správa je dešifrovaná iným kľúčom než bola šifrovaná. Žiadateľ aj kontrolér má vlastnú dvojicu kľúčov, jeden verejný kľúč (VK) a jeden súkromný kľúč (SK). Správa je zakódovaná verejným kľúčom, ktorý je k dispozícii všetkým, ale len žiadateľ ju môže dekódovať svojim privátnym kľúčom. Princíp asymetrického šifrovania ukazuje Obr. 2.3.



Obr. 2.3: Princíp asymetrického šifrovania.

Na autentizáciu sa pri asymetrickom šifrovaní využíva viacero krokov, ako je možné vidieť na Obr. 2.4. Najskôr sa vytvorí kryptogram pomocou algoritmu RSA:

$C_1 = \mathrm{E}_{\mathrm{RSA}}(K_{\mathrm{AES}}, VK_{pr})$, kde K_{AES} je kryptografický kľúč AES a VK_{pr} je verejný kľúč príjemcu. Ďalej sa vytvorí zo správy S pomocou hashovacej funkcie H hash h. Potom sa pomocou hashu h a súkromného kľúča odosielateľa SK_{od} vytvorí digitálny podpis $DS = \mathrm{PF}(h, SK_{od})$. PF je podpisová funkcia najčastejšie DSA. Za správu sa pripojí digitálny podpis $S||DS = B$. Následne sa vytvorí kryptogram $C_2 = E_{\mathrm{AES}}(B, K_{\mathrm{AES}})$. Kryptogramy C_1 a C_2 spoločne s VK_{od} sa pošlú prenosovým kanálom P príjemcovi. Príjemca najskôr dešifruje kryptogram C_1 a keď má správny súkromný kľúč dostane šifrovací kľúč AES ($\mathrm{D}_{\mathrm{RSA}}(C_1, SK_{pr}) = K_{\mathrm{AES}}$). Následne pomocou tohto kľúča dešifruje kryptogram C_2 a dostane správu S s digitálnym podpisom ($\mathrm{D}_{\mathrm{AES}}(C_2, K_{\mathrm{AES}} = B = S||DS$). Potom príjemca zo správy S vytvorí kontrolný hash h' ($h' = \mathrm{H}(S)$). Pomocou hashu h' a verejného kľúča odosielateľa (VK_{od}), vložených do verifikačnej funkcie $\mathrm{VF} = (h', VK_{od})$, príjemca zistí, či je správa autorizovaná alebo nie. Nevýhodou asymetrického šifrovania je vysoká výpočtová náročnosť. Autentizácia asymetrickou šifrou sa využíva napríklad v protokole SSH (secure shell).



Obr. 2.4: Autentizácia pri asymetrickom šifrovaní.

2.2.3 TLS

TLS (Transport Layer Security) je protokol, ktorý pracuje na transportnej vrstve OSI modelu. Podľa [5] TLS protokol možeme implementovať na dvoch úrovniach:

a) TLS Record Protocol – zabezpečí vyjednanie súkromia a spoľahlivé pripojenie medzi klientom a serverom
b) TLS Handshake Protocol – autentizuje komunikáciu medzi klientom a serverom pri začatí komunikácie, dohodne použitie šifrovacieho algoritmu a šifrovacieho kľúča. Následne môže aplikačný protokol posielať dáta.

3 MOŽNÉ ÚTOKY

Útoky, ktoré by mohli byť použité na platobný systém môžeme rozdeliť na tri časti:

a) útok na dôvernosť – útočník odpočúva dáta zo siete sa snaží sa analyzovať dátový tok
b) útok na autentickosť – útočník z odchytených dát získa údaje, ktorými sa môže voči serveru autentizovať (heslo, kľúč a pod.)
c) útok na dostupnosť – útočník zaťaží server napríklad DoS, DDOS útokom a ten sa stane nedostupným.

3.1 Replay attack

Replay attack je druh tzv. „Man in The Middle“ útoku. Útočník odchytí zašifrovaný priebeh komunikácie a následne môže ho zopakovať s tým, že nemusí poznať jej reálny obsah. V prípade, že server nedetekuje duplikovanú požiadavku, môže to vyvolať napríklad opakované zadanie platobného príkazu.

3.2 DoS, DDoS

Útok Denial of Service (DoS) je takzvané odoprenie služby. Týmto útokom môže útočník spôsobiť dočasnú alebo trvalú nedostupnosť určenej služby pre užívateľov, ako príklad môžem uviesť stratu sieťového spojenia zahltením šírky pásma, preťaženie zdrojov cieľového systému a podobne. Tento typ útokov môžeme podľa [16] rozdeliť na:

a) Flood attack – útočník zasiela množstvo paketov, ktoré postihnutý počítač nedokáže spracovať a tým zhodí celý systém
b) Ping of Death Attack – útočník využíva program ping, ktorý posiela pakety väčšie ako 65,535 bajtov
c) SYN Attack – na zahltenie komunikácie použije útočník množstvo SYN správ
d) Teardrop Attack – útočník zasiela IP pakety, ktoré sú poškodené
e) Smurf Attack – využíva zasielanie ICMP Echo Request paketov na broadcastovú adresu.

Útok DDoS (Distributed Denial of Service) je založený na princípe DoS útoku s tým rozdielom že DDoS útok prebieha z viacerých počítačov naraz, čo výrazne zvyšuje účinnosť toho útoku. Na začiatok viacero počítačov (desiatky až tisíce) vysielajú útok na server, ktorý sa v následku pretečenej pamäte, zahltenej šírky pásma alebo podobných ťažkostí reštartuje. Potom už stačí aj menšie množstvo počítačov (približne do desať), aby dokázali udržať server trvalo nedostupný. Na tento typ

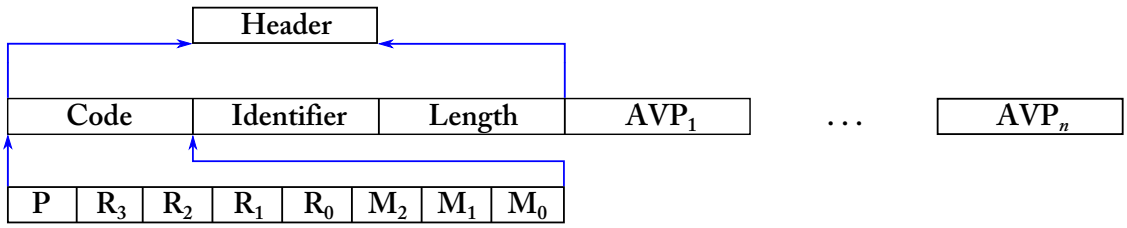
útoku sa často využívajú takzvané zombie počítače (počítače napadnuté crackerom, trójskym koňom prípadne iným vírusom), ktoré útočník ovláda pomocou botnet serveru. Botnetom označujeme sieť napadnutých (zombie) počítačov, ktoré vykonávajú nežiadúcu činnosť, ako sú spomenuté DDoS útoky ako aj rozosielanie spamu a podobné útoky.

4 PROTOKOL ACP

Protokol ACP (Access Control Protocol) je univerzálny obojstranný protokol určený na riadenie prístupu k aktívam vyvinutý na Vysokom učení technickom v Brne. Protokol nie je závislý na použitej prenosovej vrstve a ani ju nijako nešpecifikuje. Strana, ktorá požaduje prístup k aktívam sa nazýva žiadateľ a strana, ktorá tieto aktíva poskytuje sa nazýva poskytovateľ. Jeden kompletný beh protokolu ACP, tj. postupnosť správ medzi žiadateľom a poskytovateľom, ktorá súvisí s riadením prístupu k požadovaným aktívam sa nazýva tranzakcia. Protokol umožňuje zaistiť komunikáciu medzi viacerými uzlami, ktoré sa na danej tranzakcii podieľajú.

4.1 Formát správy ACP

Formát správ protokolu ACP je definovaný v [17] a je uvedený na obr. 4.1.



Obr. 4.1: Formát správy protokolu ACP.

Správa sa skladá z bloku Header (hlavičky), za ktorou nasleduje n blokov AVP (Attribute-Value Pair). Každý blok AVP obsahuje užitočné informácie prenášane správou. Štruktúru AVP správy môžete vidieť na obr. 4.2.

Header správy obsahuje pole Code (kód) s veľkosťou 8 bitov. Binárny formát je v tvare $P\ R_3\ R_2\ R_1\ R_0\ M_2\ M_1\ M_0$. Najvýznamnejší bit P slúži k odlíšeniu protokolu ACP od protokolu EAP a musí byť nastavený na 1. Nasledujú 4 rezervné bity R_3–R_0, tie musia byť nastavené na 0. Nasledujúce 3 bity M_2–M_0 určujú typ správy.

Ďalej Header obsahuje blok Identifier (identifikátor tranzakcie) s veľkosťou 3 bajty. Identifier jednoznačne priradzuje správu k tranzakcii iniciovanej jedným smerom. Aby sme mohli úplne jednoznačne identifikovať tranzakciu, tak hodnotu Identifier musíme skombinovať s bitom M_0, ktorý určuje či správa mieri od Iniciátora k Adresátovi alebo opačne.

Pole Length (dĺžka) s veľkosťou 3 bajty udáva celkovú dĺžku správy v bajtoch. Do tejto dĺžky je zhrnutá aj hlavička. Maximálna dĺžka správy je $(2^{24} - 1)$bitov $\approx$ 2MB.

Protokol ACP má definované nasledujúce typy správ:

a) Start – Zahájenie novej tranzakcie. Odosielateľom je vždy žiadateľ. Správa Start môže obsahovať kód požadovaného aktíva aj typ autentizácie.
b) Offer – Poskytovateľ posiela žiadateľovi ponuku dostupných aktív alebo typov autentizácie, ktoré sú pre dané aktívum poskytovateľom požadované.
c) Specification – Odosiela ju vždy žiadateľ, ktorý si vyberá aktívum, o ktoré má záujem, alebo typ autentizácie, ktorý chce použiť.
d) Request – Poskytovateľ posiela žiadateľovi požiadavku na autentizáciu.
e) Response – Odosiela žiadateľ a je využívaná k autentizácií žiadateľa.
f) Finish – Poskytovateľ ukončuje tranzakciu a v prípade úspechu posiela požadované aktívum.

Tranzakciu protokolu ACP ilustruje Tab. 4.1. Prvý stĺpec tabuľky uvádza správy, ktoré odosiela žiadateľ. V druhom stĺpci sú uvedené správy odoslané poskytovateľom a tretí stĺpec je venovaný poznámkam. Každý riadok tabuľky reprezentuje jeden krok protokolu ACP.

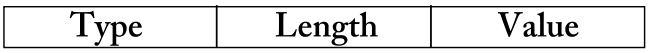
Tab. 4.1: Schéma tranzakcie protokolu ACP.

Žiadateľ	Poskytovateľ	Poznámky
Start →		Zahájenie tranzakcie. Zahajuje vždy žiadateľ.
	← Offer	Výber požadovaného aktíva. Pokiaľ žiadateľ požadované aktívum uvedie v správe Start alebo existuje jediná možnosť, tak môže byť tento krok vynechaný.
Specification →		
	← Offer	Dohodnutie typu autentizácie. Pokiaľ žiadateľ príslušný typ autentizácie uvedie v správe Start, tak môže byť tento krok vynechaný.
Specification →		
	← Request	Výmena autentizačných správ. Podľa typu autentizácie môže byť dvojíc Request-Response viac.
Response →		
	← Finish	Oznámenie žiadateľovi o schválení prístupu a o ukončení tranzakcie.

4.2 Formát AVP

AVP (Attribute-Value Pair) je dátová štruktúra, ktorá je zobrazená na Obr. 4.2.

Pole Type (typ) má dĺžku 1 oktet (bajt) a definuje význam dát v poli Value (hodnota). Pole Length (dĺžka) definuje dĺžku poľa Value v oktetoch. Pre typ SAVP



Obr. 4.2: Formát bloku AVP.

1 oktet, pre LAVP a CAVP 2 oktety. Pole Value obsahuje samotné dáta. Jeho dĺžka je menšia ako 256 oktetov pre SAVP, 256^2 pre LAVP a CAVP.

AVP rozdeľujeme na tri typy:

a) SAVP – short (krátke) AVP, pole Value obsahuje jediný typ dát o dĺžke kratšej než 2^8 oktetov. Tieto AVP sú určené na prenos krátkych blokov dát, napríklad adries. Pole Length zaberá 1 oktet. Hodnota Type týchto AVP je v rozsahu 0-127.
b) LAVP – long (dlhé) AVP, obsahuje jediný typ dát o dĺžke kratšej než 2^{16} oktetov. Pole Length zaberá 2 oktety. Tento typ je určený na prenos dlhších blokov dát, napríklad aktív. Hodnota Type týchto AVP je v rozsahu 128-191.
c) CAVP – container (kontajnerové) AVP, obsahuje jedno prípadne viac ACP ľubovoľného typu. CAVP môže obsahovať rôzne SAVP, LAVP a CAVP v ľubovoľnej kombinácii. Celková dĺžka všetkých zoskupených AVP musí byť kratšia než 2^{16} oktetov. Pole Length zaberá 2 oktety. Tento formát slúži k zoskupovaniu AVP podľa rôznych kritérií, napr. rovnaký typ AVP. Hodnota Type týchto AVP je v rozsahu 192-255.

Pokiaľ žiadateľ alebo poskytovateľ prijme správu AVP, ktorú nedokáže spracovať musí tranzakciu zrušiť. Zrušenie tranzakcie sa vykoná vymazaním prevádzkových informácií súvisiacich s danou tranzakciou. Vo všetkých zúčastnených portáloch sa zrušenie tranzakcie vykoná po vypršaní časového limitu. Poskytovateľ môže taktiež tranzakciu ukončiť zaslaním prázdnej správy typu Finish.

4.3 Typy AVP

4.3.1 Mená entít

Nasledujúce AVP (Tab. 4.2) umožňujú prenos mien entít, t.j. osôb alebo zariadení. Portály môžu podľa týchto mien identifikovať žiadateľa, poskytovateľa, autentizátor (t.j. zariadenie schopné autentifikovať žiadateľa) a účtovateľa (t.j. zariadenie, ktorému budú zasielané informácie o prístupe žiadateľa). Formát pre označenie týchto AVP je NAME_AAA_B, kde AAA je skratkou role entity (SUP = Žiadateľ, PRO = Poskytovateľ, AUT = Autentizátor, ACC = Účtovateľ). Znak na pozícii B vyjadruje či je meno lokálne (L) alebo globálne (G). Mená môžu byť napríklad e-mailového typu (entity_name@domain_name) alebo v podobe telefónneho čísla

(+xyz123456789).

Tab. 4.2: AVP s menami entit

Názov AVP	Význam AVP	Typ AVP	Typ Value
NAME_SUP_G	Globálne meno Žiadateľa	0	Text
NAME_PRO_G	Globálne meno Poskytovateľa	1	Text
NAME_AUT_G	Globálne meno Autentizátora	2	Text
NAME_ACC_G	Globálne meno Účtovateľa	3	Text
NAME_SUP_L	Lokálne meno Žiadateľa	4	Text
NAME_PRO_L	Lokálne meno Poskytovateľa	5	Text
NAME_AUT_L	Lokálne meno Autentizátora	6	Text
NAME_ACC_L	Lokálne meno Účtovateľa	7	Text

4.3.2 Adresy zariadení

Adresy zariadení popisujú AVP v Tab. 4.3. Adresa môže predstavovať IPv4 adresu s dĺžkou 4bajty, MAC adresu s dĺžkou 6 bajtov, adresu IPv6 s dĺžkou 16 bajtov alebo číslo TCP/UDP portu s dĺžkou 2 bajty.

Tab. 4.3: AVP s adresami zariadení

Názov AVP	Význam AVP	Typ AVP	Typ Value
ADDR_SUP_G	Globálna adresa Žiadateľa	16	String
ADDR_PRO_G	Globálna adresa Poskytovateľa	17	String
ADDR_AUT_G	Globálna adresa Autentizátora	18	String
ADDR_ACC_G	Globálna adresa Účtovateľa	19	String
ADDR_SUP_L	Lokálna adresa Žiadateľa	20	String
ADDR_PRO_L	Lokálna adresa Poskytovateľa	21	String
ADDR_AUT_L	Lokálna adresa Autentizátora	22	String
ADDR_ACC_L	Lokálna adresa Účtovateľa	23	String

4.3.3 Kódy metód

Nasledujúce AVP v Tab. 4.4 umožňujú prenos kódov autentizačných metód. Globálne sa predpokladá použitie metód založených na protokole EAP, a preto je definovaná SAVP typu EAP. Pre prenos kódu lokálne definovanej autentizačnej metódy je určená SAVP typu LAM. V tomto prípade kódy metód stanovuje správca lokality.

Tab. 4.4: AVP s kódmi metód

Názov AVP	Význam AVP	Typ AVP	Typ Value
EAP	Autentizačná metóda EAP	32	String
LAM	Lokálna autentizačna metóda	33	String

Pre účely autentizácie sú vyhradené AVP v rozsahoch 96-127 (short AVP), 176-191 (long AVP) a 240-255 (container AVP). Lokálna autorita ich môže využívať pre implementáciu vlastných autentizačných metód.

4.3.4 Varianty protokolu

Varianty protokolu ACP môžu byť opäť globálne alebo lokálne. Kód globálnej varianty protokolu je šesťmiestne číslo. Prvá štvorica čísel je číslo RFC, v ktorom je metóda definovaná. Posledná dvojica čísel je poradové číslo varianty v danom RFC. Varianty sa číslujú od hodnoty jedna podľa poradia ich popisu v danom RFC. Kód varianty ACP-VSA sa stanovuje na 000000. Kód lokálnej varianty protokolu ACP stanovuje správca lokality.

AVP v Tab. 4.5 umožňujú obom stranám dohodnúť variantu protokolu ACP.

Tab. 4.5: AVP popisujúce variantu protokolu

Názov AVP	Význam AVP	Typ AVP	Typ Value
GVP	Globálna varianta protokolu	34	Text
LVP	Lokálna varianta protokolu	35	Text

4.3.5 Kódy aktív

Kódy aktív popisuje AVP v Tab. 4.6.

Tab. 4.6: AVP s kódmi aktív

Názov AVP	Význam AVP	Typ AVP	Typ Value
ASSET_G	Globálny kód aktíva	36	String
ASSET_L	Lokálny kód aktíva	37	String

Globálny kód aktív určuje AVP ASSER_G. Hodnota Value obsahuje jeden oktet, ktorý môže nadobúdať hodnoty:

- 0 = implicitné aktívum - poskytovanie tohto aktíva je hlavným účelom Poskytovateľa (napríklad prístup užívateľov do siete v prípade prístupového bodu).
- 1 = autentizácia entity - poskytovateľ tohto aktíva je schopný autentizovať nejakú entitu a výsledok zaslať v správe typu Finish

Základom lokálneho kódovania aktív je hierarchický strom, ktorého každý uzol je možné pomocou jemu príslušnej oktetovej hodnoty rozvetviť do ďalších 255 uzlov. Cesta stromom je určená postupnosťou oktetov. Pokiaľ žiadateľ túto cestu nepozná pomocou správy Offer od poskytovateľa sa dozvie aktíva v aktuálnom uzle stromu. Žiadateľ si následne správou Specification vyberie jednú konkrétnu položku.

4.3.6 Výstupy tranzakcie

V správe Finish je odoslaný výsledok behu tranzakcie. V Tab. 4.7 sú uvedené AVP, ktoré obsahujú výsledok tranzakcie.

Tab. 4.7: AVP obsahujúce výstup tranzakcie

Názov AVP	Význam AVP	Typ AVP	Typ Value
RESULT	Výsledok tranzakcie	38	String
PROVE	Dokazovací faktor Žiadateľa	134	String
VERIF	Verifikačný faktor Žiadateľa	135	String
FORM_START	Obsah správy Start	212	-

Informácia o úspechu alebo neúspechu tranzakcie je prenášaná v jednom oktete AVP RESULT. Hodnota 0 znamená povolenie prístupu, hodnota 2 znamená zamietnutie prístupu a hodnota 1 vyjadruje, že tranzakcia skončila, ale proces riadenia prístupu je ešte v priebehu.

4.3.7 Interperabilita

Nasledujúce AVP uvedené v Tab. 4.8 umožňujú zaistenie interperability s ostatnými autentizačnými protokolmi.

4.3.8 Kryptografická primitíva

Dôvernosť a autentičnosť správ prenášaných protokolom ACP môže byť zaistená externe alebo interne. Externá ochrana spočíva v použití fyzicky zabezpečených spojov alebo šifrovaných spojov (napr. IPSec alebo TLS). Interná ochrana spočíva v autonómnom kryptografickom zabezpečení správ protokolu ACP. Toto zabezpečenie je

Tab. 4.8: AVP pre zaistenie interoperability

Názov AVP	Význam AVP	Typ AVP	Typ Value
MESS_RADIUS	Správa protokolu Radius	128	String
MESS_DIAMETR	Správa protokolu Diameter	129	String
MESS_KERBER	Správa protokolu Kerberos	130	String
AVP_RADIUS	AVP protokolu Radius	65	String
AVP_DIAMETR	AVP protokolu Diameter	131	String

založené na kryptografických primitívach z povinnej sady protokolu TLS. Tieto primitíva sú zobrazené v Tab. 4.9.

Tab. 4.9: AVP s kryptografickými primitívami

Názov AVP	Význam AVP	Typ AVP	Typ Value
INIT	Inicializačný vektor	48	String
PMS	Premaster secret	49	String
CERT	Certifikát	160	String
AES	Kryptogram zašifrovaný šifrou AES	161	String
ENC	Kryptogram s inicializačným vektorom	224	-
HMAC	Autentizačný kód správy	50	String
MAC	Dáta s autentizačným kódom	225	-
RSA	Kryptogram zašifrovaný šifrou RSA	162	String
PSS	Digitálny podpis	163	String
SIG	Dáta s digitálnym podpisom	226	-
CRYPT	Kryptografický kontajner	227	-

4.3.9 Doplnkové AVP

Pokiaľ predpokladáme interakciu s osobou, tak kódy autentizačných metód, aktív a pod. môžeme doplniť ich textovým popisom. Nato je určené SAVP typu TXT, ktoré slúži na textový prepis AVP, ktorý sa v správe nachádza bezprostredne pred ním. Na združenie kódu a jeho popisu slúži CAVP, ktorých názov končí príponou _TX. Tento typ CAVP obsahuje vždy dve AVP. Prvé obsahuje kód a druhé AVP obsahuje textový prepis kódu.

Pre skrátenie správ, ktoré obsahujú viac SAVP rovnakého typu, je definovaný CAVP typu LIST.

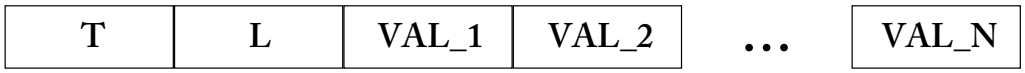
Tab. 4.10: Doplnkové AVP

Názov AVP	Význam AVP	Typ AVP	Typ Value
TXT	Popis predchádzajúceho AVP	64	-
EAP_TX	Kód metódy EAP s popisom	192	-
LAM_TX	Kód metódy LAP s popisom	193	-
ASSET_G_TX	Globálny kód aktíva s popisom	194	-
ASSET_L_TX	Lokálny kód aktíva s popisom	195	-

Tab. 4.11: Blok Value pre CAVP LIST v prípade rovnakej dĺžky SAVP

Názov AVP	Význam AVP	Typ AVP	Typ Value
LIST	Zoznam SAVP rovnakého typu	196	-

Pokiaľ majú združované SAVP rovnakú dĺžku L, tak štruktúra bloku Value tohto CAVP je v prípade N rôznych SAVP rovnakého typu T zobrazená na Obr. 4.3



Obr. 4.3: Blok Value pre CAVP LIST v prípade rovnakej dĺžky SAVP.

Popis blokov:

- T (1 oktet): Blok obsahuje typ združených SAVP.
- L (1 oktet): Blok obsahuje dĺžku hodnôt bloku Value združených SAVP.
- VAL_X (< 2^8 oktet): Blok obsahuje hodnotu bloku Value x-tého SAVP.

5 NÁVRH RIEŠENIA

Príkladom možnosti využitia protokolu ACP je systém drobných elektronických platieb. Podľa [2] tento systém by mal užívateľom umožniť realizáciu malých online platieb, ako je napríklad nákup elektronického lístka na mestskú dopravu, platba za parkovanie, nákup v predajných automatoch, či nápojových automatoch, a podobne.

V navrhovanom systéme predpokladáme existujúci platobný portál (sprostredkovateľ), na ktorý sú naviazané portály predajcov prostredníctvom dočasných alebo trvalých spojov typu TLS. Komunikácia medzi sprostredkovateľom a portálmi predajcov bude týmto spôsobom utajená a autentizovaná. Užívatelia sú vybavení vhodnými komunikačnými zariadeniami (SmartPhone alebo tablet s Androidom). Na komunikačných zariadeniach predpokladáme pripojenie WiFi internetu alebo internetu v mobile od mobilného operátora (pomocou technológie GPRS, EDGE, UMTS, HSDPA, LTE alebo LTE Advanced). Na tieto komunikačné zariadenia bude vytvorená JAVA aplikácia, ktorá sa bude spájať so sprostredkovateľom prostredníctvom protokolu ACP pomocou bezdrôtového pripojenia. Užívateľ dostane pri založení svojho účtu na platobnom portále dva tajné kľúče. Kľúč K_E bude určený na šifrovanie správ medzi sprostredkovateľom a komunikačným zariadením a kľúč K_A bude určený na autentizáciu správ vymieňaných medzi sprostredkovateľom a komunikačným zariadením. Na šifrovanie použijeme symetrickú kryptografiu. Oba kľúče budú nahrané do zariadenia pri stiahnutí aplikácie z webu portálu.

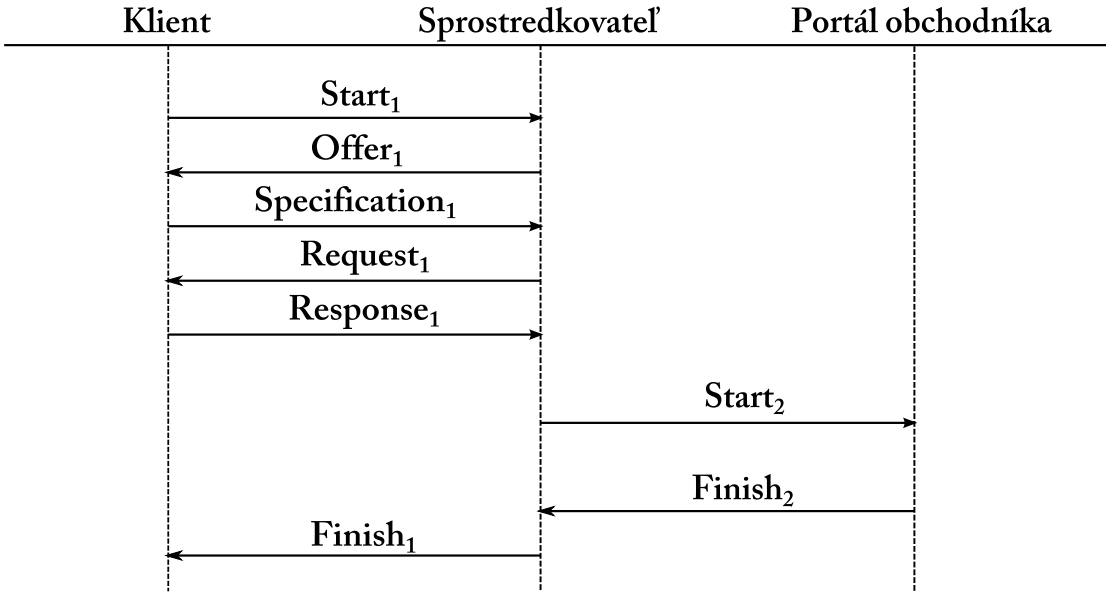
Teraz popíšeme možné riešenie komunikácie medzi klientom a sprostredkovateľom, ktoré je vyobrazené na Obr. 5.1. Predstavme si, že klient si chce kúpiť čokoládovú tyčinku z predajného automatu. Klient sa pripojí na internet pomocou niektorej z vyššie uvedených metód, následne na mobilnom zariadení spustí aplikáciu.

V správe Start_1 aplikácia uvádza AVP, z ktorých môžeme zistiť identifikátor klienta (AVP typu NAME_SUP_L) a identifikátor sprostredkovateľa (AVP typu NAME_PRO_G). Sprostredkovateľ so správy Start zistí, že s ním bola zahájená nová tranzakcia. Podľa identifikátoru klienta zistí potrebné kľúče K_E a K_A. Sprostredkovateľ zašle späť správu Offer_1, v ktorej ponúkne dostupné aktíva (AVP typu ASSET_L_TX). V tejto správe je uvedená možnosť výberu z tovaru v predajnom automate. Klientovi sa v aplikácii zobrazí ponúkaný tovar. Klient si vyberie tovar (čokoládová tyčinka) a pošle sprostredkovateľovi správu Specification_1, v ktorej je vybraný tovar zaznamenaný (AVP typu ASSET_L). Tým je dohodnuté aktívum.

Následne sa musí vykonať autentizácia klienta. Sprostredkovateľ odošle správu Request_1, ktorá pre kontrolu obsahuje vybrané aktívum (ASSET_L_TX) a náhodné číslo (AVP typu INIT). Zreťazenie kódu aktíva a náhodného čísla nazveme výzva R. Klient následne skontroluje, či sa naozaj jedná o čokoládovú tyčinku a vydá príkaz na

preukázanie identity. Aplikácia zašle v správe Response_1 odpoveď $W = \text{f}(R, K_A)$, čo je autentizačný kód (AVP typu HMAC), ktorý je príslušný výzve R. Sprostredkovateľ spraví vlastný výpočet odpovede W' a porovná ju s prijatou odpoveďou. Ak sa $W = W'$, aplikácia pozná autentizačný kľúč K_A majiteľa príslušného účtu. Tým bola prevedená autentizácia klienta, a zároveň bola overená aj jeho voľba.

Sprostredkovateľ následne zaháji tranzakciu s portálom predajných automatov, jeho cieľom je kúpa vybraného aktíva vo vybranej cene a vybranom automate, v našom prípade čokoládovej tyčinky v cene X Kč. Predpokladáme, že medzi sprostredkovateľom a portálom predajných automatov existuje kanál TLS, takže spojenie medzi touto dvojicou je utajené a autentizované. Sprostredkovateľ vyšle správu Start_2, ktorou zahajuje novú tranzakciu. V tejto správe žiada o zaslanie požadovanej čokoládovej tyčinky (ASSET_L) z vybraného automatu (NAME_PRO_L). Portál predajcu v správe Finish_2 pošle potvrdenie, že automat má túto tyčinku vydať (AVP typu PROVE). Tým sa táto tranzakcia medzi sprostredkovateľom a portálom predajných automatov končí. Prevádzkovateľ portálu dôveruje sprostredkovateľovi, a preto predpokladá, že cena za čokoládovú tyčinku (X Kč) bude v najbližšom zúčtovacom termíne sprostredkovateľom prevedená z účtu užívateľa na účet portálu. Sprostredkovateľ zašifruje potvrdenie kľúčom K_E a v správe Finish_1 (AVP typu ENC) ho odošle aplikácii. Tým je zároveň ukončená prvá tranzakcia. Aplikácia potvrdenie kľúčom K_E dešifruje a vygeneruje z neho QR kód. Tento kód z mobilného telefónu načíta predajný automat a vydá požadovanú čokoládovú tyčinku.



Obr. 5.1: Komunikácia pri nákupe.

6 NÁVRH APLIKÁCIE

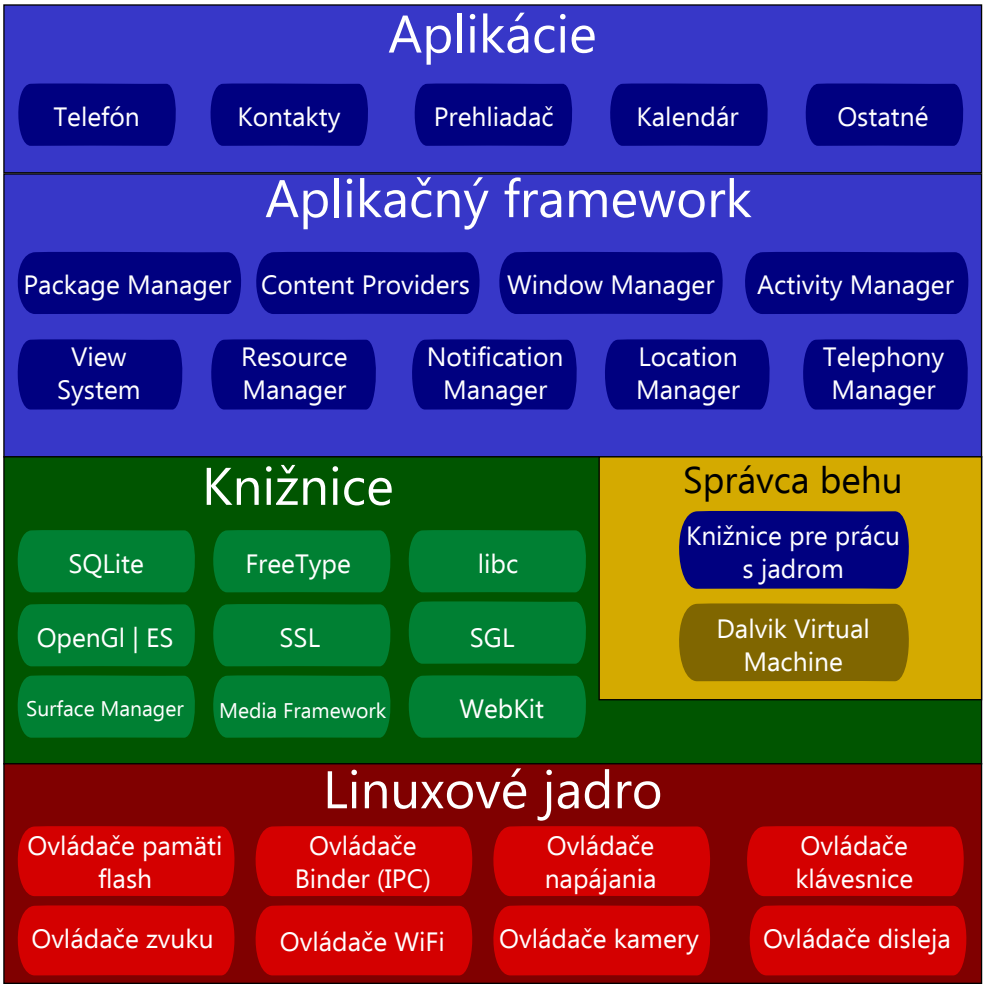
Základ aplikácie je postavený na programe vytvorenom Ing. Petrom Ležákom v jeho diplomovej práci [12]. Keďže Ing. Ležák svoj program vyvíjal v jazyku Java, zaistil tak jednoduchú prenositeľnosť jeho aplikácie na iné platformy, mimo iné aj na platformu OS Android. Android obsahuje takmer všetky balíčky Java SE, avšak chýbajú balíčky AWT a Swing. Pre zvládnutie základov programovania na platformu Android som študoval informácie na webe pre vývojárov od firmy Google [6], ktorý je prehľadne spracovaný a ponúka training. Na začiatok mi veľmi pomohol aj seriál Vyvíjame pro Android [10], kde som videl praktickú ukážku kódu.

6.1 Operačný systém Android

Android [6] je rozsiahla open source platforma, ktorá vznikla pre mobilné zariadenia (SmartPhone, tablety), ale postupne sa rozvíja napr. do náramkových hodiniek, televízii, automobilov alebo okuliarov (napr. GoogleGlass). Operačný systém bol od začiatku vyvíjaný spoločnosťou Android Inc., ktorú v roku 2005 prevzala spoločnosť Google Inc. Následne Google celú platformu aj so zdrojovými kódmi presunul pod združenie firiem Open Handset Alliance, medzi ktoré patrí aj sám Google. V roku 2008 bola vydaná prvá verejne dostupná verzia platformy spolu so všetkými súčasťami a zdrojovými kódmi [7] k dispozícii komukoľvek pod licenciou Apache a GPL v2. Operačný systém je založený na upravenom jadre Linux 2.6 a má viacero verzií (najnovšia verzia 4.2 pod označením Jelly Bean). Operačný systém má monolitické jadro, ktoré sa celé zavádza do pamäti. Systém využíva nástroj Dalvik Virtual Machine, ktorý slúži na vykonávanie bytecodu, na ktorom sú postavené vyššie vrstvy systému. Koncept Dalvik Virtual Machine je postavený na Java Virtual Machine. Kód napísaný v jazyku Java je prevádzaný pomocou nástroja DX pre prevod Java .class do Dalvik .dex súborov. Takto upravené súbory je možné spustiť v Dalvik Virtual Machine. Štruktúru vrstiev operačného systému Android popisuje Obr. 6.1.

6.2 Vývoj a dostupné nástroje

Pre vývoj aplikácii sa primárne využíva jazyk Java. Na oficiálnej stránke pre vývojárov `https://developer.android.com` je množstvo dostupných nástrojov a dokumentácie. Nástroj Android SDK (Software Development Kit) poskytuje vývojárovi API knižnice a nástroje, ktoré sú potrebné na vytvorenie, testovanie a odladovanie aplikácii pre Android OS. Všetky nástroje sú dostupné pre platformy Linux, Windows a Mac.



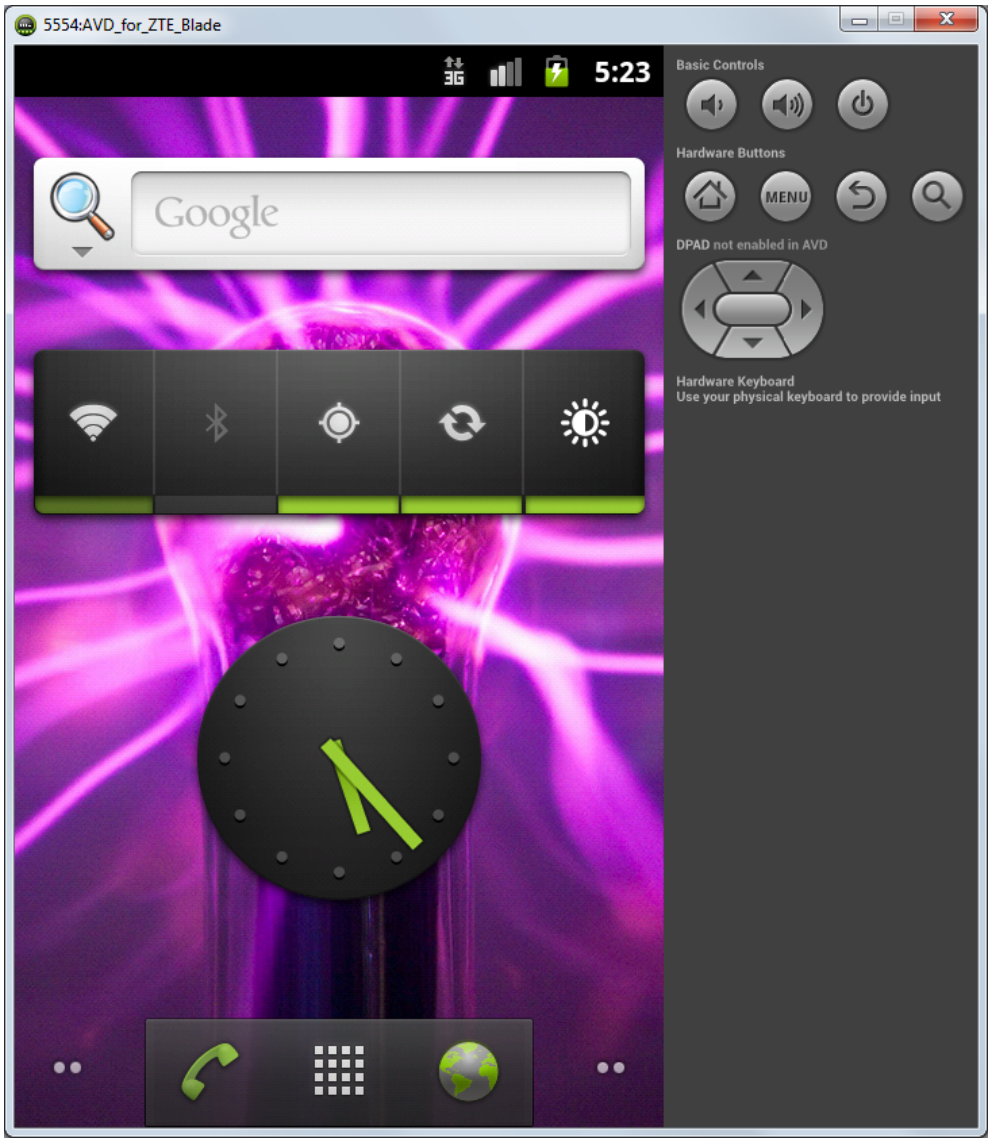
Obr. 6.1: Architektúra platformy Android.

Na vývoj aplikácii je možné využiť aj Android NDK, ktorý taktiež slúži na vývoj Android aplikácii s natívnym kódom (C/C++). Túto možnosť využijú hlavne programátori, ktorí potrebujú, aby procesor spracovával náročné operácie, ktoré nealokujú veľa pamäte (napr. spracovanie signálov, simulácia fyziky a iné).

Okrem SDK a NDK je možné si stiahnuť oficiálny ADT plugin pre Eclipse, ktorý uľahčuje prácu pri vývoji a ladení aplikácie. Eclipse je vďaka tomuto pluginu primárne IDE, v ktorom väčšina vývojárov vytvára svoje aplikácie. Návod na stiahnutie a inštaláciu do Eclipse je umiestnený na `http://developer.android.com/sdk/installing/installing-adt.html`. Pre vývojárov existujú aj ďalšie možnosti. Pre NetBeans bol vytvorený plugin pre vývoj Android aplikácii, dostupný na `https://kenai.com/projects/nbandroid/`. Ku ďalším vývojárskym nástrojom môžeme zaradiť Intellij IDEA, pomocou ktorého je od verzie 10 tiež možné vyvíjať aplikácie pre Android. Viac informácii je možné získať na `http://www.jetbrains.com/idea/features/android.html`.

6.2.1 Android SDK

Android SDK obsahuje, mimo iných nástrojov a knižníc, emulátor Android zariadení. Emulátor je samostatná aplikácia umožňujúca testovanie aplikácii bez hardwarového mobilného zariadenia, ako môžeme vidieť na Obr. 6.2. Pomocou SDK a AVD Managera je možné vytvárať a spúšťať Android Virtual Devices podľa preferovaných nastavení (verzia Android OS, druh CPU, použitie emulovanej kamery resp. webkamery, veľkosť operačnej pamäte, veľkosť vnútornej pamäte atď).



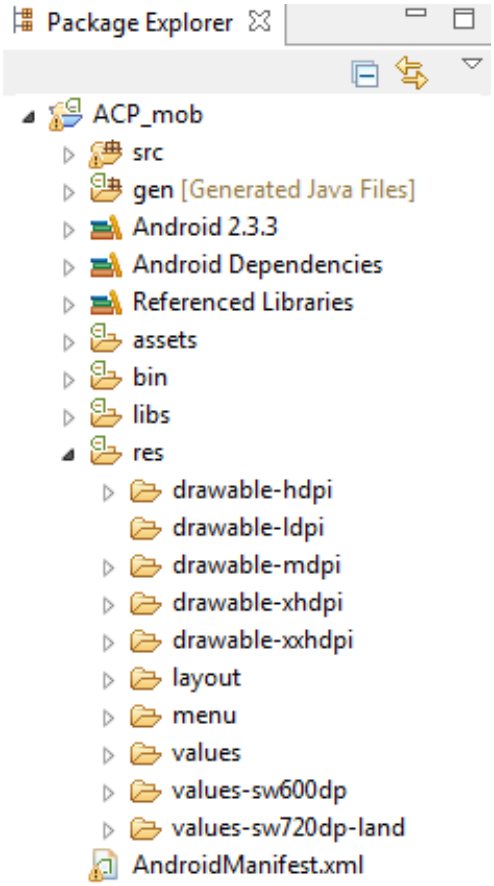
Obr. 6.2: Emulované zariadenie s Android OS.

Na ladenie aplikácii sa využíva program DDMS (Dalvik Debug Monitor Server). Tento nástroj poskytuje služby presmerovania portov, zachytávanie obrazovky na zariadení, sledovanie vláken a pamäti, sledovanie logov, informácie o procesoch

a mnoho ďalších. V prípade dostupnosti hardwarového mobilného zariadenia je možné ladiť aplikáciu priamo na tomto zariadení cez USB kábel.

Preložený kód aplikácie je spolu s ďalšími súbormi zabalený do Android balíka (*.apk), ktorý si užívateľ môže vložiť do svojho zariadenia. Na kompletovanie finálnej aplikácie sa používa nástroj AAPT (Android Asset Packaging Tool), ktorý je súčasťou Android SDK.

Adresárová štruktúra aplikácie je pevne stanovená a je vždy rovnaká, môžeme ju vidieť na Obr. 6.3. Projekt je rozdelený hierarchicky pomocou adresárov. Týmto spôsobom je rozdelený na zdrojové kódy (adresár src), zdroje dát (assets a res) a automaticky generované kódy (adresár gen). Adresár res (resources) sa člení na adresár pre ukladanie obrázkov, XML súborov popisujúcich layout (grafické rozhranie) jednotlivých aktivít, lokalizačných reťazcov textov, apod. Adresár assets nemá špecifikovanú štruktúru a slúži na súbory, ktoré sa nehodia do adresára res.



Obr. 6.3: Adresárová štruktúra projektu.

Dôležitým súborom je AndroidManiefst.xml, ktorý popisuje definované aktivity, služby, broadcast recievers a content providers. Nastavujú sa v ňom tiež prístupové práva k zabezpečeným častiam frameworku.

6.2.2 Tvorba aktivít

Aktivity si môžeme predstaviť ako jednotlivé okná aplikácie, teda každú aktivitu môžeme chápať ako obrazovku, ktorá sa zobrazí na zariadení. Grafické rozhranie týchto aktivít je definované pomocou XML súborov uložených v `/res/leyout/`. Nasledujúci zdrojový kód ukazuje princíp tvorby grafického užívateľského rozhrania v XML súbore:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <Button android:text="Akcia" android:id="@+id/btAkcia"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"></Button>
    <EditText android:text="Start" android:id="@+id/etStart"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"></EditText>
</LinearLayout>
```

Znaky `@+` v atribútoch signalizujú, že nasledujúci text sa bude interpretovať ako identifikátor, pomocou ktorého bude možné na prvok odkazovať z programového kódu. Príklad deklarácie aktivity:

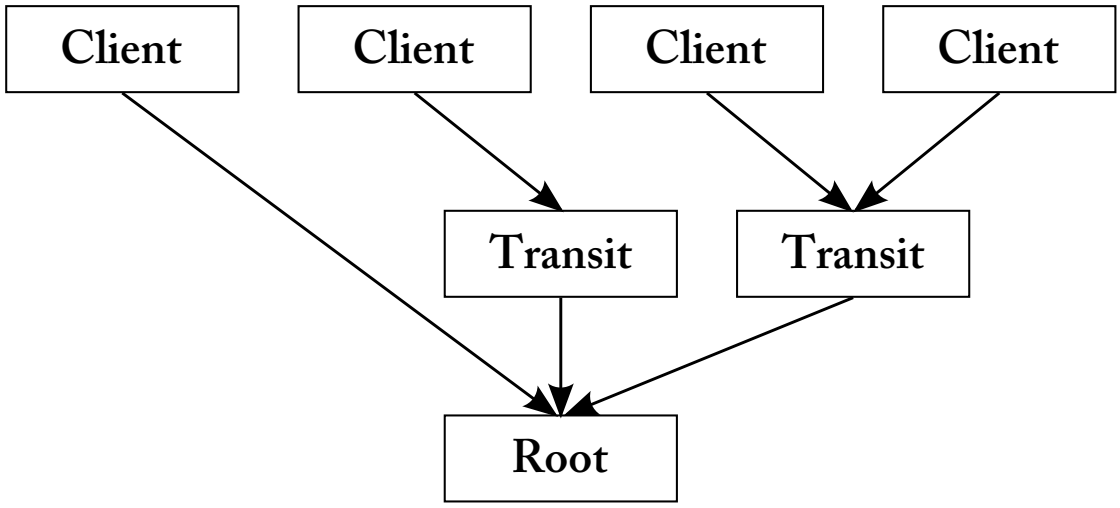
```
...
import android.app.Activity;
public class Aktivita extends Activity{
Button = button;
@Override
public void onCreate(Bundle savedInstanceState){
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
button = (Button) findViewById(R.id.btAkcia);
b1.setOnClickListener(ukaz);
}
View.OnClickListener ukaz = new View.OnClickListener() {
    public void onClick(View v) {
      Toast msg = Toast.makeText(getBaseContext(),
```

```
        "Ahoj", Toast.LENGTH_SHORT);
        msg.show();
    }
...
```

V príklade kódu je trieda Aktivita, ktorá príkazom `extends` rozširuje super triedu jadra Androidu `Activity`. Metóda `onCreate` vytvorí okno a priradí vzhľad pomocou priraďovacej metódy `setContentView` zo súboru main.xml. Tento súbor je prístupný prostredníctvom knižnice zdrojov `R` (Resouce class), ktorá pokrýva všetky dátové zdroje, obrázky atď.

6.3 Hierarchia serveru

Podľa [12] umožňuje protokol ACP vytvárať jednoduché či zložité medzi sebou komunikujúce portály. Implementáciu ACP portálu nazval Ing. Ležák ACP Server. Podľa Obr. 6.4 sa server rozdeľuje ich na tri typy – Root, Transit a Client.



Obr. 6.4: Typy serverov.

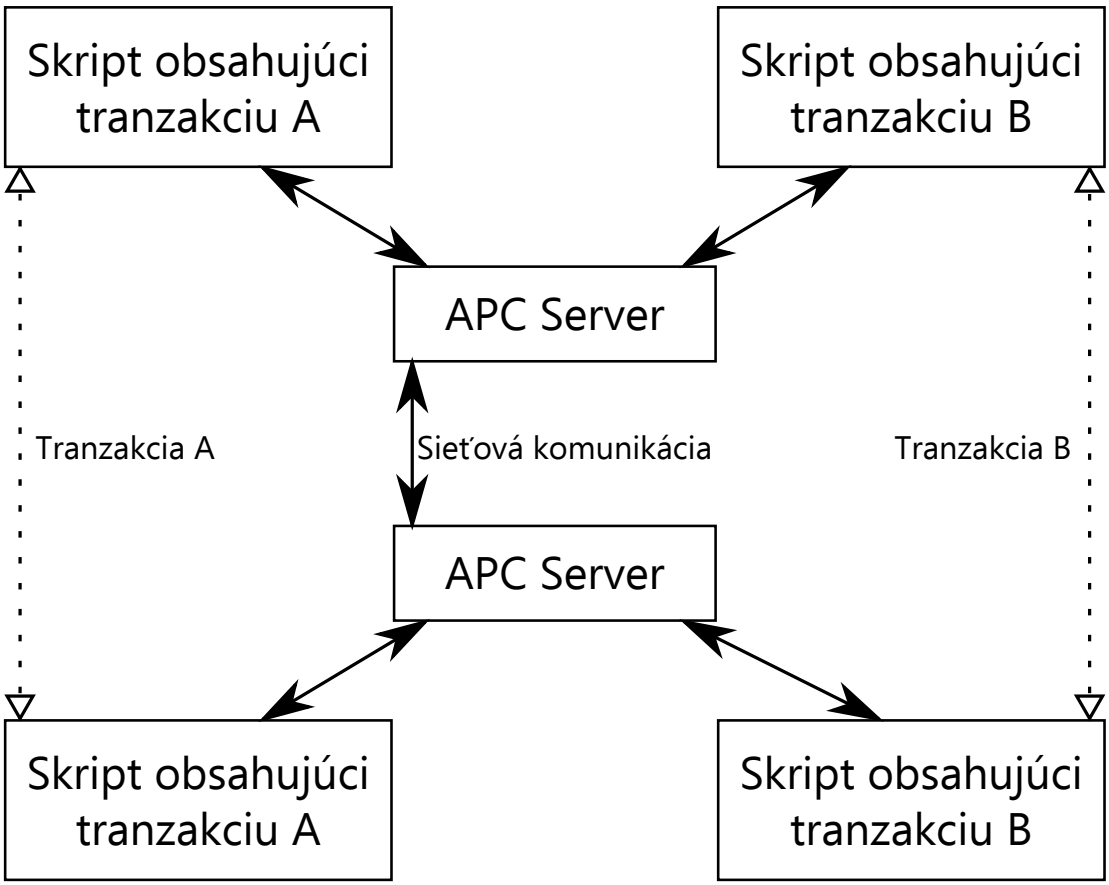
- **Typ Root** – Server postavený najvyššie v hierarchii. Je to kľúčový bod siete a v prípade jeho výpadku spadne celá sieť. Spravuje aktíva, riadi autorizáciu a autentizáciu. Je to koncový uzol pri smerovaní správ typu Start.
- **Typ Transit** – Server sám nie je účastníkom žiadnej tranzakcie, preposiela správy iba medzi inými servermi. Môže vykonávať funkciu brány a prevádzať jeden druh spojenia na iný.
- **Typ Client** – Server Client zahajuje tranzakciu. Užívateľ udáva požiadavky na aktíva a autentizačné údaje.

6.4 Štruktúra aplikácie

Implementácia aplikácie pre platformu Android je realizovaná s využitím Android SDK a pluginu ADT vo vývojovom prostredí Eclipse. Na tvorbu aplikácie je využitý programovací jazyk Java.

V aplikácii sú využité balíčky:

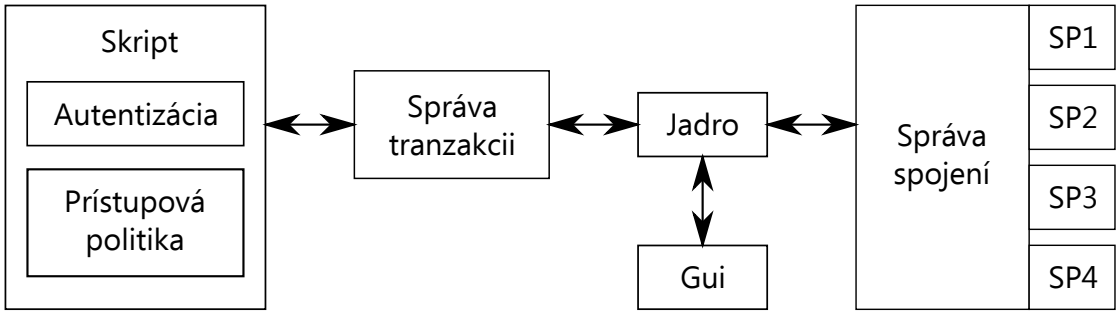
- java.security – obsahuje kryptografické primitíva nutné k autentizácii
- java.net – umožňuje realizáciu sieťovej komunikácie
- javax.net – poskytuje triedy a prepojenia nutné k použitiu SSL protokolu
- javax.xml – slúži na parsrovanie (syntaktickú analýzu) XML dokumentov
- android.os – poskytuje základné operácie systémových služieb, zabezpečuje vnútornu komunikáciu procesov na zariadení
- android.view – spracúvava rozloženie objektov na obrazovke a interakciu s užívateľom
- android.widget – widget balíček obsahuje najmä grafické elementy pre aplikáciu



Obr. 6.5: Posielanie správ medzi skriptami.

ACP server rieši posielanie správ po sieti a riadenie tranzakcií, ale neobsahuje žiadne autentizačné metódy. Tie sú naprogramované v skriptovacom jazyku JavaScript a sú umiestnené mimo ACP server. Server spustí pripravený skript pre každú tranzakciu a posiela medzi nimi správy. Nákres posielania správ medzi skriptami je načrtnutý na Obr. 6.5.

Blokový diagram aplikácie je ukázaný na Obr. 6.6. ACP server pozostáva z modulov s presne definovanými vzťahmi.



Obr. 6.6: Blokový diagram ACP serveru.

6.4.1 Jadro

Jadro serveru zaisťuje koordináciu ostatných modulov. Nie je závislé na ostatných moduloch, avšak je závislé na definovanom rozhraní týchto modulov. Jadro je teda možné v prípade nutnosti nahradiť iným.

6.4.2 Správa tranzakcií

Správa tranzakcií je modul, ktorý sa stará o správu záznamov o tranzakciách. Záznamy obsahujú informácie nutné na smerovanie správ a informácie o stave tranzakcií. Keď je k tranzakcii pripojený spustený skript, tak modul zaisťuje jeho správu a posielanie správ ACP medzi skriptom a jadrom.

6.4.3 Gui

Toto grafické užívateľské rozhranie nie je komplexné. Rozhranie je naprogramované formou aktivít, ktoré boli popísané v 6.2.2.

6.4.4 Správa spojenia

Tento modul slúži na doručovanie správ ostatným ACP serverom v sieti. Zabezpečuje kódovanie a vysielanie na strane vysielača, a zároveň prijatie a dekódovanie na

strane prijímača. Na modul správy spojení sa pripojujú moduly jednotlivých druhov spojení. Správa spojení je preto nezávislá na použitej prenosovej technológii.

6.4.5 Skript

V skripte je naprogramovaný návrh protokolu ACP. Obsahuje modul autentizácie a prístupovej politiky. Skript na strane Root serveru zabezpečuje správu aktív, autentizáciu žiadateľa a autorizáciu prístupu k aktívam. Skript na strane Client serveru simuluje chovanie žiadateľa, a taktiež posiela Root serveru požiadavky na aktíva a autentizačné údaje žiadateľa.

K Android aplikácii je nutné pridať dodatočnú knižnicu Rhnino, aby bolo možné interpretovať engine Rhino. Rhino umožňuje skriptom využívať štandardnú knižnicu jazyka Java. V skriptoch sa využíva balíček `java.security` na implementáciu autentizačných metód.

6.5 Problémy pri vytváraní mobilnej aplikácie

Vytvorenie mobilnej aplikácie v rámci tejto práce pre mňa nebolo úplne jednoduché. Dôvodom je slabá znalosť programovacieho jazyka Java a neskúsenosť s vývojom aplikácií pre platformu Android. Za najväčší problém považujem neexistenciu balíčka `javax.script` pre platformu Android. Tento balíček v JDK 6 obsahuje triedy, ktoré implementujú skriptovací engine. Snažil som sa vymyslieť spôsob, akým sa bude vytvárať skriptovací engine, pretože som chcel zachovať dynamickosť aplikácie. Implementáciou skriptovacieho enginu by sa pri zmenách autentizačných metód nemuselo zasahovať do programu.

Snažil som sa inšpirovať projektom Droidscript, ktorý vytvoril Mikael Kindborg [8] ako experimentálny open-source projekt pod MIT licenciou [15]. V tomto projekte popisuje ako využiť dynamické jazyky (Python, Lua, JavaScript, Ruby, Perl, BeanShell ...) na platforme Android. V tomto projekte sú opísané tri možnosti spúštania skriptov. Prvou metódou je použitie takzvanej Scripting Layer for Android (SL4A) [9]. SL4A slúži ako interaktívny interpreter, slúži aj na úpravu a spúšťanie rôznych skriptov. Druhou je spúštanie skriptov cez triedu WebView, ktorá umožnuje zobraziť webové stránky ako súčasť aktivity. Posledným spôsobom je využitie Mozilla Rhino engine. Napriek vynaloženému úsiliu sa mi nepodarilo úspešne implementovať skriptovací interpreter.

7 ZÁVER

Táto práca sa v úvode venuje súčasným mobilným platobným systémom a popisuje ich využiteľnosť.

V ďalšej kapitole opisuje možnosti zabezpečenia, autentizáciu a kryptografické metódy, ktoré môžu byť použité pri návrhu platobného systému.

Ďalej sú opísané možné útoky na systém, proti ktorým by mal byť systém odolný.

Hlavným cieľom bakalárskej práce bolo naštudovať a popísať protokol ACP (Access Control Protocol), ktorý je určený na riadenie prístupu k aktívam. Následne bol navrhnutý platobný systém pre zariadenia s operačným systémom Android.

Na základe návrhu bola zhotovená aplikácia, pomocou ktorej je možné sa úspešne pripojiť k ACP Servru ktorý vytvoril Ing. Ležák. Aj napriek vynaloženému úsiliu implementácia skriptovacieho intepretéra nebela úspešná a nie je teda možné posielať medzi aplikáciou a serverom správy týmto spôsobom.

Zadanie tejto práce vnímam ako veľmi persektívne, a preto by som sa mu chcel venovať aj vo svojej naväzujúcej diplomovej práci. Chcel by som aj naďalej skúmať problém s implementáciou skriptovacieho interpretéra a rozšíriť svoje znalosti v danej problematike, resp. realizovať posielanie správ medzi aplikácou a serverom iným spôsobom.

LITERATÚRA

[1] BURDA, K., STRASIL, I., PELKA T., STANCIL, P. *Access Control Protocol (ACP).* [Internet Draft]. Internet Engineering Task Force, Fremont, 5. 12. 2011. 25 s. [cit. 12. 12. 2012]. Dostupné z URL: <http://tools.ietf.org/html/draft-kaaps-acp-01>.

[2] BURDA, K., LEŽÁK P. *Aplikace univerzálního rámce řízení přístupu.* Elektrorevue č.37, 2012, Internetový časopis, ISSN 1213-1539.

[3] BURDA, K. *Bezpečnost informačních systémů.* Brno : FEKT Vysokého učení technického v Brne, 1. 11. 2005. 104 s.

[4] BURDA, K., STRASIL, I. *Zabezpečovací systémy.* Brno : FEKT Vysokého učení technického v Brne, 2011, 188 s.

[5] DIERKS, T., Allen, C. *The TLS Protocol Version 1.0* [Internet Draft]. Internet Engineering Task Force, Fremont, 1999. [cit. 12. 12. 2012]. Dostupné z URL: <http://www.ietf.org/rfc/rfc2246.txt>.

[6] Google Inc. *Android Developers.* [Online] 2011. [cit. 20. 05. 2013]. <http://developer.android.com/index.html>.

[7] Google Inc. *Android Source Code.* [Online] [cit. 20. 05. 2013]. <http://source.android.com/>.

[8] KINDBORG, M. *Droidscript* [Online] 2010 [cit. 27. 05. 2013]. <http://droidscript.blogspot.cz/>.

[9] KOHLER, D., *Android-scripting* [cit. 27. 05. 2013]. <http://code.google.com/p/android-scripting/wiki/UserGuide>.

[10] KONEČNÝ M. *Vyvíjíme pro Android.* [Online] [cit. 20. 05. 2013]. <http://www.zdrojak.cz/serialy/vyvijime-pro-android/>.

[11] KRAWCZYK, H., BALLARE, M., CANETTI, R. *HMAC: Keyed-Hashing for Message Authentication* [RFC 2104] Internet Engineering Task Force, Fremont, February. 1997. 11 s. [cit. 12. 12. 2012]. Dostupné z URL: <http://www.ietf.org/rfc/rfc2104.txt>.

[12] LEŽÁK, P. *Testovací implementace protokolu ACP*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2012. 67 s. Vedoucí práce byl doc. Ing. Karel Burda, CSc.

[13] MENEZES, A., OORSCHOT, P., VANSTONE, S. *Handbook of Applied Cryptography* Vyd. 1. Boca Raton: CRC Press, 1997, 780 s. ISBN 08-493-8523-7.

[14] MOPET CZ a. s., *Všeobecné podmínky poskytování Služby Mobito Zákazníkům* [Online] 12. 09. 2012, [cit. 12. 12. 2012].

[15] Open Source Initiative *The MIT License* [Online] [cit. 27. 05. 2013]. Dostupné z URL: <`http://opensource.org/licenses/mit-license.php`>.

[16] SPENCER, W. *Denial of Service(DoS) Attacks.* [Online] 2010 [cit. 12. 12. 2012]. Dostupné z URL: <`http://www.tech-faq.com/denial-of-service-dos-attacks.html`>.

[17] RFC. *Access Control Protocol (ACP).* Brno : VUT Brno, October 26, 2011. 22 s.

ZOZNAM SYMBOLOV, VELIČÍN A SKRATIEK

ACP Access Control Protocol

AVD Android Virtual Device

ČNB Česká národná banka

CPU Central Processing Unit

DoS Denail od Service

DDoS Distributed Denial of Service

DDMS Dalvik Debug Monitor Server

GUI Graphical User Interface

JDK Java Development Kit

MAC Massage Authentication Code

MIT Massachusetts Institute of Technology

MPS Mobilného Platobného Systému

NDK Native Development Kit

SDK Software Development Dit

TLS Transport Layer Security

SL4A Scripting Layer for Android

SSL Secure Sockets Layer

USSD Unstructured Supplementary Service Data

XML Extensible Markup Language