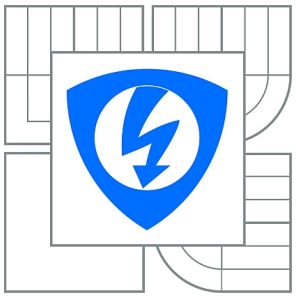
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

BEZPEČNOSTNÍ RIZIKA AUTENTIZAČNÍCH METOD

THE SECURITY RISKS OF AUTHENTICATION METHODS

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

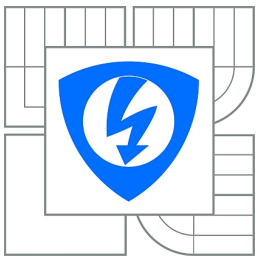
AUTOR PRÁCE — Bc. PETR DZURENDA

AUTHOR

VEDOUCÍ PRÁCE — Ing. MARTIN ROSENBERG

SUPERVISOR

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student: Bc. Petr Dzurenda ***ID:*** 106420
Ročník: 2 ***Akademický rok:*** 2012/2013

NÁZEV TÉMATU:

Bezpečnostní rizika autentizačních metod

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je podrobně rozebrat bezpečnostní rizika současných autentizačních metod. Student se zaměří na popis protokolu ACP a jeho možném využití pro autentizaci přes USB rozhraní. Vyhodnotí bezpečnost daného protokolu a navrhne vhodný autentizační systém. Výstupem práce bude implementace návrhu a tedy vytvoření systému, který autentizuje uživatele v počítači pomocí smart karty s využitím protokolu ACP.

DOPORUČENÁ LITERATURA:

[1] SMITH, Richard. Authentication : From Passwords to Public Keys. [s.l.] : Addison- Wesley Professional, 2001. 576 s. ISBN 0201615991.
[2] TODOROV, Dobromir. Mechanics of User Identification and Authentication : Fundamentals of Identity Management. [s.l.] : Auerbach Publications, 2007. 760 s. ISBN 1420052195.
[3] BOYD, Colin. Protocols for Authentication and Key Establishment . [s.l.] : Springer, 2010. 337 s. ISBN 3642077161.
[4] Burda K., Strasil I., Pelka T., Stancik P.: Access Control Protocol (ACP). IETF, Fremont 2011. Dostupné online:
<http://tools.ietf.org/html/draft-kaaps-acp-01>.

Termín zadání: 11.2.2013 ***Termín odevzdání:*** 29.5.2013

Vedoucí práce: Ing. Martin Rosenberg
Konzultanti diplomové práce:

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

UPOZORNĚNÍ:

Autor diplomové práce nesmí při vytváření diplomové práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

ABSTRAKT

Diplomová práce se zabývá bezpečnostními riziky současných autentizačních metod. Jsou zde popsány metody založené na znalostech uživatele, vlastnictví autentizačního předmětu a biometrické metody ověření. Praktická část diplomové práce se zabývá konkrétním návrhem autentizačního systému, založeného na protokolu ACP, kdy uživatel prokazuje svoji totožnost čipovou kartou u poskytovatele aktiva, kterým je ACP portál na uživatelském počítači.

KLÍČOVÁ SLOVA

Autentizace, bezpečnost autentizace, útoky, Java Card, čipová karta, ACP protokol, testování, přístupový systém

ABSTRACT

Master's thesis deals with the security risks of current authentication methods. There are described methods which are based on user's knowledge and ownership of authentication object and biometric authentication method. The practical part of this Master's thesis deals with a specific design of authentication system based on protocol ACP, when the user proves his identity by smart card on provider assets, which is represented by ACP portal on the user's computer.

KEYWORDS

Authentication, security of authentication, attacks, Java Card, Smart Card, ACP protocol, testing, access system

DZURENDA, P. *Bezpečnostní rizika autentizačních metod*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2013. 67 s. Diplomová práce. Vedoucí práce: Ing. Martin Rosenberg.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma Bezpečnostní rizika autentizačních metod jsem vypracoval 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 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.

V Brně dne ....

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu diplomové práce Ing. Martinu Rosenbergovi za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Koncem 19. a začátkem 20. století se stal přenos dat samozřejmostí, ať už se jedná o přenos informací přes internet, mobilní, satelitní či jiné formy přenosu. S rozvojem informačních systémů pak úzce souvisí i bezpečnost, kdy je potřeba zajistit přístup k aktivům pouze uživateli o jehož aktiva se jedná a ostatním nepovolaným osobám k tomuto přístupu zamezit. Za tímto účelem se používá autentizace. Je to proces, který na základě důkazu předloženého uživatelem povolí přístup k aktivům. Může se příkladně jednat o předložení hesla k přístupu do operačního systému v počítači, či ke vzdálenému přihlášení u internetového bankovnictví apod.

Cílem této diplomové práce je pak rozebrání současných autentizačních metod a rizik, která při jejich použití vznikají. Dále pak navržení a implementace autentizačního systému, který autentizuje uživatele v počítači pomocí smart karty s využitím protokolu ACP. Dle zadání, se práce skládá ze tří hlavních částí.

První část se zabývá rozborem současných autentizačních metod, založených na znalosti, vlastnictví tokenu a vlastnostech žadatele. Jsou zde popsány jejich principy, rizika, možné útoky a demonstrační ukázky bezpečnostních chyb.

Další část práce popisuje protokol ACP, použitý v následně navrženém systému. Jeho účel, strukturu zpráv a způsob komunikace. Popis typů zpráv a jejich účel. Jsou zde rozebrány jednotlivé typy AVP, přenášených ve zprávách protokolu, jejich význam a použití. Kapitola dále popisuje možné propojování více transakcí mezi zúčastněnými portály, čímž vzniká mnohem komplexnější systém řízení přístupu. Následující kapitola je pak věnována problematice technologie Java Card, jelikož je tato technologie v návrhu systému využívána. Kapitola popisuje strukturou zpráv, způsoby komunikace a bezpečnost technologie.

Poslední část práce se zabývá konkrétním návrhem autentizačního systému založeného na protokolu ACP. Je zde rozebrán způsob návrhu appletu instalovaného na kartu, návrh uživatelské aplikace, reprezentující ACP portál poskytovatele. Jsou zde popsány možné scénáře komunikace a bezpečnostní opatření, použitá při návrhu. V závěru je pak celý navržený autentizační systém testován.

1 AUTENTIZACE

Autentizací se rozumí proces, kterým se ověřuje, zda uživatel či entita jsou opravdu tím, za koho se vydávají. Proces autentizace spadá do oblasti tzv. řízení přístupu, který umožňuje uživatelům nebo entitám přístup k informačním zdrojům a naopak se snaží útočníkům k tomuto přístupu zabránit. Za procesem autentizace většinou následuje proces autorizace, který přiřadí uživateli určitá práva, jakým způsobem může s aktivy nakládat.

Způsoby autentizace se liší podle druhu jedinečné informace, na základě které se uživatel prokazuje do následujících kategorií:

1. *"Něco uživatel zná."* Žadatel se prokazuje znalostí. Tento druh autentizace tedy spočívá na principu sdílení určité tajné informace mezi autentizátorem a žadatelem. Jedná se např. o PIN (*Personal Identification Number*), heslo nebo přístupovou frázi.
2. *"Něčím uživatel je."* Žadatel je posuzován podle nějaké měřitelné biologické vlastnosti (biometriky). Jedná se zejména o části těla, jako např. otisk prstu, snímání oční duhovky atd. nebo charakteristiky např. hlas, chůze apod.
3. *"Něco uživatel má*." Žadatel se prokazuje předmětem, označovaném jako autentizační token. Tokenem mohou být platební karty, SIM (S*ubscriber Identity Module*) karty, USB (*Universal Serial Bus*) tokeny a mnohé další.

Každá z výše uvedených metod má své výhody i nevýhody a běžně se pro ověření žadatele využívají. Z hlediska bezpečnosti však tyto výše uvedené tzv. jednofaktorové autentizace nepostačují a je potřeba co nejvíce eliminovat jejich nevýhody. Z tohoto důvodu se tyto metody často navzájem kombinují. V takovém případě se hovoří o vícefaktorové autentizaci. Dojde-li ke skombinování dvou metod, tak se jedná o dvoufaktorovou autentizaci a v případě kombinace všech tří metod jde o třífaktorovou autentizaci. [1]

V současnosti je nejběžnější používána dvoufaktorová autentizace, jejímž nejběžnějším případem je autentizace mobilní stanice v síti GSM (*Global System for Mobile Communications*) či UMTS (*Universal Mobile Telecommunication System*). Každá mobilní stanice má svoji SIM kartu (token) a přístup k ní zajišťuje PIN (tajná informace).

Následující kapitoly se budou zabývat podrobnějším zkoumáním jednotlivých autentizačních metod a jejich bezpečnostními riziky.

2 KLASICKÉ METODY SE ZNALOSTÍ

Tento typ autentizace spadá do kategorie "*Něco uživatel zná*" a jedná se o nejjednodušší a také nejvyužívanější typ autentizace v současnosti. Je to dáno zejména rozvojem internetu a služeb, které se zde vyskytují. Autentizace zde probíhá za pomoci uživatelského jména a hesla.

Princip je takový, že uživatel pošle heslo serveru, který zprostředkovává autentizaci. Server obsahuje databázi všech svých klientů a po obdržení hesla, vyhledá příslušné heslo patřící uživateli a hesla porovná. Tajná informace může být v podobě hesla (password), obsahující alfanumerické znaky s optimální délkou 6-10 znaků, nebo se používá PIN, který obsahuje pouze numerické znaky, avšak bezpečnost je posílena maximálním počtem pokusů a proto bývá obvykle i kratší než heslo (běžně čtyři znaky).

2.1 Bezpečnostní rizika autentizace se znalostí

Rizik, která tato autentizace přináší je celá řada. Jednou z možností je odchycení hesla, pokud je toto heslo přenášeno pomocí nezabezpečené komunikace. Typickými příklady mohou být služby Telnet (*Telecommunication Network*), FTP (*File Transfer Protocol*) nebo HTTP (*Hypertext Transfer Protocol*). Dále jsou to metody založené na neznalosti uživatele tj. příklad, kdy uživatel sám poskytne přihlašovací údaje útočníkovi nebo používá defaultní hasla od výrobce zařízení nebo poskytovatele služeb. [1]

V této kapitole jsou rozebrány metody útoků, kdy se útočník pokouší heslo tzv. prolomit.

2.1.1 Útok hrubou silou

Metoda je známá také pod názvem *Brutus force*. Jedná se o velmi jednoduchou avšak ne příliš efektivní metodu. Princip spočívá v procházení všech možných kombinací hesla a následném jeho poslání spolu s uživatelským jménem do systému, který ověřuje identitu žadatele. Metoda je vhodná spíše pro hesla menšího rozsahu. Výhodou metody je, že vždy dojde k prolomení hesla, avšak ne vždy v reálném čase. V Tab. 1 lze vidět, potřebnou dobu k prolomení hesla v závislosti na délce a složitosti zjišťovaného hesla, za předpokladu, že je k dispozici zařízení schopné provádět milión pokusů o prolomení za vteřinu [2].

Tab. 1 Doba potřebná pro zlomení hesla v závislosti na délce a složitosti hesla

Délka hesla	26 znaků (malá písmena)	36 znaků (malá písmena a číslovky)	52 znaků (malá a velká písmena)	95 znaků (ASCII znaky)
4	0.46 s	1.68 s	7.31 s	1.36 min
5	11.9 s	1.01 min	6.34 min	2.15 hod
6	5.15 min	36.3 min	5.59 hod	8.51 dnů
7	2.23 hod	21.8 hod	11.9 dnů	2.21 roků
8	2.42 dnů	1.07 měsíců	1.70 roků	2.10 století
9	2.09 měsíců	3.22 roků	88.2 roků	20 tisíciletí

2.1.2 Slovníkový útok

Útok, také označován jako *Dictionary attack* je další úspěšnou metodou jak prolomit heslo. Útok je založen na tom, že uživatelé volí jako hesla jednoduchá slova. Nejpoužívanější hesla jsou uložena ve slovníku a jsou testována spolu s uživatelským jménem na autentizačním systému. Slovníky lze stáhnout online a mohou obsahovat data narození, jména, slova česká nebo cizojazyčná. Případně si slovník kdokoli může sám vytvořit.

Z uvedeného vyplývá, že ne vždy je slovníkový útok úspěšný. Záleží na zvolení vhodného slovníku. Příkladem může být počet slov v českém jazyce o délce čtyř písmen malé abecedy. Pokud by se toto heslo lámalo hrubou silou, pak by celkový počet slov byl 42^4=3111696 (česká abeceda má 42 písmen). V případě slovníkového útoku je počet pokusů pouze 2708, což podle *www.oficialni.cz/slova* odpovídá počtu českých slov délky čtyři. Přehled deseti nejpoužívanějších hesel světa lze vidět v Tab. 2. [3]

Tab. 2 Deset nejpoužívanějších hesel světa

Pořadí	Heslo	Pořadí	Heslo
1.	123456	6.	princess
2.	12345	7.	rockyou
3.	123456789	8.	1234567
4.	Password	9.	12345678
5.	iloveyou	10.	abc123

Příkladem aplikací provádějící slovníkový útok jsou např.: *THC Hydra* - umožňuje útoky na FTP, HTTP(S), TELNET a mnoho dalších. Dále *nCrack* - projekt začal roku 2009 a v současnosti zvládá protokoly FTP, SSH (*Secure Shell*), TELNET, HTTP(S).

Při zpracovávání této práce byl testován program *Brutus - AET2*, který podporuje útok na HTTP, FTP, POP3 (*Post Office Protocol*) a TELNET. Na *www.centrum.cz* byla vytvořena emailová schránka *diplmka@centrum.cz* a jako heslo bylo zvoleno 10. nejpoužívanější heslo světa *abc123*. Toto heslo bylo programem prolomeno za pouhých 12s. Doba trvání se ovšem může prodloužit vzhledem k pozici slova ve slovníku, případně se heslo nemusí podařit vůbec nalézt.

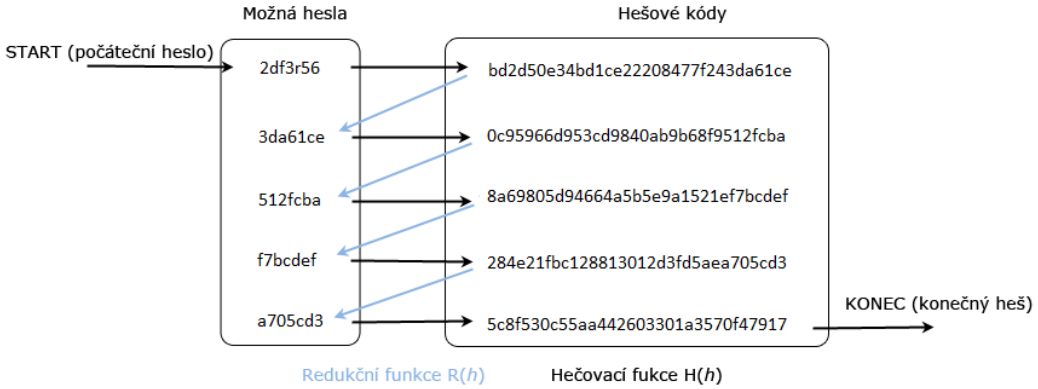
2.1.3 Duhové tabulky

Metoda útoku pomocí *Duhových tabulek*, známá spíše pod anglickým označením *Rainbow tables* je nejmladší a velmi perspektivní metoda pro prolomení hesla. Je založena na kombinaci *Brutus force* a *Dictionary attack.*

V současné době, se pro zabezpečení hesla na straně autentizačního systému používá uložení hesla v podobě jeho otisku, tzv. heše (ang. hash). Z hesla je tedy vybranou hešovací funkcí vytvořen jeho otisk a ten je následně uložen v databázi. Jelikož se jedná o jednosměrnou funkci, tak nelze ze znalosti heše určit původní heslo. Mezi nejznámější hešovací funkce patří SHA-1 (*Secure Hash Algorithm*) a MD5 (*Message-Digest algorithm*).

Úkolem Duhových tabulek je zjistit ze znalosti heše původní heslo. Duhová tabulka je tvořena jednotlivými řetězci skládajících se z hesla a příslušného heše. Je

důležité si uvědomit, že tento heš není výstupem hešovaní tohoto hesla. Princip je takový, že na počátku se vybere libovolné heslo *p* z množiny možných a to se uloží do tabulky. Následně se toto heslo zahešuje příslušnou hešovací funkcí *H(p)*, čímž vznikne heš tohoto hesla *h*. Na tento heš se uplatní určitá redukční funkce *R(h)*, pomocí které se zpětně získá heslo. Důležité je, že redukční funkce není inverzí hešovací funkce, tudíž nevzniká stejný řetězec znaků, co byl na počátku, ale úplně jiná posloupnost znaků. Na takto vzniklé heslo se opět použije hešovací funkce a na vzniklý heš redukční funkce. Tento cyklus se opakuje dokud se nezvolí konec. Poslední vypočtený heš se zapíše do tabulky vedle hesla z kterého se původně vycházelo. Následující řádky tabulky jsou tvořeny dalšími hesly z množiny uvažovaných hesel a konečných hešů. Příklad takové tabulky a odvození heše je zobrazen na Obr. 1. [4]

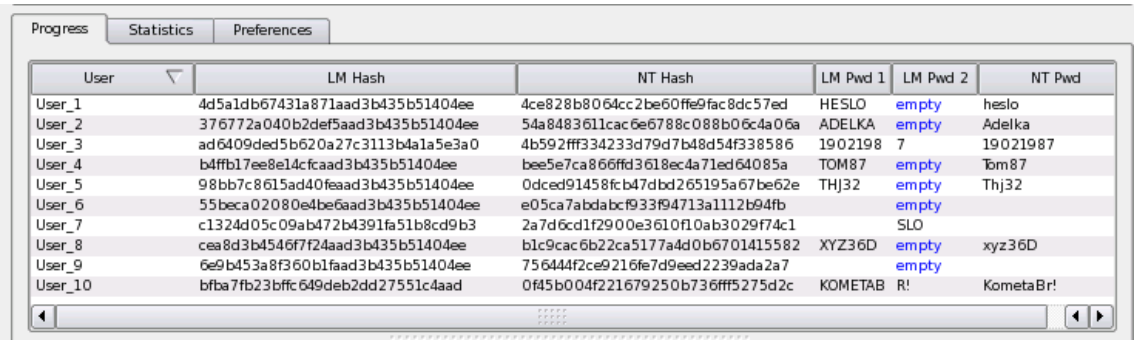


Obr. 1 Vytváření Duhové tabulky

Vyhledávání v tabulce probíhá tím způsobem, že se porovnává známý heš s heši obsaženými v Duhové tabulce. V případě, že nebude nalezena žádná shoda, tak se použije na hledaný heš redukční funkce *R(h)* a na výsledné heslo hešovací funkce. Takto vzniklý heš se opět porovnává s heši v tabulce. Tímto způsobem se pokračuje do té doby, než se nalezne odpovídající heš nebo dokud se nedosáhne konečného počtu cyklů redukce heše. V případě, že byl nalezen stejný heš, tak se vezme heslo odpovídající danému řetězci a provádí se na něm cyklicky funkce hešování a redukování, dokud vypočtený heš nebude identický s původním hešem, který má být rozluštěn.

Velikosti Duhových tabulek se pohybují řádově v GB až TB a lze si je zakoupit, případně volně stáhnout, avšak v menším provedení. Dále je možné využít tzv. *On-line Cracking*, kdy výkonné počítače a obrovská databáze zjistí hledané heslo během okamžiku.

Program využívající duhové tabulky je např. *Ophcrack*, volně stažitelný na *ophcrack.sourceforge.ne*. Za jeho pomoci bylo při demonstrační ukázce prolomeno 7/10 uživatelských hesel u Microsoft Windows Home Edition, běžící na počítači s procesorem AMD Athlon 64 X2 5600+ a operační pamětí 4 GB během 10 min. Nutno podotknout, že se jednalo pouze o duhovou tabulku *XP free small* o velikosti 380MB. Prolomení hesel lze vidět na Obr. 2 a časy jejich prolomení jsou uvedeny v Tab. 3.



Obr. 2 Prolomení uživatelských hesel u Microsoft Windows XP

Tab. 3 Časy prolomení hesel v MS Windows Home Edition

Pořadí	Heslo	Čas prolomení [min]	Pořadí	Heslo	Čas prolomení [min]
1.	heslo	1,28	6.	heslo?	-
2.	Adelka	1,40	7.	Moje heslo	-
3.	19021987	2,38	8.	xyz36D	2,77
4.	Tom87	1,76	9.	x?92KL	-
5.	Thj32	1,70	10.	KometaBr!	2,18

2.2 Eliminace rizik, která tato autentizace přináší

Z výše uvedeného vyplývá, že bezpečnostní rizika spjatá s tímto druhem autentizace není radno podceňovat. Důležitým prvkem autentizace je hlavně uživatel, který by měl zvolit dostatečně silné heslo. Měl by se vyvarovat heslům tvořených z jednoduchých slov jako jsou jména, data narození a jiných běžných slov. Tímto způsobem uživatel znemožní využití Slovníkového útoku.

Dále by heslo mělo být dlouhé alespoň osm znaků a mělo by obsahovat velká i malá písmena, číslice a speciální znaky např.: *k23Qp@L!*. Tato hesla jsou však těžko zapamatovatelná, proto je možno si je vytvořit např. ze známé věty, kde se použijí vždy první písmena použitých slov např.: *Toto je moje 1. diplomová práce!* heslo by poté bylo *Tjm1.dp!*. Dle výzkumu Thomase Baekdala je možné vytvořit heslo pro použití na internetu pomocí jednoduchých frází. Jelikož je většina služeb v rychlosti omezena, ať už síťovou režií nebo maximálním počtem pokusů o přihlášení. Takové heslo by potom mohlo mít podobu *du-bi-du-bi-dub* a k jeho prolomení by došlo dle [5] za 531 855 448 467 let. Dalším požadavkem na zajištění utajení hesla je toto heslo nijak nezveřejňovat a s autentizačním systémem komunikovat po zabezpečeném kanálu.

Na straně autentizačního systémů by mělo být zabráněno útočníkovi, aby se dostal k databázi uživatelů. Hesla by měla být uložená v zakódované podobě, v současnosti jsou hesla hešována. Aby se zabránilo útoku pomocí duhových tabulek mělo by se využívat ještě dalších technik [6] jako je *technikou mnohonásobných iterací* spočívající v cyklickém hešování (výstupní heš je přiveden na vstup např. 1000x) a *technikou zasolení* spočívající v přidání náhodně vygenerovaného řetězce bitů (tzv. salt) ke každému uživatelskému heslu. Různou modifikací hešovacích funkcí je tedy stejnému heslu přiřazen jiný heš.

3 AUTENTIZAČNÍ TOKENY

Tento druh autentizace spadá do oblasti *"Něco uživatel má*." Jedná se o obdobný princip jako je autentizace znalostí, s tím rozdílem, že tajná informace je uložena v nějakém záznamovém zařízení (tokenu), který musí mít uživatel neustále u sebe. Autorita, která token vydala, pak musí tento token zabezpečit proti padělání. Výhodou tokenů je, že si uživatel nemusí pamatovat žádné autentizační údaje, jelikož ty jsou obsaženy v tokenu a navíc jsou delší a více odolnější vůči slovníkovému útoku. Nevýhodou tohoto druhu autentizace je bezesporu možnost ztráty tokenu.

Tokeny se dělí do základních kategorií:

1. *úložiště* - tajná informace je zde pouze uložena, proces autentizace se provádí v jiném zařízení (autentizační procesor). Existují dva druhy úložišť:
   (a) *úložiště s nechráněnými daty* - tajnou informaci si lze za pomoci vhodného čtecího zařízení přečíst, jelikož není nijak chráněná. Typickým příkladem je karta s magnetickým proužkem.
   (b) *úložiště s chráněnými daty* - umožňují vyšší ochranu, jelikož pro přístup k paměti je vyžadováno heslo nebo je tajná informace chráněna kryptografickou šifrou a pro dešifrování je potřeba znát klíč.
2. *autentizátory* - jedná se o zařízení, které obsahuje jednak tajnou informaci, ale také autentizační procesor. Jelikož se provádí proces autentizace přímo v nich, tak se nemá jak dostat tajná informace z autentizačního předmětu. Z tohoto důvodu poskytují autentizátory nejvyšší bezpečnost. Autentizátory lze členit do následujících typů:
   (a) *kontaktní Smart karty* - skládají se z čipu s integrovaným obvodem, plastového podkladu a kontaktní oblasti. Kontaktní oblast obsahuje 6 nebo 8 kontaktů čtvercového nebo oválného tvaru. Příkladem může být platební karta či SIM karta.
   (b) *bezkontaktní Smart karty* - karty obsahují vestavěnou indukční anténu a čip s integrovaným obvodem. Integrovaný obvod se skládá z bezkontaktního radiofrekvenčního rozhrání a mikrořadiče. Bezkontaktní radiofrekvenční rozhrání je připojeno k anténě smart karty a používá střídavé elektromagnetické pole vytvářené snímacím zařízením pro získání napájecí energie pro smart kartu a pro výměnu dat mezi kartou a snímacím zařízením.
   (c) *USB tokeny* - jedná se o miniaturní zařízení, připomínající USB paměť, která slouží k bezpečnému uložení soukromého klíče a certifikátu. Rozhraní je tvořeno dvěma vodiči sloužícími k napájení a dvěma k přenosu dat. [6]

Následující podkapitoly se zabývají bezpečnostními riziky výše uvedených autentizačních předmětů.

3.1 Čipové karty

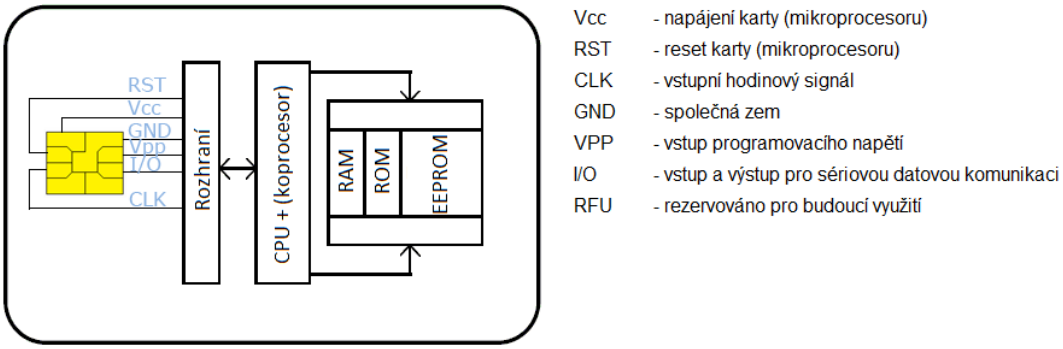
Čipové karty jsou dvojího druhu - kontaktní a bezkontaktní a jedná se o plastové destičky, jejichž velikost udává norma ISO 7816. Dále karta obsahuje čip, přičemž bezpečnost závisí na schopnostech tohoto čipu. Čipové karty se pak dělí na:

1. *paměťové čipové karty* - příkladem je telefonní karta,
2. *paměťové čipové karty se speciální logikou* - například čipové karty pro satelitní, přijímače zpřístupňující placené TV kanály.
3. *procesorové čipové karty* - nejznámější jsou platební karta či SIM karta

Poslední zmíněné se označují jako inteligentní nebo též smart karty. Jedná se o autentizátory, které kromě tajné informace obsahují i autentizační procesor, který zajišťuje autentizaci viz. předchozí kapitola. Komunikace probíhá na principu klient/server. Klient (termínál) pošle žádost na server (smart karta), ten jej vyhodnotí a pošle zpět patřičnou odpověď. [16]

3.1.1 Kontaktní Smart karty

U tohoto druhu karet je zapotřebí fyzického kontaktu s čtecím zařízením. Karty jsou charakteristické kontaktní oblastí o velikosti 1cm^2, stávající se z osmi pozlacených plíšků viz Obr. 3. [17]



Obr. 3 Struktura kontaktní čipové karty a význam jednotlivých pinů, dle normy ISO 7816-2

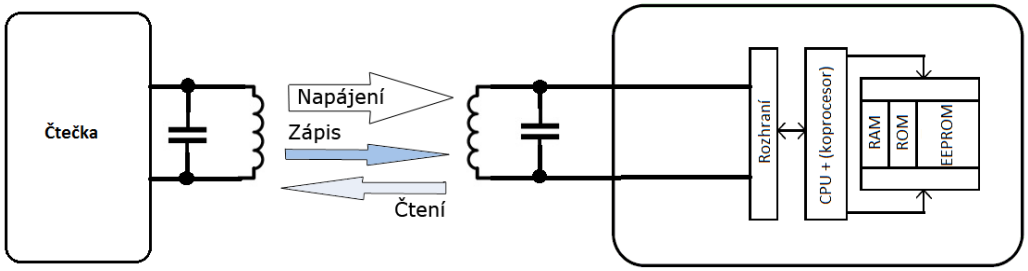
Pod ploškou elektronických kontaktů se nachází mikroprocesor. Běžně jsou použity 8 a 16 bitové mikroprocesory s hodinovým kmitočtem 16MHz. Některé smart karty obsahují navíc i koprocesor. Ten slouží k provádění kryptografických operací, které jsou značně výpočetně náročné, např. pomocí algoritmů RSA (*Rivest, Shamir, Adleman*).

Dále karta obsahuje paměť, která se skládá z pracovní paměti RAM (Read Access Memory), sloužící k ukládání aktuálních hodnot pro výpočty. Po odpojení karty od napájení je veškerý její obsah smazán. Velikost se pohybuje v řádu jednotek kB. ROM (Read Only Memory) je nepřepisovatelná a jsou zde uložena stálá data od výroby. Je zde uložen operační systém karty. Při ztrátě napájení data zůstávají. Velikost je řádově v desítkách kB výjimečně pár stovek kB. EEPROM (Electrically Erasable Programmable Read Only Memory) je elektricky programovatelná a mazatelná. Po odpojení napájení nedochází ke ztrátě dat. Velikost se pohybuje v desítkách kB. [18]

3.1.2 Bezkontaktní Smart karty

Tento druh karet neobsahuje žádnou kontaktní plochu. Komunikace probíhá na principu modulace elektromagnetického pole. Čipy pro RFID (*Radio frequency identification*) využívají různé nosné kmitočty, přičemž nejběžnějším je 13,56MHz. Karta se pro autentizaci pouze k čtecímu zařízení přiblíží, přičemž čtecí vzdálenost se pohybuje okolo 10cm (u platebních karet s procesory) až po 1m (u jednoduchých identifikačních čipů). Bezkontaktní čipové karty jsou standardizovány normou ISO 14443, která udává jejich rozměry, odolnost vůči ultrafialovému a rentgenovému záření, parametry datového přenosu, definici komunikačních protokolů a mnohé další.

Bezkontaktní karty, stejně jako kontaktní karty, neobsahují vlastní zdroj napájení. Napájení probíhá přes indukční vazbu cívky fungující jako anténa karty s čtecím zařízením. Bezkontaktní karty mají jinak stejnou strukturu jako karty kontaktní, tzn. obsahují mikroprocesor, případně koprocesor pro šifrovanou komunikaci a paměť (RAM, ROM, EEPROM). Princip komunikace bezdrátové smart karty s čtecím zařízením je zobrazen na Obr. 4. [7]



Obr. 4 Základní schéma komunikace bezkontaktní smart karty s čtečkou

Bezkontaktní čipové karty mají vyšší životnost a větší spolehlivost než karty kontaktní, avšak problémem je menší poskytnutá bezpečnost, jelikož komunikují v radiovém prostředí a je tedy možné zachytit jejich komunikaci s čtecím zařízením.

3.2 Bezpečnost čipových karet

Bezpečnost čipových karet je obecně na velmi vysoké úrovni. Útoky na čipové karty se rozdělují do tří oblastí:

Fyzické útoky

Je zapotřebí získat kartu a mít k dispozici kvalitní laboratoř, ve které se následně provádí analýza a následná modifikace čipu. K útoku se využívají různá rozpouštědla a leptavé látky, za pomocí kterých se útočník dostává k čipu. Pomocí rastrovacího elektronového mikroskopu je pak možné číst obsah RAM pamětí, pomocí kontrastu napětí, či pomocí sond připojených k čipu odposlechnout či modifikovat komunikaci na vnitřní sběrnici.

Logické útoky

U tohoto druhu útoku je opět zapotřebí mít kartu v držení. Útoky jsou prováděny softwarově a jedná se převážně o objevování softwarových chyb, kvůli kterým jsou pak data dostupná aniž by byl znám PIN. Jedná se zejména o příkazy používané při testování či u jiných aplikací, které nebyly korektně smazány. Případně se může jednat o špatně nastavená přístupová práva souborů.

Postraní útoky

U tohoto typů útoku již není potřeba kartu odcizit. Jedná se o pozorování charakteristik chování čipové karty při jejím používání. Pokud je např. nějaká informace zachycena a je spjata např. s tajným klíčem, je teoreticky možné se k tomuto klíči dopátrat.

Mezi tyto útoky patří např. *časová analýza*, která zkoumá dobu kterou potřebuje kryptografické zařízení ke zpracování zprávy. Útočník pak disponuje množinou zpráv a měří jak dlouho trvá zařízení jejich zpracování. Typ tohoto útoku byl testován na algoritmus RSA, využívající Montgomeriho násobení. [20]

Odběrová analýza využívá toho, že se pozoruje energie spotřebovávaná kryptografickým zařízením a díky tomu je možné zjistit jaké operace se provádějí a na tomto základě odvodit tajný klíč. Odběrová analýza může být *jednoduchá SPA* (Simple power analysis), pomocí které je možné odvodit informace o operacích nebo o klíči, který zařízení používá během výpočtu. Případně se může jednat o *diferenciální DPA* (Differential power analysis), která používá k zjišťování klíče statické metody. DPA může využít metody, kdy zná vstupní nebo šifrovaný text a díky tomu může odhalit šifrovací nebo dešifrovací klíče. U DPA se neměří pouze jedna výkonová křivka jako u SPA ale je potřeba provést více měření a poté ze získaných výkonových křivek pomocí moderních statistických metod, odvodit šifrovací klíče a jiná utajovaná data. Např. je teoreticky možné na základě spotřeby energie odhadnout použitý klíč šifrování DES (Data *Encryption Standard*) [18].

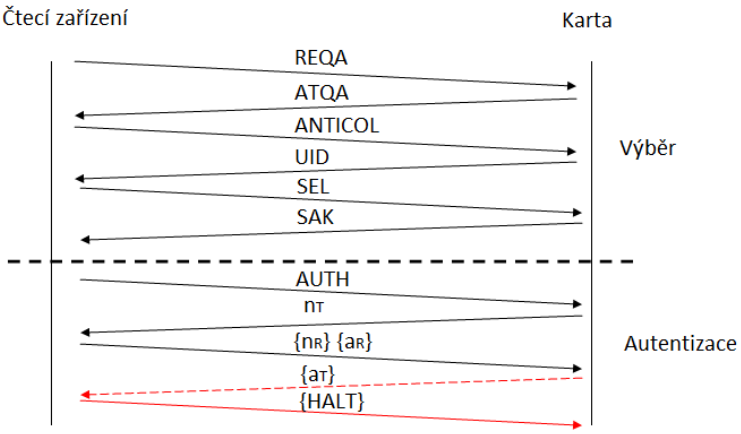
Útoky založené na *zavádění chyb* jsou založeny na tom, že se útočník snaží zavést do průběhu výpočtu určitou chyby, aby se na jejím základě dozvěděl nějaké informace o systému. Vyvolání chyb se může provést např. krátkodobým zvýšením nebo snížením napájecího napětí, změnou taktovací frekvence, použitím extrémních teplot nebo ozařováním. [21]

Následující kapitola ukazuje využití útoku postranním kanálem na bezkontaktní čipovou kartu Mifare, kdy je na základě náhodného čísla N_k poslaného kartou a kryptogramu C_2 možné odvodit tajný klíč.

3.2.1 Útok na Mifare Classic

Bezkontaktní čipové karty založené na technologii Mifare Classic využívají šifrovací algoritmus Crypto1. Technologie *Mifare Classic*, která byla považována za relativně bezpečnou a hojně se využívala. V ČR byly tyto karty použity v projektu městských karet tzv. *OpenCard*. U těchto karet bylo možné jejich kopírování a navyšování stavu elektronické peněženky. Z tohoto důvodu byly od 2.8.2008 vydávány OpenCards na bázi technologie MIFARE DESFire (4K), která používá bezpečnější druh šifrování 3DES.

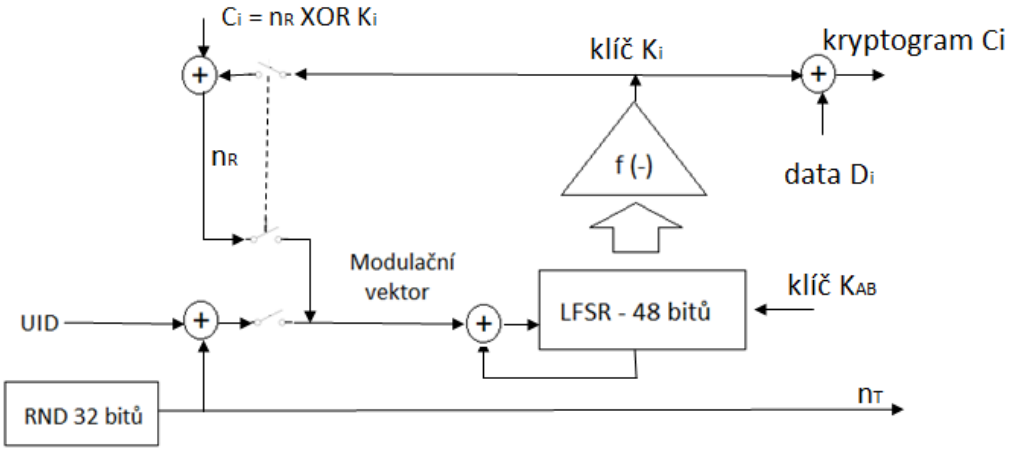
Na Obr. 5 je zobrazen průběh výběru a autentizace čipové karty na bázi technologie Mifare Classic a čtecího zařízení.



Obr. 5 Průběh výběru a autentizace u bezkontaktní čipové karty Mifare Classic

První část je tvořena výběrem karty, od žádosti terminálu *REQA*, odpovědi na výzvu kartou *ATQA* přes antikolizní proceduru *ANTICOL*, *UID* až k provedení výběru *SEL* a jeho potvrzení *SAK*. Výše zmíněný proces výběru je definován standardem ISO 14443. Důležitý je zde proces autentizace, kde dochází k bezpečnostní chybě.

Pro pochopení problematiky autentizace je nezbytné pochopit princip proudové šifry Crypto1. Princip využití šifrování je znázorněn na Obr. 6.



Obr. 6 Proudová šifra Crypto1 na straně karty

Blok **RND** označuje generátor pseudonáhodných čísel. Jedná se o lineární posuvný registr délky 16, se zpětnou vazbou $L(x_0,x_1,\ldots,x_{15}) = x_0+x_2+x_3+x_5$ *(mod 2)*. Inicializační vektor je na kartě pevně nahrán. RND generuje náhodná čísla n_T o délce 32bitů. **LFSR** je lineární posuvný generátor délky 48, se zpětnou vazbou $L(x_0,x_1,\ldots,x_{47})=x_0+x_5+x_9+x_{10}+x_{12}+x_{14}+x_{15}+x_{17}+x_{19}+x_{24}+x_{25}+x_{27}+x_{29}+x_{35}+x_{39}+x_{41}+x_{42}+x_{43}$ *(mod 2)*. Inicializační vektor je dán klíčem pro příslušný blok a je znám oběma stranám. **f(-)** je nelineární filtr, který je definován předpisem $f(x_0,x_1,\ldots,x_{47})=f_c(f_a(x_9,x_{11},x_{13},x_{15}), f_b(x_{17},x_{19},x_{21},x_{23}), f_b(x_{25},x_{27},x_{29},x_{31}), f_a(x_{33},x_{35},x_{37},x_{39}), f_b(x_{41},x_{43},x_{45},x_{47}))$, kde vnitřní funkce ***fa*** a ***fb*** jsou definovány

předpisy $f_a(y_0,y_1,y_2,y_3) = ((y_0 \text{ or } y_1) \text{ xor } (y_0 \text{ and } y_3)) \text{ xor } (y_2 \text{ and } ((y_0 \text{ xor } y_1) \text{ or } y_3))$, $f_b(y_0,y_1,y_2,y_3) = ((y_0 \text{ and } y_1) \text{ or } y_2) \text{ xor } ((y_0 \text{ xor } y_1) \text{ and } (y_2 \text{ or } y_3))$ a vnější funkce ***fc*** je definována předpisem $f_c(y_0,y_1,y_2,y_3,y_4) = (y_0 \text{ or } ((y_1 \text{ or } y_4) \text{ and } (y_3 \text{ xor } y_4))) \text{ xor } ((y_0 \text{ xor } (y_1 \text{ and } y_3)) \text{ and } ((y_2 \text{ xor } y_3) \text{ or } (y_1 \text{ and } y_4))$. [19]

Proces autentizace pak probíhá následovně. Terminál zahájí autentizaci příkazem AUTH, který obsahuje číslo požadovaného bloku, typ klíče a kontrolní součet CRC(*Cyclic Redundancy Check*). Terminál se dále inicializuje klíčem K_{AB} (buď klíč A nebo B, podle toho který zvolil při AUTH) LFSR.

Karta poté odpovídá výzvou n_T délky 4B (32b) vygenerovanou v RND a zároveň si inicializuje svůj LFSR klíčem K_{AB} a modifikačním vektorem $V=n_T$ XOR UID, čímž se změní stav LFSR karty a vytvoří se klíč K_1. Do této části nebyl přenos nijak šifrován. Následující komunikace je již šifrována.

Terminál příjme číslo n_T a modifikuje jím stav svého LFSR. Obě zařízení se nyní nacházejí ve stejném stavu. Nyní terminál pomocí svého RND generátoru vytvoří svoje číslo n_R a na základě n_T odvodí následovníky $n_T´$ a $n_T´´$(RND je inicializován náhodným číslem n_T, a následně se provede 32 posuvů pro získání $n_T´$ a dalších 32 pro získání $n_T´´$). Poté zašle kryptogramy kartě. Kryptogram své výzvy $\{n_R\}=n_R$ XOR K_1 a kryptogram odpovědi na výzvu karty $\{a_R\}=n_T´$ XOR K_2. LFSR terminálu se pak nachází ve stavu S_2.

Karta příjme první kryptogram $\{n_R\}$ a za pomoci klíče K_1 jej dešifruje. Vzniklým číslem n_R modifikuje stav svého LFSR a dostane se do stavu S_2 a zároveň získá klíč K_2, s jehož pomocí dešifruje druhý kryptogram $\{a_R\}$ a získá číslo $n_T´$. Pokud se získané dešifrované číslo shoduje s náhodným číslem RND, pak vygeneruje dalšího následníka $n_T´´$, zašifruje jej a odešle kryptogram odpovědi na výzvu terminálu $\{a_T\}$. V tuto dobu se LFSR nachází ve stavu S_3.

Terminál poté příjme kryptogram $\{a_T\}$ a s pomocí klíče K_3 jej dešifruje a ověří, jestli se shoduje dešifrovaná hodnota s $n_T´´$.

V případě, že vše proběhlo jak má, tak jsou obě strany navzájem autentizovány a registry LFSR na obou stranách jsou ve stavu S_3. [8]

Výše zmíněný proces autentizace vycházel z předpokladu, že se nejedná o korektní proces autentizace. První možností útoku je odposlech komunikace. Jelikož se náhodné číslo n_T posílá v nešifrované podobě, tak s jeho pomocí lze zpětně pomocí XOR odvodit z kryptogramu $\{a_R\}$ klíč K_2. Dále bylo zjištěno, že většina terminálů při neodeslání odpovědi $\{a_T\}$, tzn. dojde k vypršení časového limitu, posílají kryptogram HALT XOR K_3. Binární podoba HALT je známá a lze tedy zpětně z kryptogramu odvodit klíč K_3. Pomocí opakované autentizace, lze pro různá n_T zjišťovat klíče K_2 a K_3 a pomocí zpětného inženýrství odvodit, počáteční stav LFSR a tím tedy klíč K_{AB}. Bylo zjištěno, že postačí $4096=4^{12}$ částečných autentizací pro zjištění klíče K_{AB}. Jelikož je možné získat 5 až 35 částečných aktualizací za sekundu, tak je reálné získat klíč do 14 minut.

Další možný útok je ten, že na základě zjištění klíče K_2 se pomocí inverzní funkce f(-) zjistí množina 65536 možných klíčů K_{AB}. Na základě opakování autentizace a používání jednotlivých klíčů se lze dopátrat k hledanému klíči K_{AB}. [9]

3.3 USB Token

USB token je malé zařízení připomínající USB flash disk, avšak jejich architektura odpovídá již popsaným čipovým kartám. Některé tokény dokonce využívají stejné čipy jako některé smart karty. USB token obsahuje tajné klíče, certifikáty a jiné tajné informace a chrání je proti krádeži. Tajná informace neopouští token. Pokud je například potřeba podepsat email, tak se email posílá do tokenu, kde je podepsán a následně je vrácen. Před zneužitím je většinou chrání PIN.

USB token nepotřebuje speciální čtecí zařízení, jelikož komunikuje přes USB rozhraní, kterým je vybavena většina zařízení. Vtom je jeho výhoda oproti čipovým kartám, které vyžadují speciální čtecí zařízení. Velikost paměti u USB tokenů se standardně pohybuje okolo 8kB, 16kB či 32kB.

Mezi tokeny jsou poměrně velké rozdíly. Jedná se zejména o rozdíly ve funkci, možnostech a také zabezpečení. Nejjednodušší tokeny obsahují pouze paměťovou jednotku. Složitější tokeny obsahují i procesor, který řídí přístup k datům a koprocesor zajišťující šifrování jako tomu bylo u čipových karet. Nejpoužívanější šifrovací mechanizmy jsou RSA a 3DES. Jiné druhy tokenů prostě generují číslo (třeba každou minutu), které se zobrazí na černobílém displeji. Číslo následně uživatel opíše do autentizačního formuláře.

Tokeny jsou také často chráněny proti mechanické manipulaci. Nejčastěji se to provádí slepením krytu, dále mohou mít plošný spoj zalitý epoxidem.

3.3.1 Útok na iKey 2000

USB tokeny obsahují čipy, které jsou často shodné s těmi u čipových karet a je tedy možné aplikovat stejné útoky. Útoky na USB tokeny jsou usnadněny tím, že čip a příslušné obvody jsou chráněny pouze krytem a případně jsou zality epoxidem viz Obr. 7. K desce plošných spojů se pak tedy lze dostat bez velkých obtíží a přitom nejsou patrné známky útoku.

Obr. 7 USB token iKey 2000

IKey 2000 umožňuje tzv. správcovský přístup pomocí MKEY (*Master Key*). Správce potom může iKey konfigurovat a má přístup ke všem soukromím informacím na tokenu. MKEY má délku až 256 ASCII znaků a je uložen na paměti EEPROM. MKEY zde není umístěn přímo ve volném textu, ale nejdříve je hešován hešovací funkcí MD5, přičemž prvních 8 bajtů se tzv. zatemní, pomocí nějakého předdefinovaného algoritmu. Těchto 8 bajtů (zatemněný MKEY) je následně uloženo v paměti EEPROM. Postup vytvoření zatemněného MKEY je zobrazen na Obr. 8.

Obr. 8 Vytvoření zatemněné části MKEY

Přístup k iKey je pak takový, že PC aplikace využívající iKey vytvoří ze zadaného hesla heš a ten posílá do zařízení. IKey však vyžaduje pouze prvních 8 bajtu tohoto heše. Útok je možno realizovat tak, že se do paměti EEPROM pošle známý vhodně zvolený heš a zkoumá se jak byl ovlivněn zatemňovacím algoritmem. Příkladem může být poslání heše MKEY *0000 0000 0000 0000* a výsledný zatemněný heš bude odpovídat *1F01 0F10 1F07 0FF3*. Na základě dalších iterací lze dojít k tomu, že k zatemnění dochází dle následujícího algoritmu:

B1 = A1 XOR 1F
B2 = A2 XOR (A1 + 01)
B3 = A3 XOR 0F
B4 = A4 XOR (A3 + 10)
B5 = A5 XOR 1F
B6 = A6 XOR (A5 + 07)
B7 = A7 XOR 0F
B8 = A8 XOR (A7 + F3)

kde Ai je i-tý bajt heše MKEY a Bi je i-tý bajt zatemněného heše MKEY. Zpětně lze tedy získat heš MKEY, který byl původně D2DD B960 B0D0 F499.

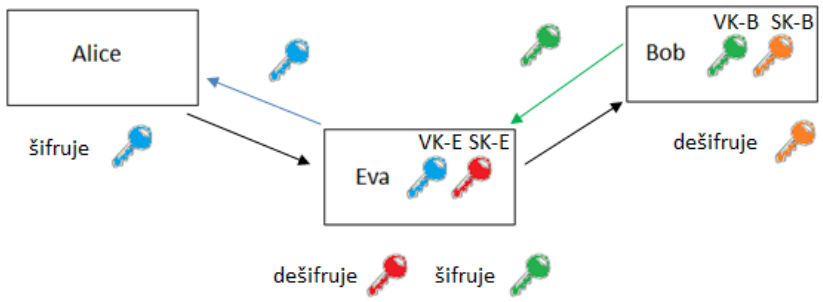
CD = D2 XOR 1F
13 = DD XOR (CD + 01)
B6 = B9 XOR 0F
A6 = 60 XOR (B6 + 10)
AF = B0XOR 1F
66 = D0XOR (AF + 07)
FB = F4XOR 0F
77 = 99 XOR (FB + F3)

Tímto způsobem byl získán potřebný heš MKEY CD13B6A6AF66FB77. MKEY pak odpovídá prvním osmi bajtům heše defaultního hesla rainbow, který má MD5 heš cd13b6a6af66fb774faa589a9d18f906. Tímto způsobem je pak možné se do iKey přihlásit jako administrátor. [10]

4 AUTENTIZACE CERTIFIKÁTEM

V současné době je u nasazovaných autentizačních metod a protokolů pro ověření autentičnosti uložených dat na tokenu využíváno asymetrické kryptografie, která je založena na principu matematické složitosti. Princip je takový, že se generuje dvojice veřejný klíč VK a soukromý klíč SK. Soukromý klíč je uložen na tokenu a jedná se o tajnou informaci. Veřejný klíč je známý všem. Tento systém se pak používá pro elektronické podpisy nebo k vytvoření zabezpečeného kanálu.

Samotná asymetrická kryptografie zajišťuje velmi kvalitní zabezpečení a v případě vytvoření zabezpečeného kanálu je pro útočníka nemožné se k přenášeným datům dostat v reálném čase. Riziko však přináší útok MITM (*Man in the middle*), kdy může dojít k podvržení veřejného klíče VK viz Obr. 9.



Obr. 9 Podvržení veřejného klíče

Na Obr. 9 lze vidět komunikaci, kdy na jedné straně uživatel označován jako Alice žádá o zaslání dat protistraně označované jako Bob přes zabezpečený kanál. Bob tedy vytvoří dvojici klíčů VK-B a SK-B. Soukromý klíč SK-B si uloží do paměti a veřejný klíč VK-B pošle Alici. V cestě mezi nimi se však nachází útočník tzv. Eva. Eva vygeneruje vlastní klíče VK-E a SK-E. VK-E zašle Alici a SK-E spolu s VK-B uloží do paměti. Alice obdrží veřejný klíč v domnění, že se jedná o veřejný klíč Boba a zprávu jemu určenou šifruje klíčem VK-E. Eva tato zašifrovaná data zachytí a dešifruje je svým soukromím klíčem SK-E. Nyní zná obsah dat a případně je může pozměnit. Data poté zašifruje klíčem VK-B a pošle Bobovi. Bob obdrží data a dešifruje je svým SK-B. Bob tedy obdržel data od Evy ale bere je jako data od Alice. [12]

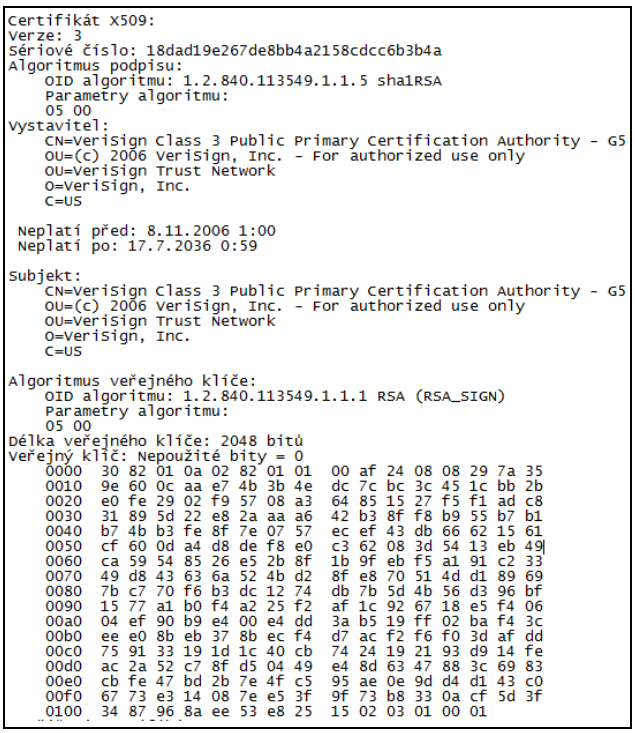
Aby se zabránilo možnému podvržení VK je potřeba nějakým způsobem zajistit, že VK opravdu náleží entitě za kterou se vydává. Tento problém řeší certifikáty.

4.1 Certifikát

Certifikátem se rozumí datová struktura jejímž úkolem je ověření identity vlastníka veřejného klíče. Certifikát lze přirovnat k občanskému průkazu jelikož má obdobnou strukturu, jen s tím rozdílem, že certifikát je popsán v jazyce ASN.1 a občanský průkaz je v tištěné podobě. Certifikát se přenáší mezi počítači pomocí kódování DER (*Distinguished Encoding Rules*), případně PEM (*Privacy Enhanced Mail*) kódování

Base64. Nejběžnější forma certifikátu je definována normou x.509 a v současnosti je nejběžnější certifikát verze 3.

Pro převod certifikátu do čitelné podoby, je možné v příkazovém řádku uživatelského PC zadat příkaz *CertUtil certifikat.cer >> certifikat.txt*. Čitelná struktura certifikátu je zobrazena na Obr. 10.



Obr. 10 Ukázka certifikátu

Z obrázku je patrné, že certifikát obsahuje *Verzi,* která udává podle jaké verze normy x.509 byl certifikát odvozen. *Sériové číslo,* označující certifikát v rámci dané certifikační autority a spolu s vystavitelem tvoří jednoznačný identifikátor certifikátu. *Algoritmus podpisu,* udává algoritmus, kterým byl certifikát podepsán. Zde byla použita hešovací funkce SHA1 a asymetrický algoritmus RSA. *Vystavitel* - specifikuje vystavitele, který certifikát vystavil a ručí za něj. Obsahuje: CN (*Common Name*) - název objektu, OU (*Organization Unit*) - název části firmy, O (*Organization*) - název firmy, C (*Country*) - stát. *Platnost* udává od kdy do kdy certifikát platí. *Subjekt* specifikuje majitele VK, který o certifikát žádal. Skládá se ze stejných částí jako vydavatel. *Algoritmus veřejného klíče*, obsahuje veřejný klíč vlastníka. Skládá se z algoritmu, kterým je klíč tvořen, zde je to RSA a samotného veřejného klíče. [12]

4.2 Certifikační autorita

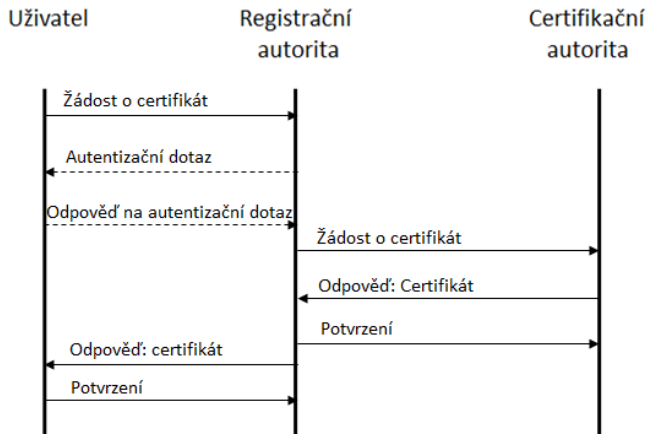
Certifikát jako takový je možné podvrhnout a uskutečnit tak útok MITM. Aby se tomuto útoku zabránilo, je potřeba nějakým způsobem zjistit, zda daný certifikát opravdu patří protistraně. Tento problém řeší certifikační autorita (CA), které všichni uživatelé věří a ona svým podpisem garantuje, že daný certifikát byl vydán právě danému subjektu.

Certifikační autorita zajišťuje podepisování certifikátů svým soukromým klíčem, vydávání certifikátů a odvolání certifikátů. Z uvedeného tedy plyne, že na uchování tajnosti SK certifikační autority stojí celý systém. Pro ochranu SK se používá tzv. HSM (Hardware Security Module), což jsou speciální úložiště pro ukládání kryptografických materiálů (soukromých klíčů CA), které jsou vybaveny ochranou proti vniknutí či neoprávněné manipulaci, kdy dochází k zašifrování či smazání kryptografického obsahu.

Certifikáty tvoří stromovou strukturu, kde nejvýše je postaven kořenový certifikát a nejníže klientský certifikát. [12]

4.3 Registrační autorita

Registrační autorita (RA) zajišťuje ověření žadatele, který zasílá svou žádost o certifikát, která obsahuje VK a identifikaci žadatele. Na základě žádosti si RA ověří zda klient opravdu zná SK, tzn. autentizuje jej. Pokud je autentizace úspěšná, tak teprve poté posílá RA žádost CA, která jí obratem pošle vystavený certifikát. RA potvrdí přijetí a pošle certifikát klientovi, který příjem také potvrdí viz Obr. 11.



Obr. 11 Žádost o certifikát

4.4 Seznam odvolaných certifikátů

Jedná se o seznam všech odvolaných certifikátů CRL (Certificate Revocation List), které daná CA vydala. Seznam obsahuje datum vydání, seznam odvolaných certifikátů a podpis CA. V seznamu jsou uvedeny sériová čísla certifikátů, datum odvolání a důvod jejich odvolání. Odvolané certifikáty jsou uchovávány v CRL do té doby, než vyprší jejich platnost.

Důvodem pro odvolání certifikátu může být prozrazení SK a následná žádost o odvolání certifikátu ze strany uživatele. Pro odvolání certifikátu je možné využít elektronickou cestu, např. emailem, avšak musí být podepsána soukromým klíčem. Některé certifikáty nejsou určeny pro elektronické podpisy a proto některé CA pro

možnost odvolání vydávají jednorázová hesla s jejichž pomocí je možné certifikát odvolat po telefonu či přes webový formulář.

Dalším důvodem odvolání certifikátu může být iniciativa ze strany CA, kdy došlo k porušení smluvních podmínek např. použití certifikátu za účelem na který nebyl vystaven.

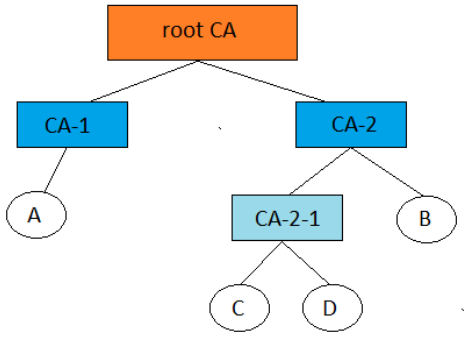
Každá aplikace, která pracuje s certifikáty ověří platnost certifikátu, podpis certifikátu důvěryhodnou CA a také prohledá seznam CRL, zda se zde nenachází daný certifikát.

CRL jsou vydávány danou CA v pravidelných intervalech a jsou převážně zveřejněny na jejích webových stránkách, avšak mohou být určena i jiná místa. V rozšíření certifikátu jsou uvedena distribuční místa odvolaných certifikátů - URL (*Uniform Resource Locator*), na kterých CA zveřejňuje své CRL.

Jelikož jsou CRL vydávány v pravidelných intervalech, může nastat případ, že certifikát byl odvolán, ale nebyl vydán ještě CRL, tzn. účinnost odvolání není okamžitá. Tento problém řeší přímý online dotaz na aktuální stav certifikátu u certifikační autority, která certifikát vydala. Používá se protokol OCSP (*OnLine Certificate Status Protocol*), který využívá spojení klient/server, kdy klient zasílá serveru dotaz obsahující sériové číslo certifikátu a server vrací odpověď zda je daný certifikát odvolán. [12]

4.5 Hierarchie CA

Hierarchická struktura certifikačních autorit je založena na principu přenosu důvěry. Existují CA, které jsou vůči jiným CA nadřazeny nebo jim podřízeny. V hierarchii existuje jedna centrální CA tzv. *kořenová certifikační autorita* (root CA), která vydává své vlastní certifikáty a sama si je i podepisuje. Dále podepisuje certifikáty jí podřízeným certifikačním autoritám. Podřízené CA podepisují certifikáty konečným uživatelům, případně dalším podřízeným CA. Tento princip je patrný z Obr. 12.



Obr. 12 Princip hierarchie CA

Pokud uživatel A ověřuje podpis uživatele C, zjistí, že certifikát podepsala certifikační autorita CA-2-1. V tuto chvíli je tedy certifikát pro uživatele nedůvěryhodný. Z certifikátu CA-2-1 zjistí, že byl podepsán certifikační autoritou CA-2, za kterou se zaručuje root CA, které důvěřuje i uživatel A, jelikož je to nadřízená autorita vydavatele (CA-1) jeho vlastního certifikátu.

Při ověřování certifikátu se tedy musí projít celá cesta od ověřovaného certifikátu, přes certifikační autority, které za tento certifikát ručí až k autoritě, které uživatel sám důvěřuje. Tento princip ověřování důvěryhodnosti se označuje jako *certifikační cesta*. Pokud by neexistovala společná důvěryhodná autorita, je certifikát nedůvěryhodný a záleží na uživateli zda mu bude či nebude důvěřovat. [11]

4.6 Bezpečnostní rizika certifikátů

Certifikáty slouží k účelům zajištění bezpečnosti u asymetrické kryptografie, kdy hrozí útok MITM. Certifikát zaručuje, že VK udávaný protistranou, je důvěryhodný a nejedná se o podvržený klíč.

Jak již bylo uvedeno, celý systém stojí na principu přenosu důvěry a proto musí být certifikát podepsán nějakou důvěryhodnou certifikační autoritou. Pokud tomu tak není, vzniká zde opět možnost útoku MITM, kdy je certifikát podvržen útočníkem.

Dalším i když v současnosti zanedbatelným rizikem je současná asymetrická kryptografie, která je pro certifikáty použita. Ta je založena na matematické složitosti, která zabraňuje útočníkovy v podvržení certifikátu např. jeho modifikace, přičemž by ji neodhalil elektronický podpis. S rostoucím výpočetním výkonem odolnost použitých algoritmů klesá, a proto jsou certifikáty vydávány na určité období. Životnost certifikátu by měla být mnohem kratší, než doba za kterou je možné odhalit SK v době vydání certifikátů.

Nejslabším článkem ověření certifikátu je však samotný uživatel, který software obsluhuje. Pokud samotný uživatel či útočník vloží do úložiště certifikátů falešný kořenový certifikát, tak dojde k úplnému znehodnocení principu přenosu důvěry, jelikož podvržená autorita, které certifikát patří může podepisovat další certifikáty, které se budou tvářit jako důvěryhodné.

5 BIOMETRICKÁ AUTENTIZACE

Tento druh autentizace spadá do oblasti "*Něčím uživatel je*". Identifikace žadatele probíhá na základě porovnání aktuálních naměřených charakteristik žadatele a důvěryhodných dříve změřených charakteristik bezpečně uložených, např. v kontroléru. Podle míry právě změřených charakteristik a zaznamenaných charakteristik je žadateli povolen/odepřen přístup.

Na rozdíl od výše uvedených typů autentizace, kde důkaz identity probíhal na základě tajné informace, byl způsob povolení přístupu na principu *zná/nezná* tajnou informaci, tzn. *je to uživatel/není to uživatel*. U autentizace biometrikou proces probíhá na základě míry pravděpodobnosti, tzn. *pravděpodobně to je uživatel/pravděpodobně to není uživatel*. S tím souvisí parametry *pravděpodobnost chybného odmítnutí* FRR (*False rejection rate*) a *pravděpodobnost chybného povolení* FAR (*False acceptance rate*). Tyto parametry jsou navzájem spjaty a tedy jeden ovlivňuje druhý. Pokud by se nastavil parametr FAR na minimum, tak by se zabránilo povolení přístupu útočníkovi. Na druhé straně by však vzrostl parametr FRR a tím by docházelo k tomu, že i platnému uživateli by nebyl umožněn přístup. Proto je potřeba volit kompromis mezi FAR a FRR. Některé hodnoty FAR a FRR pro jednotlivé typy autentizace jsou uvedeny v Tab. 4. [13]

Tab. 4 Hodnoty FRR a FAR pro různé druhy autentizace

Zkoumaná charakteristika	FRR [%]	FAR [%]
otisk prstu	0,5	0,01
otisk ruky	10	0,05
skenování obličeje	14	0,1
skenování sítnice	10	0,0001
styl chůze	3	3

V současnosti jsou rozlišovány dva typy biometrických metod a to biologické a behaviorální.

5.1 Biologické metody

Tyto metody jsou založeny na identifikaci žadatele podle unikátních fyziologických a anatomických parametrů lidského těla. Mezi nejznámější patří následující metody.

Otisk prstů

Princip spočívá na snímání papilárních linií. Ke snímání se může použít optický, kapacitní či ultrazvukový snímač. V sejmutém otisku se následně porovnávají specifické body, tzv. *markanty*, jako je zakončení linie, rozvětvení linie, výběžky, zkřížení a pod. Optické snímače lze oklamat trojrozměrným modelem prstu, což přináší jisté riziko.

Skenování oka

Mezi dva základní principy patří skenování duhovky a skenování sítnice. Při autentizace podle duhovky se vychází z individuálních vlastností duhovky tj. rozmístění a tvaru skvrn na duhovce. Při skenování sítnice se naopak zkoumá cévní řečiště v oční sítnici. U metody snímání duhovky postačí jako snímací zařízení kamera, zatím co u skenování sítnice je snímací zařízení vybaveno speciální kamerou s laserem, který oční sítnici ozařuje.

Geometrie obličeje

Tato metoda zkoumá specifické rysy obličeje a jako snímací zařízení postačí kamera s dostatečným rozlišením. U obličeje se zkoumá např. rozmístění základních bodů tváře očí, nosu, brady a pod. Dříve se používal sken pouze v 2D prostoru, což přinášelo riziko oklamání systému fotografií. V současnosti se používá sken v 3D, což přináší další rozměr a tím zvyšuje kvalitu autentizace.

Geometrie ruky

Metoda založená na individuálnosti lidské ruky, kdy se zkoumá délka, šířka, hloubka prstů a dlaně. Ke snímání se používají třídimenzionální skener a dále je na snímacím zařízení podložka s pěti kolíky pro zajištění správné polohy ruky pro snímání.

Identifikace na základě struktury žil

Metoda je založena na individuálnosti struktury žil. Ke snímání se využívá speciální kamera, která snímá v infračervené oblasti elektromagnetického spektra (760nm). U snímků se poté zkoumá tvar a tloušťka žil. Metoda je velice přesná, jelikož struktura žil je jedinečná a navíc některé systéme dokáží rozpoznat průtok krve.

Dále existují metody ne tak časté a známe, ke kterým patří *identifikace na základě DNA* - zatím však neexistuje přístroj, který by dokázal porovnávat DNA v reálném čase, *charakteristika ušního boltce* - používá se skenování v 2D či 3D, otisk ušního boltce či rozložení tělesné teploty na ušním boltci, *podle plantogramu* - který zkoumá otisk bosé nohy zatížený vlastní vahou. [14], [15]

5.2 Behaviorální metody

Behaviorální metody jsou založeny na sledování individuálních charakteristik chování člověka. Mezi nejznámější metody patří:

Identifikace podle charakteristiky hlasu

Metoda je založena na zkoumání charakteristik hlasu řečníka, kdy zkoumá kmitočtové spektrum či rytmus mluvení. Metoda je náchylná na útok, kdy se vytvoří záznam mluvení řečníka a ten se následně přehraje autentizačnímu zařízení.

Identifikace podle charakteru chůze

Metoda je založena na základě porovnání drah křivek, které opisují určité body na lidském těle. K analýze se využívá filmového záznamu, který poté vyhodnocuje dráhy křivek.

Dále existují metody, které využívají individuálnosti podpisu, dynamiky psaní na klávesnici, dynamiky pohybu myší, způsobu pohybu očí či tvaru a pohybu rtů.[14], [15]

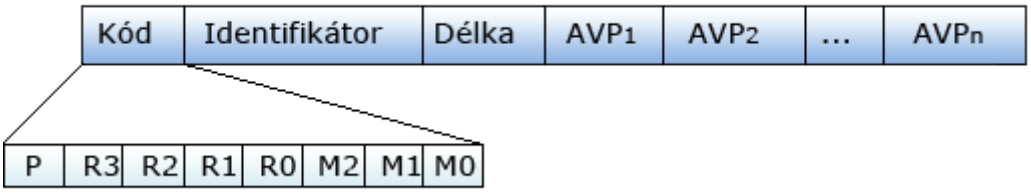
6 PROTOKOL ACP

Protokol ACP (Access Control Protocol) je vyvíjen Vysokým učením technickým v Brně a slouží k jednotné implementaci různých metod řízení přístupu k aktivům síťových zařízení. Protokol umožňuje sjednat požadovaná aktiva, sjednat autentizační metodu, schválit přístup a provádět účtování. Tato kapitola se zaměří právě na popis tohoto protokolu, přičemž bylo čerpáno hlavně ze zdrojů [22], [23] a [24].

Jelikož jsou současné systémy řízení přístupu založené na protokolech, jako je např. KERBEROS, RADIUS (Remote Authentication Dial In User Service) apod., které jsou velmi různorodé a nejsou schopny vzájemné spolupráce, jeví se protokol ACP jako velmi perspektivní. Protokol je založen na myšlence, že všechna síťová zařízení budou obsahovat tzv. *AC portály* (Access Control porátly), které budou používat ACP protokol k řízení přístupu k vlastním aktivům, nebo vyjednávat přístup k aktivům jiných zařízení. Protokol ACP je dvoustranným protokolem, kde na jedné straně je Portál žádajícího zařízení, který je označován jako *Žadatel* a na druhé straně je portál poskytující žádaná aktiva, označován jako *Poskytovatel*. Proces sjednání přístupu k jednomu konkrétnímu aktivu je označován jako *transakce*. Vzájemná komunikace mezi žadatelem a poskytovatelem může probíhat přímo a nebo může být zprostředkována prostřednictvím jiných portálů. Další výhodou je, že protokol ACP je schopen spolupráce s výše jmenovanými protokoly.

6.1 Zprávy ACP

Pro přenos dat mezi portály v rámci transakce, se používají zprávy ACP, přičemž mohou být tyto zprávy přenášeny komunikačními protokoly z různých vrstev ISO modelu (linková vrstva až aplikační vrstva). Formát zprávy ACP je podobný zprávě EAP (Extensible Authentication Protocol). Skládá se z hlavičky a těla, které je složeno z jednoho a více AVP (Attribute Value Pair). AVP blok také nemusí být ve zprávě obsažen vůbec. V takovém případě se jedná o prázdnou zprávu. Struktura zprávy je zobrazena na Obr. 13.



Obr. 13 Struktura zprávy ACP

Význam jednotlivých polí je následující:

- *Kód* - má velikost 1 bajt, a označuje typ zprávy a zároveň odlišuje protokol ACP od protokolu EAP. První bit označen jako P, slouží právě k odlišení protokolů. Pro protokol ACP je hodnota 1. Bity R3,R2,R1 a R0 jsou rezervní bity a musí být nastaveny na 0. Bity M2, M1 a M0 slouží k označení typu zprávy, přičemž

poslední bit M0 zároveň určuje zda byla zpráva poslána žadatelem nebo poskytovatelem.

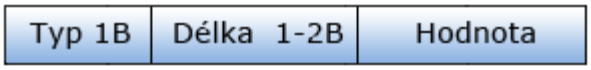
- *Identifikátor* - má velikost 3 bajty, a slouží k identifikaci zprávy jedné transakce od správ jiných transakcí. Hodnotu v poli identifikátoru určuje vždy žadatel a to nezávisle na protějším terminálu. Jelikož však může nastat případ, že se vyskytne transakce v opačném směru, kde bude žadatelem protější portál, a ten zvolí stejný identifikátor, dochází k jednoznačnému odlišení transakcí pomocí bitu M0.
- *Délka* - má velikost 3 bajty, a udává celkovou velikost zprávy včetně hlavičky v bajtech.
- *AVP* - jedná se o pole, která přenáší samotná data protokolu ACP. Velikost je volitelná avšak nesmí překročit velikost 64kB. Více o AVP v kapitole 6.2.

Protokol ACP definuje následující typy zpráv, pomocí kterých je uskutečněna transakce:

- *START* - zasílána vždy žadatelem, označuje zahájení transakce. V poli Kód, je tato zpráva indikována hodnotami bitů M2=0, M1=0 a M0=0. V této zprávě mohou být obsaženy informace jako identifikátor žadatele, identifikátor autentizátoru schopného autentizovat žadatele, kód požadovaného aktiva, typ autentizace a pod.
- *OFFER* - poskytovatel v této zprávě nabízí žadateli dostupná aktiva a typy autentizačních metod, případně se na nějaké parametry může dotázat. Zpráva je indikována hodnotami bitů M2=0, M1=0 a M0=1.
- *SPECIFICATION* - v této zprávě si žadatel vybírá aktivum nebo autentizační metodu z nabízených ve zprávě OFFER. Zpráva je indikována hodnotami bitů M2=1, M1=1 a M0=0.
- *REQUEST* - zprávou poskytovatel posílá žadateli data potřebná k autentizaci. Zpráva je indikována hodnotami bitů M2=1, M1=0 a M0=1.
- *RESPONSE* - zprávou žadatel zasílá data k provedení autentizace (autentizuje se). Zpráva je indikována hodnotami bitů M2=0, M1=1 a M0=0.
- *FINISH* - zpráva slouží k ukončení transakce a je poslána vždy poskytovatelem. Obsahem zprávy může být povolení/zamítnutí přístupu, požadované aktivum, autentizační prvek pro následující autentizaci a pod. Zpráva je indikována hodnotami bitů M2=1, M1=1 a M0=1.

6.2 Struktura AVP

Pole AVP (Attribute-Value Pair) přenáší samotná data protokolu ACP a jeho struktura je zobrazena na Obr. 14.



Obr. 14 Struktura pole AVP

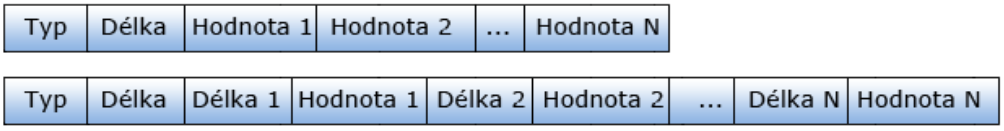
Význam jednotlivých polí je následující:

- *Typ* - má velikost 1 bajt a definuje význam dat v poli Hodnota.
- *Délka* - má délku 1 nebo 2 bajty a udává velikost dat v poli Hodnota v bajtech.
- *Hodnota* - má velikost maximálně 256B (256^1) nebo 64k (256^2) a nese samotná data.

6.2.1 Formáty AVP

U protokolu ACP jsou definovány tři formáty AVP. Prvním je krátké SAVP (Short AVP) sloužící pro přenos krátkých zpráv u zařízení s omezeným výpočetním výkonem. Obsahuje jediný typ dat s maximální velikostí 256B (28B). Hodnota v poli Typ je pro SAVP v rozsahu 0-127. Dalším typem jsou dlouhá LAVP (Long AVP). Tyto AVP stejně jako SAVP obsahují pouze jediný blok dat, avšak velikost tohoto bloku je až 64kB (216B). Hodnota v poli Typ pro LAVP je v rozsahu 128-191. Posledním typem jsou kontejnerové CAVP (Container AVP), která mohou obsahovat více typů AVP, avšak jejich celková délka je maximálně 64KB (216B). Hodnota v poli Typ pro CAVP je v rozsahu 192-255.

V případě, že zpráva obsahuje více SAVP stejného typu, je definován CAVP typu LIST, který umožňuje velikost zprávy zmenšit. Přičemž existují dvě varianty struktury CAVP. Pro sdružená SAVP se stejnou délkou a s různou délkou. Rozdílem je obsah pole Délka, které v případě stejných délek SAVP udává celkovou délku sdružených SAVP. V případě různých délek má pole Délka hodnotu 0 a před každým polem Hodnota, je uvedena délka SAVP v poli DélkaN, kde N udává pořadí SAVP v CAVP. Hodnota v poli Typ pro CAVP typu LIST je 196, a struktura tohoto CAVP je zobrazena na Obr. 15.



Obr. 15 Struktura CAVP typu LIST, pro a) SAVP stejné délky b) SAVP různé délky

6.2.2 Typy AVP

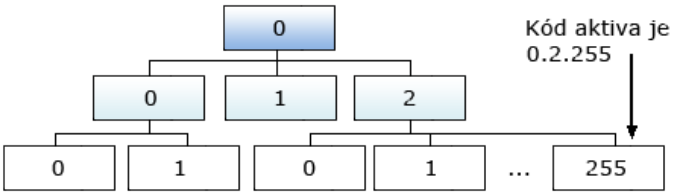
Protokol ACP definuje také několik typů AVP, podle typu obsahu, který přenášejí. Může se jednat o názvy entit pro jejich snazší identifikaci a tedy určení jejich role v systému. Jména mohou mít tvar např. emailové adresy.

Dále mohou být přenášeny jednotlivé *adresy zařízení*. Typ adresy je pak dán její délkou. Platí, že pro délku 16B se jedná o adresu IPv6, pro 6B o MAC adresu, pro 4B o adresu IPv4 a pro 2B o číslo portu TCP/UDP. Adresa je vždy zadána číselně.

AVP může také obsahovat *kódy autentizačních metod*. AVP může být typu EAP, vtom případě pak jsou vněm obsaženy kódy metody protokolu EAP, nebo může být typu LAM (Local authentication method) a kódy metod jsou stanoveny lokálním správcem.

Některé typy AVP mohou také sloužit, k tomu aby mohly dvě komunikující strany *sjednat variantu protokolu* ACP tj. konkrétní způsob výměny zpráv protokolu ACP se stanovením obsahu těchto zpráv. Pro kód globální varianty platí, že je tvořeno šesti čísly 1234XY, přičemž první část 1234 udává číslo RFC (Request for Comments), ve kterém je metoda definována a druhá část XY udává pořadové číslo varianty v tomto RFC. Pro každou variantu jsou rezervovány typy AVP v rozsahu 96-127 (SAVP), 176-191 (LAVP) a 240-255 (CAVP). Význam jednotlivých AVP se pak odvíjí právě od zvolené varianty protokolu ACP.

Jednotlivá AVP umožňují přenášet *kódy aktiv*. V doporučení globálního kódování aktiv se udává, že velikost pole Hodnota je 1B a má hodnotu 0 pro implicitní aktivum nebo 1 pro autentizaci entity. U lokálního kódování je základem hierarchický strom, jenž může obsahovat až 256 větví, přičemž každá větev může obsahovat dalších 256 dalších větví. K určitému aktivu se pak přistupuje pomocí řetězce oktetů, který je identifikuje viz. Obr. 16.



Obr. 16 Hierarchický strom

Kódy aktiv jsou zasílány poskytovatelem ve zprávě OFFER a žadatel si z nich vybírá jeden ve zprávě SPECIFICATION, který může být konkrétní aktivum a nebo skupina aktiv na hierarchicky nižší úrovni. V tomto případě by poskytovatel zaslal další zprávu OFFER s kódy aktiv z této úrovně. Výběr aktiva tedy probíhá na základě sekvencí výměn zpráv OFFER a SPECIFICATION.

U zprávy FINISH se objevuje typ AVP obsahující *výstup transakce*, který sděluje žadateli, že přístup k aktivu byl povolen/zamítnut, případně uvádí samotná aktiva. Aktivy mohou být jména, kódy nebo autentizační faktor pro přístup k dalším aktivům např. v případě, následující autentizace u jiného portálu, který vlastní žádaná aktiva. V takovém případě mohou být obsahem zprávy také adresy zařízení, dokazovací faktor žadatele (k dokázání své identity), verifikační faktor (k ověření identity protistrany) popřípadě formát zprávy START. Pro přenos autentizačních parametrů jsou určena LAVP a pro specifický formát zprávy START CAVP.

Protokol ACP je také schopen spolupráce se systémy založenými na protokolech RADIUS, DIAMETER a KERBEROS (je tzv. *interoperabilní*). A to díky typům AVP z rozsahu 128-131, které obsahují celé zprávy těchto systémů nebo jejich AVP.

AVP mohou také obsahovat *kryptografická primitiva* zajišťující bezpečnost přenášených zpráv.

V případě, že je potřeba zaslat také popis kódu, používají se tzv. *doplňková AVP*, které přenáší text popisující AVP kódu nacházející se bezprostředně před ním.

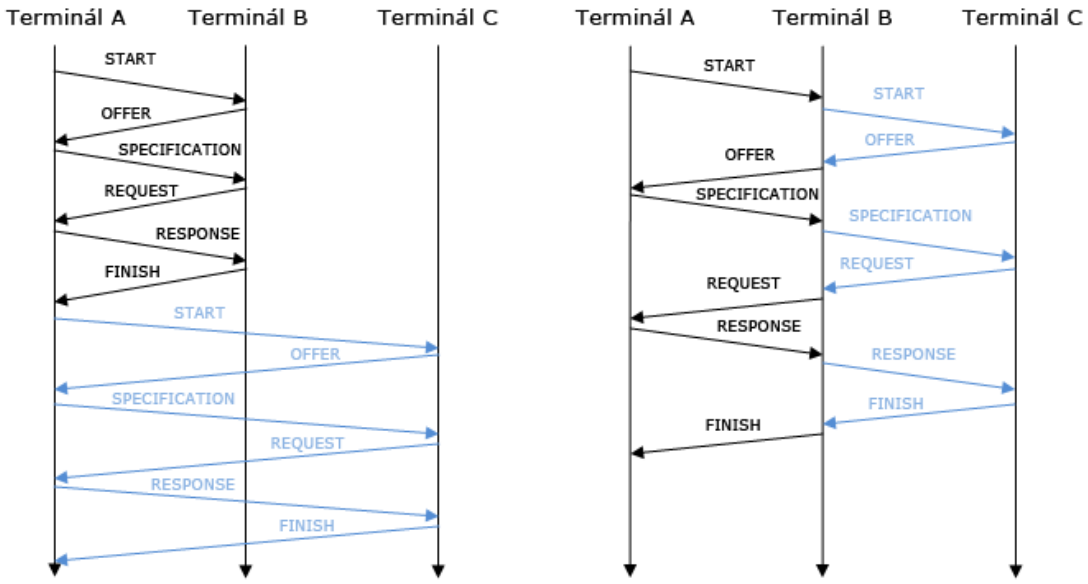
6.3 Transakce u ACP

U protokolu ACP platí, že komunikaci řídí vždy Poskytovatel. Každá transakce je zahájena zprávou START vyslanou žadatelem a ukončena zprávou FINISH zaslanou Poskytovatelem. Základní způsob komunikace je komunikace pomocí krátkých odpovědí, která je označována jako ACP-VSA (Variant of Single Answers), kdy žadatel ve svých odpovědích posílá pouze jediné AVP.

Výhodou protokolu ACP je bezesporu zřetězování a vkládání dalších transakcí, čímž je možné do systému zapojit další zařízení.

V případě *zřetězení transakcí* se jedná o případ, kdy výstupem jedné transakce je získání aktiva potřebného pro získání aktiva následují transakce, tzn. nová transakce vzniká po ukončení předchozí transakce. Příkladem může být, že Portál A získá jako výstup transakce s Portálem B určitý autentizační faktor např. heslo, které mu umožní v následující transakci s Portálem C získat další aktivum, např. přístup k souborům.

Pokud se jedná o *vložení transakcí*, tak je nová transakce vytvořena a ukončena během jiné transakce za účelem jejího ukončení. Příkladem může být, že Portál A žádá Portál B o určitá aktiva, Portál B však není schopen žadatele autentizovat a proto vytvoří další transakci s Portálem C, který je autentizase schopen. Žadatel se tedy autentizuje u Portálu B, který jeho zprávy přeposílá Portálu C a tvoří tedy tranzitní uzel. Výstupem vložené transakce je pak výsledek autentizace, který zašle Portál C Portálu B ve zprávě FINISH a transakce se ukončí. Portál B pak rozhodne na základě tohoto výsledku a výsledku první transakce a zašle jej Portálu A ve zprávě FINISH a transakce se také ukončí. Výše popsané možnosti propojení transakcí je zobrazen na Obr. 17.



Obr. 17 Možné scénáře propojování transakcí Vlevo: zřetězení transakcí, Vpravo vložení transakce

7 JAVA CARD

V předchozí kapitole byl rozebrán protokol ACP, který má být implementován v autentizačním systému, který je cíleným výstupem této práce. Jelikož je autentizační systém založen na autentizaci uživatele za pomoci tokenu, jímž je konkrétně čipová karta, založená na platformě Java Card, bude se tato kapitola zabývat bližším náhledem na tuto technologii. Čerpáno bylo především z literatury [25], [26], [27] a [28].

7.1 Specifikace technologie

Java Card je jednou z dílčích platforem zastřešovaných Java Platformou (další platformy jsou např. Java SE - pro aplikace na PC, Java EE - pro aplikace pro informační systémy nebo Java ME - pro aplikace pro mobilní telefony). První karta Java Card byla představena již v roce 1996 společností Schlumberger. Všechny produkty Java Card jsou postaveny na specifikaci Java Card Platformy, vyvinuté firmou Sun Mircosystems.

Technologie Java card přizpůsobuje platformu Java pro použití na čipových kartách a dalších podobných s nižším výpočetním výkonem a menším paměťovým prostorem. Jednotlivé aplikace napsané v jazyce Java jsou pak na těchto zařízeních spustitelná. Tyto aplikace se nazývají applety (Java Card applet). Mezi hlavní výhody technologie Java Card patří:

- *Multiplatformovost* - technologie je nezávislá na platformě a proto applety vytvořené pro jeden typ karet jsou použitelné i u jiných karet různých výrobců, založených na technologii Java Card. Java Card je tedy multiplatformí, stejně jako Java.
- *Multifunkčnost* - technologie poskytuje možnost uložení více aplikací na kartu, čímž se karta stává multifunkční, přičemž jednotlivé applety jsou od sebe vzájemně oddělené pomocí ferewallu, který tím poskytuje určitou míru bezpečnosti.
- *Kódování* - Odlišným kódování známého z Javy, je dosaženo optimalizace zdrojového kódů pro velikost a proto má mnohem menší nároky na paměť, která je u karet velmi ceněná.
- *Kompatibilita* - technologie je kompatibilní se současnými existujícími standardy čipových karet, jako je ISO 7816.

Technologie Java Card definuje sadu specifikací pro vývoj appletů na čipové karty. Jedná se o určitou podmnožinu technologie Java. Díky tomu je možné na čipových kartách spouštět applety napsané v Javě. Zachovává si mnoho výhod programovacího jazyka Java, jako je bezpečnost, robustnost, přenositelnost. Virtuální stroj, definice jazyka a balíčky jsou více kompaktní a stručnější, aby bylo možné technologii použit na čipových kartách. Zařízení podporující tuto specifikaci jsou označována jako Java Card Platforma. Java Card Platforma se skládá ze tří základních částí dle [25]:

- *Java Card Virtual Machine (JCVM)* - definuje podmnožinu jazyka Java vhodnou pro použití na čipových kartách. Jedná se o virtuální stroj (počítač), zajišťující interpretaci bajtkódu (bytecode) na čipové kartě. Bajtkódem se rozumí přeložený zdrojový kód, který je obdobný klasickému bajtkódu známého z Javy, pouze je použito odlišné kódování, optimalizované pro velikost. Jednotlivé applety tedy neběží přímo ve strojovém kódu zařízení ve kterém jsou spuštěny ale právě v JCVM.
- *Java Card Runtime Environment (JCRE)* - jedná se o běhové prostřední, které poskytuje vše potřebné pro běh appletů na čipové kartě. Provádí kompilaci a spuštění programu psaného v jazyce Java, správu paměti zacházení s applety a jinými potřebnými prvky.
- *Java Card Application Programming Interface (API)* - definuje framework a rozšíření Java Card balíčků a tříd pro applety čipových karet.

Bližší popis jednotlivých součástí popisují následující podkapitoly.

7.1.1 JC virtuální stroj (JCVM)

JC virtuální stroj umožňuje spouštění appletů nahraných na čipové kartě v bajtkódu (respektive v optimalizovaném formátu CAP (Converted applet)) a dále má na starosti správu paměti a objektů. Oproti klasickému JVM (Java Platforma), je hlavní rozdíl v tom, že se JCVM skládá ze dvou částí, jedna pro běh mimo kartu a druhá pro běh na kartě:

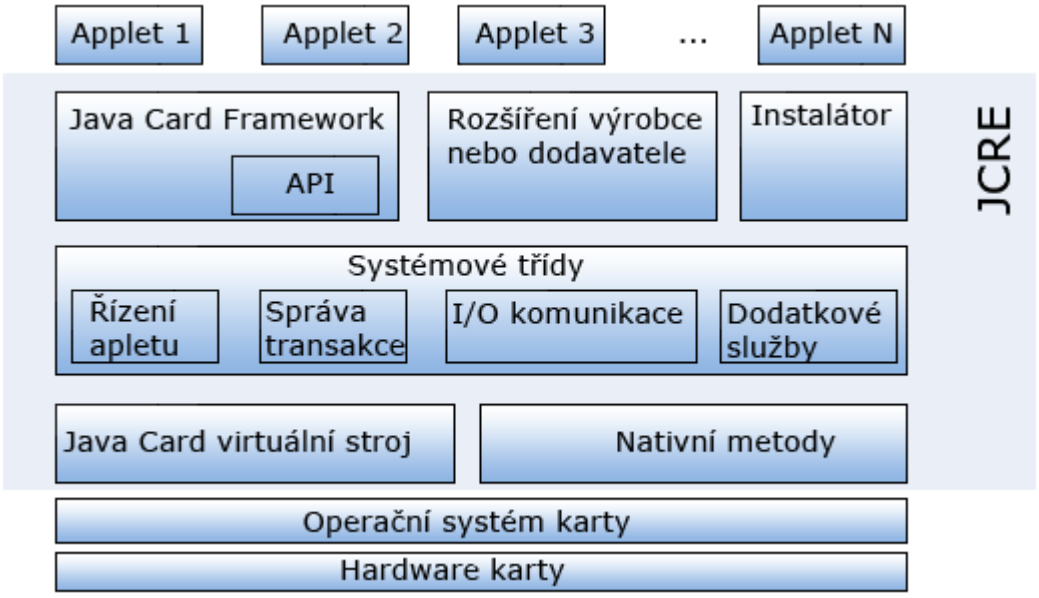
- *Převodník (converter)* - nachází se mimo kartu např. v uživatelském počítači. Jeho úkolem je načíst, ověřit a také připravit data všech tříd a připravit je pro použití na čipových kartách. Princip je takový, že při kompilaci appletu, vzniká standardní bajtkód (soubor s příponou *.class) jazyka Java. Všechny třídy a potřebné soubory jsou pak kompilérem předány překladači. Převodník pak optimalizuje bajtkód a vytváří *exportní soubor* a *CAP soubor*, který je spustitelný překladačem. CAP (converted applet) je nový typ souboru navržený speciálně pro účely použití na kartách Java Card. CAP soubor je na kartě spustitelný. Exportní soubor je obdoba hlavičkového souboru v jazyce C a není tedy na kartě spustitelný. Applet je následně uložen na kartu. Převodník také detekuje všechny prostředky jazyka Java, které nejsou specifikací Java Card podporovány.
- *Překladač (interpreter)* - nachází se na kartě a zajišťuje spuštění bajtkódu, zajišťuje také přidělování paměti a vytváření objektů.

Jak již bylo uvedeno, JC virtuální stroj definuje pouze určitou podmnožinu jazyka Java. Veškeré konstrukce jsou totožné s tímto jazykem, avšak mnoho funkcí a datových typů Java Card nepodporuje. Mezi zásadní rozdíly oproti klasické Javě patří:

- Java Card podporuje pouze primitivní datové typy, tzn. nepodporuje datové typy jako jsou `char`, `double`, `long`, `float`, podpora pro `int` je volitelná (naopak je zachována podpora `boolean`, `byte`, `short`).
- Java Card podporuje pouze jednorozměrná pole nikoli vícerozměrná pole.
- Java Card nepodporuje vlákna

7.1.2 JC běhové prostředí (JCRE)

Běhové prostředí Java Card JCRE lze obecně považovat jako operační systém čipové karty. Toto prostředí je na kartu nahráno při výrobě a nelze jej odstranit jinak, než fyzickou likvidací. Zajišťuje správu JCVM, knihoven a zdrojů, zajišťuje síťovou komunikaci a bezpečnost celého systému na kartě. Prostředí zahrnuje komponenty jako je výše uvedený virtuální stroj a konstrukční základ tříd API, díky čemuž je spuštění appletů na různých platformách jednodušší. Dále pak blok nativních metod, který pracuje s transportními protokoly, kryptografickou kontrolou a správou pamětí. Systémové třídy zajišťují běh appletů (řízení appletů), o správu transakcí (správa transakce), o výměnu dat mezi kartou a čtecím zařízením (I/O komunikace), případně mohou obsahovat nějaké dodatkové služby. JCRE obsahuje také instalátor, který zajišťuje instalaci (nahrátí) appletu na čipovou kartu (pracuje se soubory CAP). Instalátor není povinnou součástí JCRE, avšak při jeho absenci není možné nahrávat na kartu nové applety, a je tedy nutné si vystačit pouze s applety, které byly na kartu nahrány při její výrobě. Rozšíření výrobce nebo dodavatele obsahují specifické knihovny, potřebné pro dosažení určitého cíle, např. při vyšších potřebách zabezpečení u bankovních operací. Výše popsaná struktura JCRE je zobrazena na Obr. 18.



Obr. 18 Umístění JCRE na platformě Java Card

Jelikož jsou čipové karty napájeny pouze v případě, kdy jsou připojeny do čtecího zařízení, má JCRE a všechny vněm obsažené komponenty unikátní životní cyklus, který začíná vyrobením karty a končí zničením nebo vyřazením karty. JCRE je inicializovánou pouze jednou, poté je stav JCRE a všech komponent uložen v napěťově nezávislé paměti EEPROM. V případě odpojení napájení je běh virtuální stroj přerušen a obsah paměti RAM je ztracen. Všechny stavy jsou však uloženy v paměti EEPROM ze které se po opětovném připojení napájení opětovně obnoví pro běh VS a všech ostatních komponent a očekávají se další vstupy.

7.1.3 JC aplikační programové rozhraní (API)

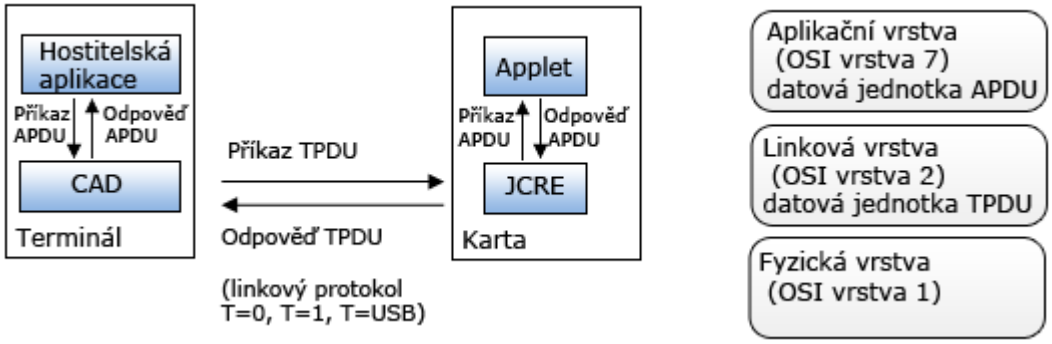
Jedná se o programovací rozhraní pro psaní zdrojových kódu za pomocí balíčků a knihoven technologie Smard Card. Obsahuje malou podmnožinu API tradičního programovacího jazyka Java. Neexistuje zde podpora pro řetězce, ani pro více vláken. Kromě malé podmnožiny tříd z Javy, definuje také vlastní sadu tříd pro podporu Java Card aplikací, v balíčcích:

- `java.lang` - v tomto balíčku se nachází třída `Object`, která definuje kořenovou oblast hierarchie Java Card, nebo také třída `Throwable`, potřebná pro vytváření různých výjimek a detekci chyb. Žádné jiné třídy z tradiční `java.lang` zde nejsou zahrnuty.
- `javacard.framework` - balíček obsahující třídy sloužící k běhu appletu, např. třídu `Applet` definující jeho zpuštění, třídu `APDU` poskytující metody pro komunikaci pomocí datových jednotek APDU, třídu `AID` slouží k identifikaci appletu a další.
- `java.rmi` - definuje třídy `RemoteInterface` a `RemoteException`. Jinak zde nejsou zahrnuty žádné z tradičních tříd.
- `java.io` - obsahuje pouze třídu `IOException`.
- `javacard.security` - balíček definující třídy pro podporu zabezpečení (symetrický (např. AES nebo asymetrické (např. RSA) šifrovací algoritmy, metody pro výpočty kontrolních součtů CRC, hashů či podpisů)..

7.2 Princip komunikace

Komunikace terminálu a čipové karty na aplikační úrovni je založena na výměně datových jednotek označovaných jako APDU (Application Protocol Data Unit). APDU je logický datový paket, který lze rozdělit do dvou typů a to příkazu APDU (command) a odpovědi APDU (response). Příkazy APDU posílá vždy terminál směrem ke kartě, na níž se nachází JCRE (konkrétně Java Card Framework), která tyto příkazy na základě identifikátoru AID (Application identifier) zasílá příslušnému appletu. Applet poté zpracuje příkaz a zašle přes JCRE odpověď APDU zpět terminálu. Formát žádosti a odpovědi jsou standardizovány normami ISO/IEC 7816-3 a 7816-4. Občas je možné se také setkat s označením CAD (Card Acceptance Device), jedná se o zařízení, které se nachází mezi hostitelskou aplikací a kartou např. čtecí zařízení připojené k terminálu pomocí USB rozhraní nebo je vněm přímo integrovaná (bankomaty).

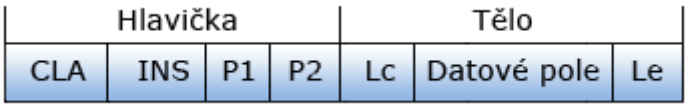
Při komunikaci mezi čtecím zařízením a kartou se obvykle používá jeden ze dvou linkových protokolů *T=0* - bajtově orientovaný protokol nebo *T = 1* - blokově orientovaný protokol, přičemž je možné použít také alternativní protokoly *T = USB* nebo *T = RF*. Datovou jednotkou na transportní úrovni je TPDU (Transport Protocol Data Unit). Výše popsaný princip komunikace je zobrazen na Obr. 19, vpravo je pak zobrazena komunikace z pohledu OSI modelu. [26]



Obr. 19 Princip komunikace mezi terminálem a kartou

7.2.1 Příkaz APDU

Příkaz APDU (command) je posílán vždy směrem na kartu a jeho strukturu udává první bajt. Příkaz se skládá z povinné hlavičky a volitelného těla. Struktura je zobrazena na Obr. 20.

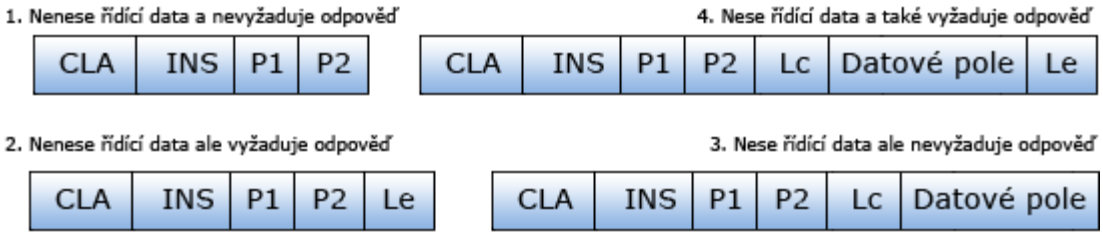


Obr. 20 Struktura APDU příkazu

Význam jednotlivých částí je následující:

- *CLA* - (class) má velikost 1 bajt a slouží k identifikaci instrukční třídy, platné hodnoty jsou definovány normou ISO 7816-4.
- *INS* - (instruction) má velikost 1 bajt, a označuje konkrétní instrukci z instrukční třídy definované v CLA. Opět jsou hodnoty INS definovány v normě ISO 7816-4.
- *P1* - (parameter 1) má velikost 1 bajt, slouží k definici parametru 1.
- *P2* - (parameter 2) má velikost 1 bajt, slouží k definici parametru 2.
- *Lc* - (length command) má velikost 1 bajt, a udává velikost datového pole v bajtech (nepovinné pole).
- *Datové pole* - toto pole má proměnnou velikost (udána v Lc). Obsahem jsou transportní data (nepovinné pole).
- *Le* - velikost 1 bajt, a určuje maximální velikost datového pole v očekávané odpovědi v bajtech (nepovinné pole)

Z výše uvedeného tedy vyplývá, příkaz APDU může mít čtyři struktury, viz Obr. 21.



Obr. 21 Možné struktury příkazu APDU

7.2.2 Odpověď APDU

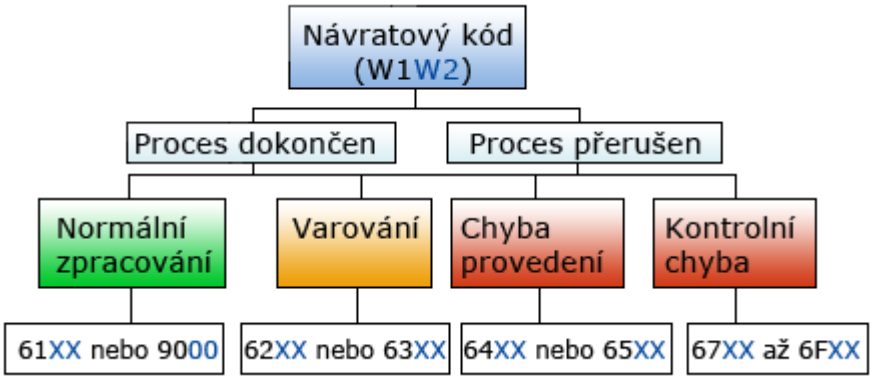
Odpověď APDU (response) je posílána vždy směrem od karty a je rekcí na příkaz APDU. Odpověď se skládá z volitelného těla a povinného zápatí. Má jednodušší strukturu než příkaz APDU, viz. Obr. 22.

Obr. 22 Struktura APDU odpovědi

Význam jednotlivých částí je následující:

- *Datové pole* - velikost je proměnná avšak předem dána maximální velikost v položce Le v příkazovém rámci. Obsahuje transportní data odeslaná appletem (nepovinné pole).
- *SW1* - (status word 1) má velikost 1 bajt a jedná se o 1. stavové slovo, které určuje třídu dané indikace nebo chyby.
- *SW2* - (status word 2) má velikost 1 bajt a jedná se o 2. stavové slovo, které specifikuje detaily.

Hodnoty stavových slov jsou definovány normou ISO 7816-4. Význam návratových kódu definovaných normou jsou zobrazeny na Obr. 23.

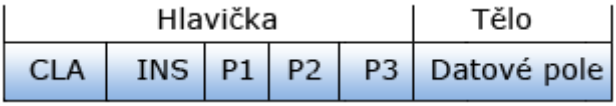


Obr. 23 Význam návratových kódů definovaných normou ISO 7816-4

Například tedy, pokud je návratový kód ve tvaru *9000* (došlo k normálnímu zpracování, bez další specifikace), *6101* (došlo k normálnímu zpracování avšak v odpovědi bylo odesláno o 1 bajt dat více, než bylo specifikováno v poli Le v příkazovém APDU), *6281* (varování, část dat může být poškozena), *6700* (kontrolní chyba, špatná délka, nesouhlasí s hodnota v Lc).

7.2.3 Transportní protokol

Transportní protokoly pracují na úrovni linkové vrstvy a jsou definovány normou ISO/EIC 7816-3. Datové jednotky přenášené transportními protokoly, označované jako TPDU, neobsahují mechanizmus pro opravu chyb. Mezi nejpoužívanější transportní protokoly se řadí T=0 - jedná se o starší protokol, který je bajtově orientovaný, poloduplexní, asynchronní a odděluje příkazy a odpovědi na příkazy, T=1 novějším protokolem, který je blokově orientovaný, u kterého příkazy a odpovědi nejsou striktně odděleny, díky čemuž je efektivnější. Je rovněž poloduplexní a asynchronní. Alternativou je pak T=USB, který dosahuje vyšších přenosových rychlostí než předchozí zmíněné protokoly a navíc zajišťuje kompatibilitu s PC prostředím. Transportní protokoly pak mapují datové jednotky APDU do TPDU. Přenosová datová jednotka TPDU se skládá z hlavičky obsahující stejné parametry CLA, INS, P1, P2 jako APDU. Navíc však obsahuje také parametr P3, který jako jediný poskytuje informaci o velikosti jak dat u příkazového APDU (indikované Lc) tak očekávané velikosti odpovědi APDU (indikované Le). Dále pak obsahuje samotná data viz Obr. 24.



Obr. 24 Struktura příkazu TPDU u použitého protokolu T=0

7.3 Java Card aplikace

Kompletní Java Card aplikace se skládá z několika komponent, které dohromady tvoří bezpečnou end-to-end aplikaci. Patří sem back-end aplikace a systém, hostitelská aplikace běžící mimo kartu (off-card), čtecí zařízení (CAD), applet běžící na kartě (on-card), osobní informace uživatele a další podpůrné softwary.

7.3.1 Back-end aplikace

Back-end aplikace jsou prvky systému, které poskytují služby appletům na kartě. Aplikace může například poskytovat spojení s bezpečnostním systémem a spolu s ověřením na kartě tak docílit vyšší bezpečnosti. Příkladem může být například elektronický platební systém, kde back-end aplikace poskytuje přístup ke kreditní kartě a k platebním informacím.

7.3.2 Hostitelská aplikace

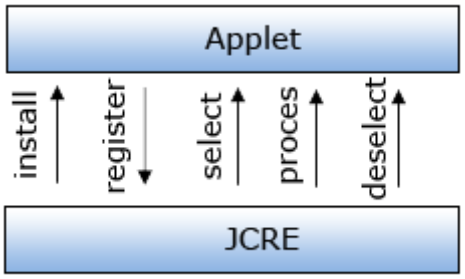
Nachází se na terminálu, kterým může být PC, bankomat, mobilní telefon a pod. Zajišťují komunikaci mezi uživatelem, appletem na kartě a poskytovatelem back-end

aplikací. Zprostředkování komunikace mezi hostitelskou aplikací a appletem poskytuje CAD (Card Acceptance Device). Jedná se o čtečku čipových karet a kromě zprostředkování komunikace také poskytuje napájení čipové kartě. CAD může být připojen k terminálu (např. PC) pomocí sériového portu nebo do něho integrován jako je tomu u bankomatů.

7.3.3 Java Card Applet

Java Card applet je aplikace (program) napsaná v jazyce Java, dodržující sadu pravidel, jenž mu umožňuje běh pod JCRE. Každý applet musí rozšiřovat třídu `javacard.framework.Applet`, která je základní třídou pro všechny applety a definuje potřebné proměnné a metody.

Nahrání appletu na kartu provádí JCRE voláním metody `Applet.install()`. Jakmile je applet nahrán na kartu, je propojen s ostatními balíčky na kartě. Poté se applet registruje u JCRE pomocí metody `Applet.register()`. Jakmile je applet nainstalován a zaregistrován, tak se nachází v neaktivním stavu a je připraven k výběru hostitelskou aplikací a následně zpracování jejích APDU příkazů. Teprve potom přechází do aktivního stavu, tzn. Applet je tedy pasivní aplikace, která čeká až na příkaz od terminálu. Vybrání appletu je způsobeno zasláním žádosti terminálu SELECT APDU nebo MANAGE CHANNEL APDU směřované JCRE. Pro výběru appletu, který je hostitelskou aplikací vybrán, volá JCRE metodu `Applet.select()`. Je-li applet vybrán, JCRE přeposílá příchozí APDU příkazy appletu pomocí metody `Applet.process()`. K zrušení výběru dochází, pokud uživatelská aplikace zašle APDU příkaz JCRE s výběrem jiného appletu. JCRE pak volá metodu `Applet.deselect()`, čímž se applet vrátí do neaktivního stavu. Výše uvedená komunikace bývá také označována jako životní cyklus appletu viz. Obr. 25.



Obr. 25 Životní cyklus appletu

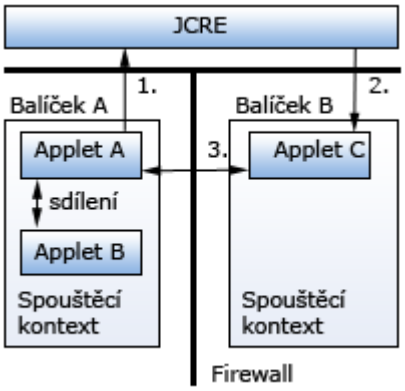
Jelikož je technologie Java Card multiaplikační, tak se na kartě může vyskytovat více apletů a proto je každá instance appletu označena unikátním identifikátorem AID (Application identifier). V případě, že hostitelská aplikace chce komunikovat s konkrétním appletem, vyšle příkaz SELECT APDU obsahující AID daného appletu. JCRE pak směřuje následující příkazy od terminálu právě appletu s daným AID. AID má velikost 5 - 16 bajtů a je definován normou ISO 7816-5.

7.4 Bezpečnost technologie

Jak již bylo zmíněno, technologie Java Card umožňuje umístění více appletů na kartě, což by mohlo přinášet určitá bezpečnostní rizika. Aby se zabránilo vzájemnému ovlivňování appletů, je každý applet přiřazen do tzv. kontextu řízení, který řídí přístup k objektům, které se vněm nacházejí. Hranice mezi jednotlivými kontexty řízení tvoří Firewall. Firewall vytváří virtuální oddíly, ve kterých jednotlivé applety mají navzájem přístup k svým objektům, které jsou označeny modifikátorem public. Všechny applety jednoho balíčku tedy spadají do jednoho kontextu řízení. Pokud potřebuje určitý applet přístup k objektům jiného appletu v jiném spouštěcím kontextu, je postup následující:

Applet A vyžaduje přístup k objektu appletu C. Volá metodu `JCSystem.getAppletShareableInterfaceObject()`, žádost o sdílení zasílá JCRE. JCRE zažádá applet C jménem appletu A o sdílení, voláním metody `getShareableInterfaceObject()`. Pokud applet C sdílí tento objekt, tak na něj applet A získá odkaz a bude mít tedy k dispozici objekty, které sám vytvoří a také všechny sdílené objekty appletu C.

V případě, že se v kontextu řízení, kde se vyskytuje Applet A nachází také jiný applet např. Applet B, bude mít i ten přístup k sdíleným objektům Appletu C. Výše uvedené je zobrazeno na Obr. 26.



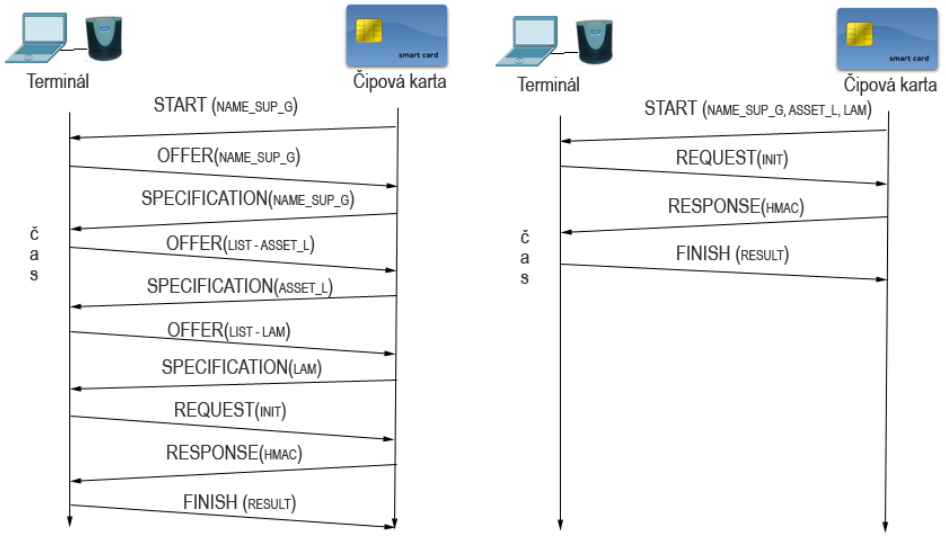
Obr. 26 Java Card sdílení objektů

8 PRAKTICKÁ ČÁST

Praktická část diplomové práce se zabývá konkrétním návrhem autentizačního systému, využívající protokol ACP popsaný v kapitole 6. Cílem je vytvoření a realizace návrhu systému autentizace založeném na protokolu ACP, kde se autentizuje uživatel disponující čipovou kartou vůči aplikaci spuštěné na uživatelském počítači.

8.1 Návrh komunikačního schématu

Navržený systém se stává ze dvou aplikací. Aplikace běžící na čipové kartě (applet) a aplikace běžící na uživatelském počítači (hostitelská aplikace). Schéma průběhu autentizace je zobrazeno na Obr. 27.



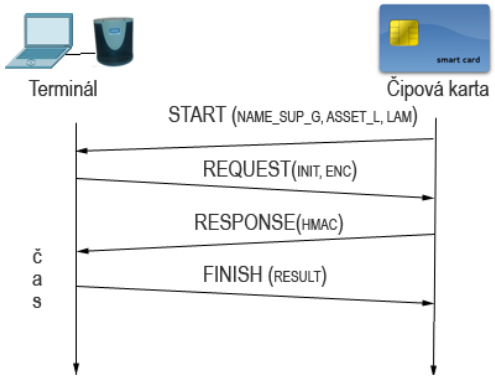
Obr. 27 Navržené komunikační schéma Vlevo: neúplná zpráva START, Vpravo: úplná zpráva START

Žadatelem je čipová karta a poskytovatelem je uživatelský počítač disponující čtečkou čipových karet. Při autentizaci jsou možné dva scénáře. V prvním scénáři Obr. 27 (vlevo) pošle žadatel neúplnou zprávu START tzn. neobsahuje všechna náležitá AVP potřebná k ověření přístupu. Například pošle pouze své jméno NAME_SUP_G. Poskytovatel tedy správu START úplně ignoruje a přejde do stavu vyjednávání, kdy odesílá zprávu OFFER s prázdným AVP typu NAME_SUP_G, tzn. má délku rovnu nule a neobsahuje žádnou hodnotu. Uživatel musí reagovat zasláním zprávy SPECIFICATION s AVP stejného typu, přičemž jako hodnotu uvádí své jméno. Dále poskytovatel zasílá AVP typu ASSET_L, kde uvádí seznam dostupných aktiv, přičemž žadatel volí právě jedno aktivum např. autentizace uživatele pro účely otevření dveří a volbu zasílá zpět poskytovateli ve zprávě SPECIFICATION obsahující jediné AVP typu ASSET_L nesoucí kód zvoleného aktiva. Poskytovatel poté ověří, zda uživatel má vůbec práva o přistupování k danému aktivu. Pokud nemá, je zaslána zpráva FINISH informující o zamítnutí přístupu. V opačném případě, je zaslán seznam podporovaných

autentizačních metod pro vybrané aktivum, na což žadatel opět reaguje vybráním konkrétní metody a jejím zasláním zpět. Poté následuje fáze autentizace, kdy poskytovatel zašle zprávu REQUEST obsahující AVP typu INIT, nesoucí náhodné číslo R, na což žadatel musí reagovat zprávou RESPONSE, obsahující AVP typu HMAC, nesoucí hash SHA-1 řetězce náhodného čísla spojeného s heslem žadatele. Pokud proběhla autentizace úspěšně je odeslána zpráva FINISH, obsahující AVP typu RESULT s hodnotou 1 (autentizace s žadatelem proběhla úspěšně, ale samotný proces ještě běží). V opačném případě nese hodnotu 2 (zamítnutí přístupu k aktivu).

Celý proces je možné zkrátit zasláním zprávy START obsahující všechna potřebná AVP, čímž dojde k vynechání fáze sjednávání a ihned začne fáze autentizace, což celý proces urychlí viz. Obr. 27 (vpravo).

Další navržený scénář zavádí větší míru zabezpečení, jelikož je zde vytvořena dvoufaktorová autentizace. K autentizaci uživatele, je kromě přístupové karty vyžadována také znalost pinu karty. Čímž se snižuje riziko zneužití karty neoprávněnou osobou. Tento scénář je zobrazen na Obr. 28.



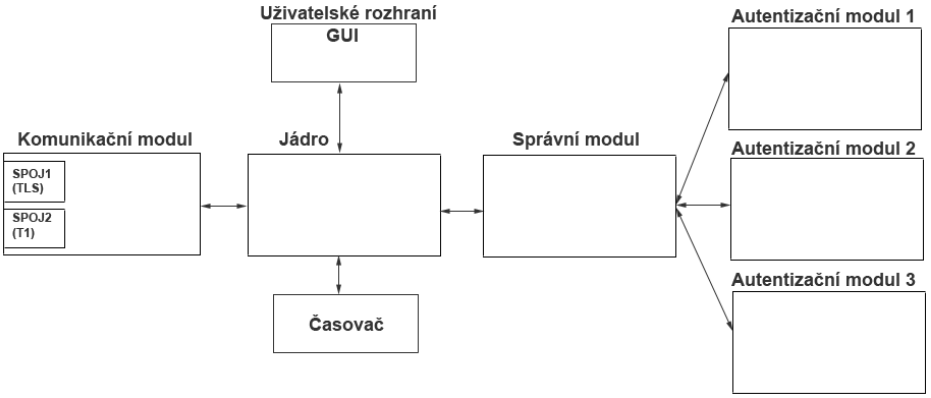
Obr. 28 Scénář dvoufaktorové autentizace

Jedná se o obdobu předchozího scénáře, v tomto případě však terminál spolu s AVP INIT zasílá také kontejnérové AVP typu ENC, které obsahuje AVP INIT (obsahuje inicializační vektor iv) a AVP 3DES. AVP typu 3DES bylo zvoleno, z důvodu nepodporování šifrování AES na čipové kartě. Pro 3DES byl zvolen typ 176, který spadá do oblasti rezervovaných AVP. V tomto AVP je pak přenášen heš pinu zadaný uživatelem. Karta následně ověří předložený pin a jako odpověď odešle podepsané náhodné číslo výzvy obsažené v INIT. Server ověří pravost podpisu a na jejím základě povolí či zamítne přístup k požadovanému aktivu a o výsledku autentizace informuje kartu ve zprávě FINISH. Za předpokladu, že portál nedisponuje veřejným klíčem karty, je vytvořena nová transakce, s domácím portálem žadatele, který autentizaci provede. Tento scénář je pak testován v kapitole 11.

Z výše uvedeného je patrné, že navržený autentizační systém využívá variantu ACP protokolu jednoduchých odpovědí tzv. ACP-VSA (Variant of Single Answers). Tato varianta je založena na principu, že poskytovatel zasílá žadateli dotaz (prázdné AVP požadovaného typu) nebo nabídku tj. N různých AVP stejného typu, přičemž žadatel ve svých odpovědích poskytovateli odesílá pouze jediné zvolené či vybrané AVP. Zašle-li jich více, portál poskytovatele ukončí transakci a na další zprávy od žadatele již nereaguje.

9 NÁVRH ACP PORTÁLU (UŽIVATELSKÁ APLIKACE)

ACP portál je součástí síťových zařízení, sloužící k řízení přístupu jiných zařízení ke svým aktivům nebo pro vyjednávání přístupu k aktivům jiných zařízení. Portál se skládá z jádra a volitelných modulů, kterých může být v portálu více jednoto typu. Mezi volitelné moduly patří komunikační modul umožňující komunikaci dvou zařízení pomocí vybraného spoje. V navrženém systému je implementována podpora spojů TCP, TLS a T1. Dále sem patří správní modul, definující algoritmus řízení přístupu k danému aktivu a tedy řízení způsobu průběhu transakce mezi žadatelem a poskytovatelem. Správní modul obsahuje také databázi aktiv, a uživatelů. Autentizační modul pak zajišťuje samotné ověřování uživatele za pomoci určité autentizační metody. Jádro portálu se stará o samotné předávání zpráv, udržování informací o aktivních spojích a transakcích, o jejich vytváření a rušení a přijímá požadavky od uživatele prostřednictvím uživatelského rozhraní. Jednotlivé části portálu jsou dle výše uvedeného strukturovány v programu do jednotlivých balíčků dle jejich funkce v ACP portálu. Vzájemné propojení jednotlivých součástí portálu je zobrazeno na Obr. 29.



Obr. 29 Struktura ACP portálu

9.1 Komunikační modul

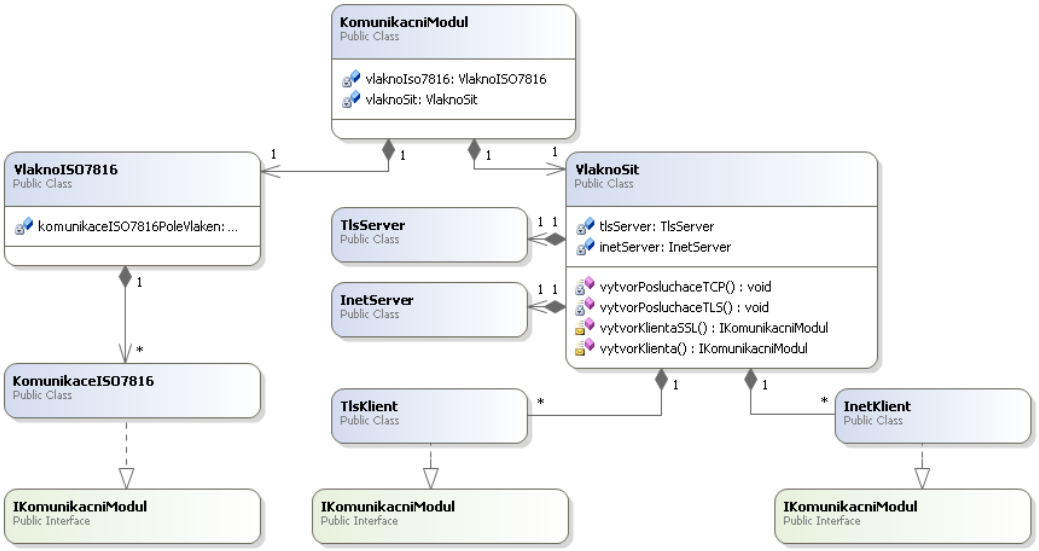
Tato část programu zajišťuje komunikaci portálu s portály implementovanými na vzdálených zařízeních. Hlavním úkolem modulu je aktivovat podporovaná komunikační rozhraní a přijímat a odesílat s jejich pomocí data (ACP zprávy) protistraně. Třídy realizující činnost komunikačního modulu jsou obsaženy ve stejnojmenném balíku `komunikacniModul`.

Při inicializaci jádra portálu, se vytváří instance komunikačního modulu. Díky čemuž se komunikační modul aktivuje a ihned inicializuje. Tento modul v současnosti implementuje rozhraní ISO 7816 definující komunikaci s kontaktními čipovými kartami a síťové rozhraní. Přičemž jednotlivá rozhraní jsou realizována samostatnými vlákny sloužícími pro obsluhu daného rozhraní. V případě potřeby rozšíření komunikačního modulu o další typ rozhraní je tedy možné program jednoduše o toto rozhraní rozšířit vytvořením dalšího vlákna obstarávajícího tuto funkci.

9.1.1 Komunikace s čipovou kartou

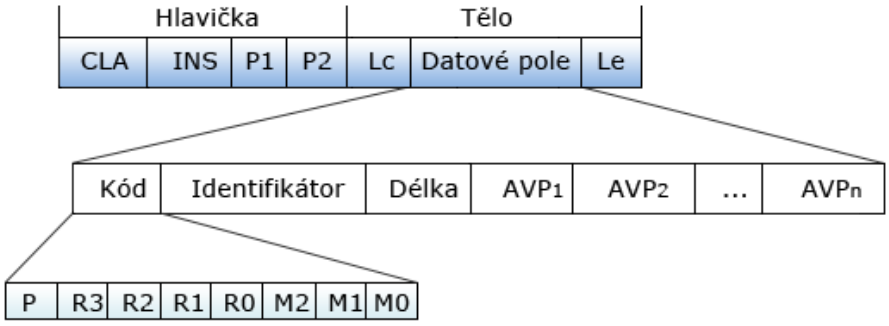
Úkolem rozhraní ISO 7816 (realizovaného třídou `VlaknoISO7816`) je zjišťovat přítomnost čtecích zařízení a pro všechna připojená zařízení vytvářet vlákna pro jejich obsluhu. Především se jedná kontrolu připojení čipové karty, výběr appletu a přijímání a odesílání APDU správ čipové kartě. Rozhraní dále udržuje seznam všech aktivních připojených zařízení.

Vlákna obsahují ukazatel na jádro portálu, jehož zprostředkováním dochází k předávání zpráv mezi jednotlivými spoji a jádrem portálu. Na Obr. 30 je zobrazena hierarchie struktury komunikačního modulu.



Obr. 30 Hierarchie komunikačního modulu

V případě rozhraní ISO 7816 předává jádro portálu proud dat k odeslání objektu `KomunikaceISO7816`, který je koncovým prvkem v hierarchii komunikačního modulu a zprostředkovává samotné odeslání dat. Jedná-li se o proud dat samotné zprávy ACP, je tato zpráva zapouzdřena do APDU příkazu viz. Obr. 31. Všechny koncové prvky komunikačního modulu musí implementovat rozhraní `IKomunikacniModul`.



Obr. 31 Zapouzdření ACP zprávy do APDU rámce

V případě bezchybného přenosu, tzn. příchozí zpráva od čipové karty obsahuje návratový kód ve tvaru 9000. Dochází k vyjmutí ACP zprávy z APDU odpovědi

a následnému sestaveni struktury ACP zprávy a jejímu předání jádru portálu. V opačném případě, není zpráva dále zpracovávána a následně dochází k vypršením časovače a k zrušení transakce.

9.1.2 Síťová komunikace

Při inicializaci správního modulu, je také spuštěno vlákno zajišťující komunikaci portálu po síti. Toto vlákno je vytvořeno pouze jedno viz. Obr. 29. Objekt `vlaknoSit`, vytváří a poté udržuje ukazatele na objekty `InetServer` a `TlsServer`. `InetServer` reprezentuje posluchače pro spoj TCP. Naslouchá na předdefinovaném portu (získáno z nastavení - v práci port 12345). Připojí-li se na tento port nový klient dojde k přijetí klienta, pomocí metody `accept()`, výsledkem je pak získání socketu reprezentujícího dané spojení. Následně je vytvořen `InetKlient`, jemuž je soket v konstruktoru předán. Tento klient pak již přebírá obsluhu požadavků vzdáleného portálu. Posluchač poté opět naslouchá a očekává připojení dalšího klienta. `TlsServer` je obdobou `InetServer`, s tím rozdílem, že reprezentuje posluchače pro spoj TLS a poskytuje tedy zabezpečený přenos dat proti odposlechu. Defaultní port pro naslouchání je získán z nastavení (v práci port 43215). Jelikož, je TLS spoj vytvořen nad spojem TCP, je `TlsServer` potomkem třídy `InetServer` viz Obr. 32.

Obr. 32 Závislost spoje TCP a TLS

`TlsServer` dědí všechny vlastnosti od posluchače TCP, s tím, že navíc obsahuje inicializační metodu, ve které jsou definovány úložiště klíčů (tzv. keystore), které obsahují jednak certifikát předkládaný serverem klientu, soukromý klíč, tak i certifikáty důvěryhodných certifikačních autorit. Pomocí metody `setNeedClientAuth(true)` je zajištěno, že svoji identitu musí prokázat také žadatel.

`InetKlient` a `TlsKlient` jsou koncovými body komunikace, jejichž instance si mezi sebou vzájemně předávají zprávy. Obsahují metody pro příjem a odeslání dat a musí implementovat rozhraní `IKomunikacniModul`. Příchozí zprávy od vzdáleného portálu pak předávají jádru prostřednictvím objektu `KomunikacniModul`. Z Obr. 32 je patrné, že opět třída `TlsKlient` je potomkem třídy `InetKlient` a přebírá tedy všechny jeho vlastnosti. Opět navíc obsahuje úložiště klíčů pro vzájemnou autentizaci. Pomocí metody `startHandshake()` dojde k vzájemné autentizaci a následně k vybudování zabezpečeného spoje (jeli autentizace úspěšná).

9.1.3 Vytvoření certifikátů

Navržený autentizační systém využívá pro komunikaci dvou ACP portálů zabezpečený spoj TLS, který využívá k vytvoření bezpečného kanálu certifikáty, jak bylo řečeno v předchozí kapitole. Konkrétně se zde jedná o tzv. oboustrannou autentizaci (*Two Way Autentizacion*), kdy je nezbytné pro vytvoření bezpečného kanálu autentizace obou stran tj. serveru i klienta. Aby nemusely všechny zúčastněné portály obsahovat certifikáty ostatních portálů, jsou v systému použity certifikáty podepsané CA, přičemž každý portál musí obsahovat certifikát této certifikační autority. Certifikační autorita byla vytvořena pouze pro účely této práce, přičemž v praxi by tato CA měla být nahrazena nějakou veřejnou důvěryhodnou certifikační autoritou. K vygenerování vlastního certifikátu a soukromého klíče, představující CA byl použit program OPENSSL. Certifikát *CA_xdzure00_DP_cert.pem* platný po dobu tří let a soukromý klíč *CA_xdzure00_DP_key.pem* reprezentující tuto CA se získají následujícím příkazem

```
OpenSSL> req -new -x509 -nodes -out CA_xdzure00_DP_cert.pem -keyout
CA_xdzure00_DP_key.pem -days 1095
```

Následně je potřeba vyplnit potřebné parametry jako je stát, název organizace apod. Vytvořený certifikát a soukromý klíč je možní si prohlédnout pomocí příkazů

```
OpenSSL> x509 -in CA_xdzure00_DP_cert.pem -text

OpenSSL> rsa -in CA_xdzure00_DP_key.pem -text
```

Certifikáty jednotlivých portálu, jsou vytvářeny pomocí programu *keytool*, který je součástí Javy a umožňuje vytvářet úložiště klíčů (keystore) chráněné heslem. Úložiště se soukromým klíčem a certifikátem se vytvoří příkazem

```
keytool -genkey -alias VUT_xdzure00_DP -keyalg RSA -keystore
VUT_xdzure00_DP_keystore.jks -keysize 1024
```

Opět se doplní požadované informace. Stejným postupem, se vytvoří úložiště klíčů pro druhý ACP portál *MU__xdzure00_DP_keystore.jks*. Poté je potřeba vygenerovat žádosti o podepsání CA, k tomu slouží následující příkaz

```
keytool -certreq -alias VUT_xdzure00_DP -keystore
VUT_xdzure00_DP_keystore.jks -file VUT_xdzure_request.csr
```

Jsou-li vytvořeny žádosti, je potřeba vystavit na základě těchto žádostí podepsané certifikáty, to se provede opět přes OPENSSL

```
OpenSSL>x509 -req -days 365 -in VUT_xdzure00_DP_request.csr -CA
CA_xdzure00_DP_cert.pem -CAkey CA_xdzure00_DP_Key.pem -set_serial 01 -
out VUT_xdzure00_DP_cert.crt
```

Tímto způsobem je vytvořen podepsaný certifikát certifikační autoritou. Stejným způsobem se to provede pro druhý certifikát. Po vytvoření certifikátu je potřeba jimi nahradit původní certifikáty v příslušných úložištích, předtím je však potřeba nahrát do úložiště certifikát CA, jinak by podepsaný certifikát nebyl považován jako důvěryhodný.

```
keytool -import -trustcacerts -alias CA_xdzure00_DP_root -file
CA_xdzure00_DP_cert.crt -keystore VUT_xdzure00_DP _keystore.jks
```

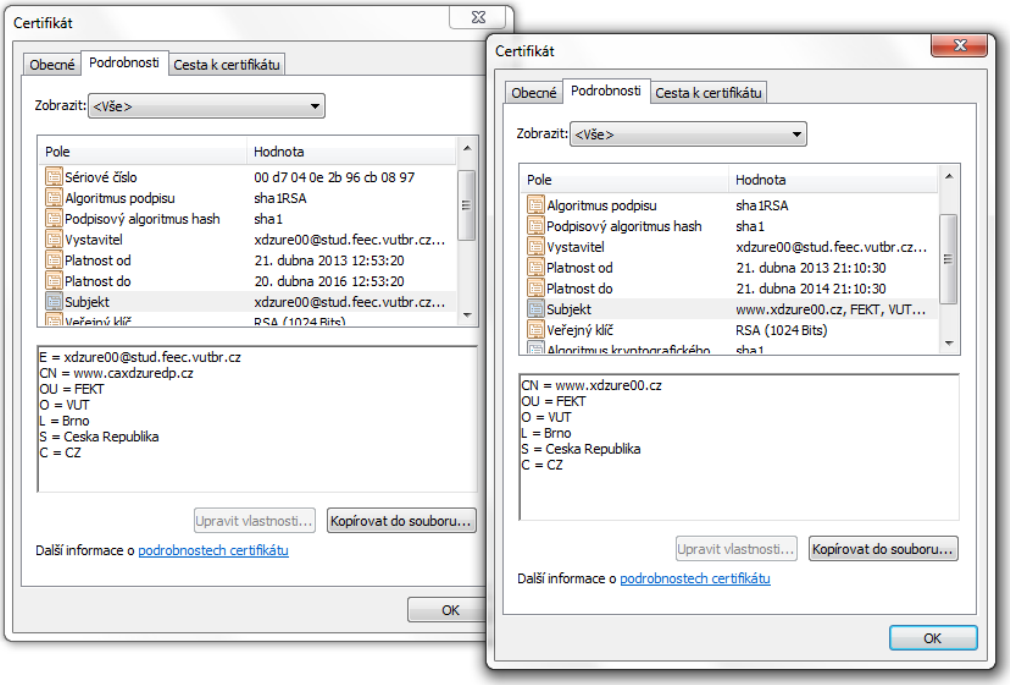
Nyní je možné nahradit starý certifikát v keystore. Každý portál pak obsahuje vlastní úložiště klíčů, chráněné heslem, kde má obsažen certifikát se svým veřejným klíčem, podepsaný CA, které důvěřují všechny portály. Svůj soukromý klíč a veřejný klíč CA.

```
keytool -import -alias VUT_xdzure00_DP -keystore
VUT_xdzure00_DP_keystore.jks -file VUT_xdzure00_DP_cert.crt
```

Zobrazit obsah keystore je možné příkazem

```
keytool -list -v -keystore VUT_xdzure00_DP_keystore.jks
```

Výsledný podepsaný certifikát používaný pro autentizaci portálu, při vytváření spoje TLS a certifikát důvěryhodné certifikační autority, která tento certifikát podepsala jsou zobrazeny na Obr. 33.



Obr. 33 Podepsaný certifikát důvěryhodnou CA pro ACP portál

9.2 Správní Modul

Stejně jako komunikační modul, tak i správní modul se nachází ve stejnojmenném balíku `spravniModul`. Vstupní třídou je zde `SpravniModul.class`. Reference na tento objekt je obsažena v záznamu pro danou transakci. Správní modul řeší průběh transakce a tede přístupovou politiku k požadovanému aktivu. Portál je konfigurován tak, aby implementoval základní variantu protokolu ACP, tzn. variantu jednoduchých odpovědí ACP-VSA (Variant of Single Answers).

Reference na tento objekt je v záznamu pro danou transakci vytvořen vždy při vytvoření nového záznamu a je tedy nezávislý na typu požadovaného aktiva. Je zde obsažena databáze aktiv, která poskytovatel poskytuje a také databáze všech autentizačních metod, která jsou na portálu definována. Správní modul dále obsahuje databázi uživatelů, respektive instanci třídy `PristupDatabaze`, která obsahuje metody pro načtení potřebného záznamu z databáze SQL. Více v kapitole 8.4.1.

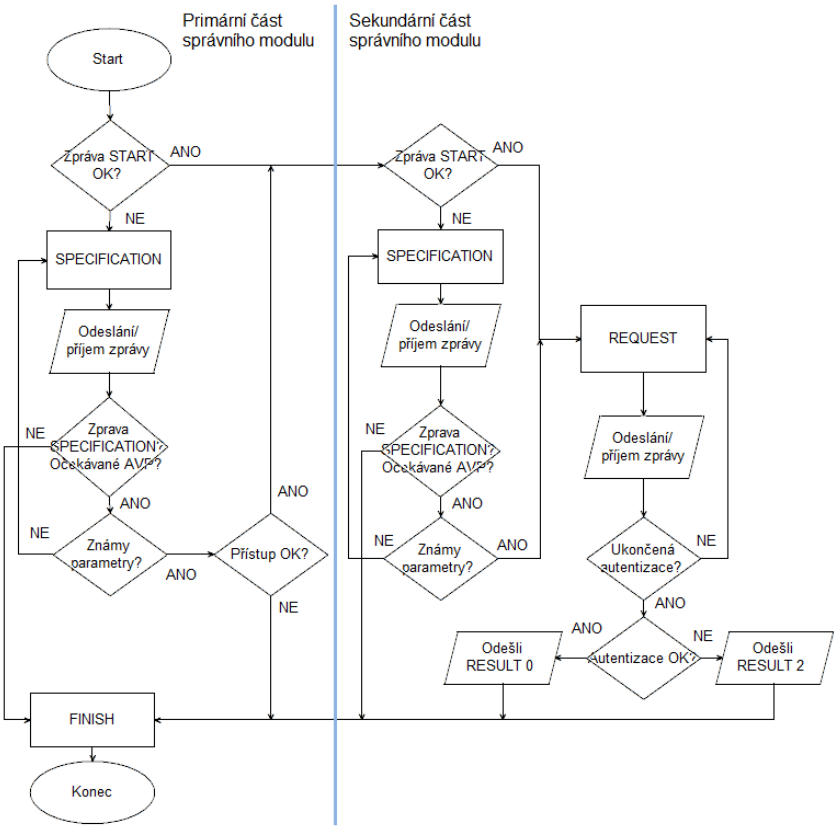
Hlavním úkolem objektu `SpravnihoModul` je kontrola příchozí zprávy START, zda obsahuje všechna potřebná AVP. Jedná se převážně o AVP typu NAME_SUP_G a ASSET_L, která poslouží k prvotní kontrole přístupu v rámci přístupové politiky. Správní modul pomocí jména žadatele zjistí, zda má uživatel přístupová práva k vybranému aktivu. Pokud žadatel toto právo nemá, je transakce ukončena odesláním prázdné zprávy FINISH a následně jsou vymazány všechny provozní informace související s danou transakcí.

V případě, že příchozí zpráva START neobsahuje potřebná AVP, je zahájena fáze sjednávání. V sekvenci zpráv OFFER a SPECIFICATION zjistí potřebná AVP. Provede to odesláním zprávy OFFER s prázdným AVP typu NAME_SUP_G. Žadatel pak na tuto žádost reaguje odesláním zprávy SPECIFICATION obsahující AVP typ NAME_SUP_G a jako hodnotu uvede své jméno. Obdobně je tomu i při zjišťování požadovaného aktiva, s tím rozdílem, že poskytovatel zasílá seznam dostupných aktiv. V případě, že žadatel na výzvu nereaguje, odešle AVP jiného typu než je vyžadováno nebo odpověď obsahuje více AVP, dojde k odeslání prázdné zprávy FINISH a k zrušení všech provozních informací souvisejících s danou transakcí.

Za předpokladu, že uživatel má právo přístupu k požadovanému aktivu, je z objektu reprezentujícího vybrané aktivum získán Správní modul, realizující přístupovou politiku k tomuto aktivu. Tomuto objektu je předán ukazatel na jádro portálu, záznam dané transakce a získaná AVP od žadatele a objekt reprezentující uživatele. Ukazatel na správní modul aktiva je poté uložen v záznamu transakce a následný průběh transakce se již odvíjí od definice přístupové politiky definované v příslušném správním modulu.

V případě Správního modulu aktiva, definovaného třídou `SpravniModulVO`, se nejprve zjišťuje typ obdržené zprávy. Po obdržení zprávy START, správní modul zjistí, zda jsou uvedeny všechny potřebné informace, zpravidla výběr autentizační metody. Pokud žadatel neuvedl autentizační metodu, je žadateli zaslána zpráva OFFER se seznamem podporovaných autentizačních metod pro vybrané aktivum. V případě, že je autentizační metoda známa, přejde správní modul do fáze autentizace.

Jednotlivé správní moduly jsou koncipovány formou stavového automatu, viz Obr. 34, kde je také patrné, jakým způsobem se přechází z primární části správního modulu (`SpravniModul.class`) do sekundární části správního modulu např. `SpravniModulVO`. Všechny správní moduly musí implementovat rozhraní `ISpravníModul`.

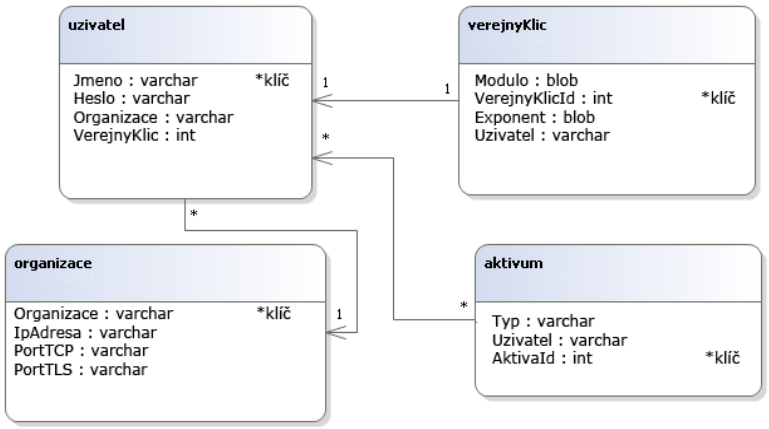


Obr. 34 Vývojový diagram funkcionality správního modulu

9.2.1 Databáze uživatelů

Databáze uživatelů je uložena v SQL databázi. Pro účely návrhu a testování byl použit program XAMPP simulující webový server Apache a databázi MySQL. Přistup do databáze a její správa je pak možná přes webové rozhraní pomocí phpMyAdmin.

Na serveru byla vytvořena databáze `databaze_uzivatelu_dp_xdzure00` a v ní tabulky, reprezentující uživatele, firmy, poskytovaná aktiva a veřejné klíče uživatelů. Vztahy jednotlivých tabulek jsou zobrazeny na Obr. 35.



Obr. 35 Závislosti tabulek v databázi

Tabulka `organizace` uchovává záznamy jednotlivých organizací zapojených do autentizačního systému. Jedná se o název organizace, ip adresu portálu této organizace a čísla portů, na kterých portál naslouchá. Jako klíč je v tabulce označen název organizace, tzn. název organizace se v tabulce smí nacházet pouze jednou. Záznam uživatele pak obsahuje informace o jménu konkrétního uživatele, jeho heslo a název organizace, podle kterého lze získat potřebné parametry pro případnou autentizaci u jeho domácí organizace. Každý uživatel může nakládat s určitým množstvím aktiv, která jsou vedena jako záznamy `aktivum`, přičemž mapování jednotlivých záznamů aktiv je realizováno pomocí jedinečného identifikátoru `UzivatelId`. Přístup do databáze realizuje třída `PristupDatabaze`, která obsahuje jednotlivé metody pro vyhledávání, ukládání a mazání jednotlivých záznamů v databázi pomocí SQL dotazů.

Jestliže Správní modul (jeho primární část) obdrží zprávu AVP typu NAME_SUP_G připojí se pomocí instance této třídy do databáze a vyhledá uživatele dle tohoto jména. Ze záznamu se zjistí, zda uživatel spadá do lokální organizace. V případě, že je uživatel lokální pokračuje proces autentizace na tomto portálu. V opačném případě jsou zjištěny směrovací informace tj. ip adresa vzdáleného portálu a port naslouchání na TLS spoji. Poté jsou tyto informace předány sekundárnímu správnímu modulu, který s nimi naloží dle své přístupové politiky. Obvykle dojde k autentizaci uživatele výzvou INIT a poté je vytvořeno spojení se vzdáleným portálem za účelem ověření uživatele. Po dokončení autentizace na základě zprávy FINISH vyhodnotí lokální portál, zda se uživateli povolí přístup k volenému aktivu a žadatele o této skutečnosti informuje. Příchodem zprávy FINISH se ruší záznamy dané transakce ve všech zúčastněných portálech.

Za předpokladu, že žadatel ve zprávě START uvede AVP typu ADDR_PRO_G, které obsahuje ip adresu poskytovatele, která není adresou portálu, přebírá portál funkci tranzitního portálu a předává zprávy od žadatele poskytovateli na dané adrese na spoji TCP. Pokud ve zprávě není uveden konkrétní port je volen port nastavený na portálu.

9.3 Zabezpečení databáze

Přístup do databáze je chráněn uživatelským jménem a heslem. Dále pak je možné povolit danému uživateli přístup pouze z požadované IP adresy. Aby byl zajištěn bezpečný přenos informací mezi ACP portálem a databází, je potřeba vytvořit a umístit do databáze certifikát. Tento certifikát se vytváří obdobně, jako certifikáty pro vzájemnou komunikaci portálů v kapitole 8.3.3. Opět je využit program OPENSSL. Pomocí nějž je vytvořen soukromý klíč a žádost o podepsání certifikátu certifikační autoritou, příkazem

```
req -newkey rsa:2048 -days 1000 -nodes -keyout
MYSQL_xdzure00_DP_key.pem -out MYSQL_xdzure00_DP_req.pem
```

Tímto byl vytvořen soukromý klíč a žádost, která se podepíše stejně jako v kapitole 8.3.3 zde vytvořenou CA, čímž je získán certifikát MYSQL_xdzure00_DP_cert.pem. Dále je potřeba vytvořit RSA soukromý klíč, což se provede příkazem

```
rsa -in MYSQL_xdzure00_DP_key.pem -out MYSQL_xdzure00_DP_key.pem
```

Aby databáze s klíčí mohla pracovat, musí vědět, kde se nacházejí. Z tohoto důvodu je potřeba do konfiguračního souboru uvést cestu. Konfigurační soubor má název my.ini (windows) nebo my.cnf (linux). Potřeba do něj zapsat

```
# The MySQL server
 [mysqld]
ssl=1
ssl-ca=C:/xampp/mysql/bin/etc/mysql/cert/CA_xdzure00_DP_cert.pem
ssl-cert=C:/xampp/mysql/bin/etc/mysql/cert/MYSQL_xdzure00_DP_cert.pem
ssl-key=C:/xampp/mysql/bin/etc/mysql/cert/MYSQL_xdzure00_DP_key.pem
```

Správnost konfigurace a tedy i aktivování SSL komunikace, je možné ověřit příkazem v databázi

```
mysql -u root -p
show variables like "%ssl%";
```

Výstup by pak měl odpovídat výstupu na Obr. 36.

```
mysql> show variables like "%ssl%";
+---------------+-----------------------------------------------------------+
| Variable_name | Value                                                     |
+---------------+-----------------------------------------------------------+
| have_openssl  | YES                                                       |
| have_ssl      | YES                                                       |
| ssl_ca        | C:/xampp/mysql/bin/etc/mysql/cert/CA_xdzure00_DP_cert.pem |
| ssl_capath    |                                                           |
| ssl_cert      | C:/xampp/mysql/bin/etc/mysql/cert/MYSQL_xdzure00_DP_cert.pem |
| ssl_cipher    |                                                           |
| ssl_key       | C:/xampp/mysql/bin/etc/mysql/cert/MYSQL_xdzure00_DP_key.pem |
+---------------+-----------------------------------------------------------+
```

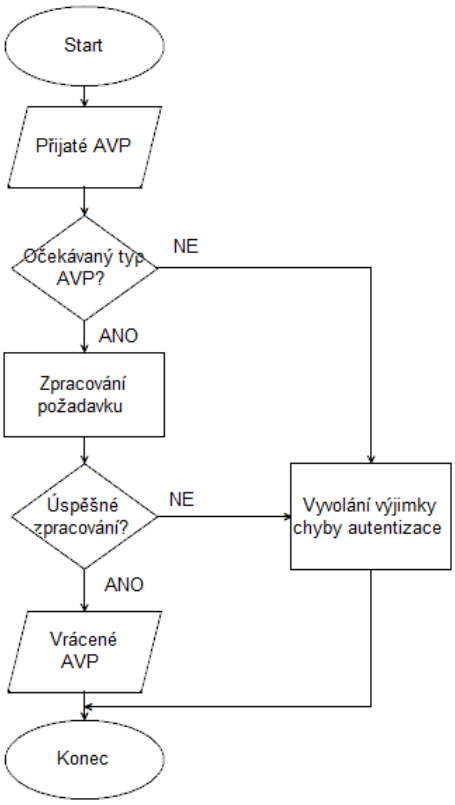
Obr. 36 Aktivace SSL v MYSQL databázi

9.4 Autentizační modul

Autentizační modul slouží k provedení samotné autentizace uživatele. Veškeré třídy realizující autentizační modul se nacházejí v balíčku `autentizacniModul` a implementují rozhraní `IAutentizace`. Autentizační modul je využíván správním modulem k ověření žadatele, přičemž typ způsobu ověření volí uživatel s dostupných autentizačních metod. Autentizační modul je stejně jako správní modul koncipován jako stavový automat viz. Obr. 37.

Komunikace s ním probíhá přes rozhraní definující metodu `zpracujData`, kde je vstupním parametrem AVP, určené ke zpracování a návratovou hodnotou je již zpracované AVP. Metoda `zpracujData` na základě typu přijatého AVP vykoná potřebnou proceduru, jako je např. generování náhodného čísla, ověření příchozího heše a podobně. Zpracované AVP je vráceno zpět správnímu modulu. Za předpokladu nežádoucího stavu, jako je zejména přijetí neočekávaného typu AVP, chybného ověření je v autentizačním modulu vyvolána výjimka, kterou odchytí správní modul a na jejím základě ukonči danou transakci. V současnosti je implementována autentizace typu výzva-odpověď. Kdy poskytovatel zašle žadateli náhodnou výzvu R. Žadatel příjme výzvu od poskytovatele, k této výzvě připojí své tajné heslo a následně na tento řetězec provede funkci hešování, pomocí hešovací funkce SHA-1. Tento heš zašle poskytovateli. Poskytovatel příjme zprávu a přes správní modul je odpověď od žadatele

předána autentizačnímu modulu, který provede stejnou operaci jako žadatel. Tzn. přidá heslo k náhodné výzvě a provede hešování SHA-1. Oba haše porovná a pokud se shodují je autentizace úspěšná. Zpráva s náhodným číslem, je přenášena v AVP typu INIT a očekávaná odpověď v AVP typu HMAC. Tento autentizační modul je realizován třídou `AutMet_VyzvaOdpovedSHAImp.class`



Obr. 37 Vývojový diagram funkcionality autentizačního modulu

Třída `AutMet_VyzvaOdpovedRSA.class` pak zajišťuje vyšší stupeň zabezpečení, kdy je vyžadována verifikace pinu (probíhá offline tzn. na kartě) a náhodné číslo výzvy je žadatelem podepisováno soukromým klíčem žadatele. To umožňuje sdílet veřejný klíč pro ověření uživatele s portálem provádějící autentizaci. Pin je přenášen žadateli v šifrované podobě, pomocí blokové šifry 3DES.

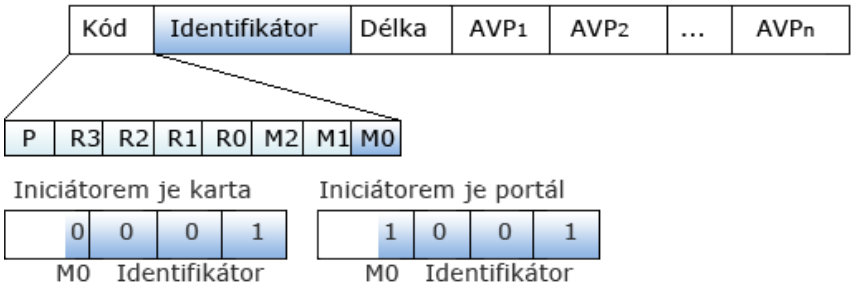
9.5 Jádro portálu

Jádro ACP portálu se nachází v balíčku `acp` a je realizováno třídou `AcpJadro.class`. ACP jádro se na rozdíl od jednotlivých modulů nachází v portálu pouze jedno a je přes něho vedena veškerá komunikace.

Jádro obsahuje ukazatel na komunikační modul, který slouží k aktivaci komunikačních rozhraní, která se aktivují při spuštění aplikace a jsou již schopna přijímat a odesílat data protější straně. Jádro se stará o vytváření nových spojů a zároveň uchovává záznamy o aktivních spojích pomocí vlastnosti (property) `zaznamySpojeni`, realizovaného Hash mapou, kde klíčem popisujícím daný spoj je

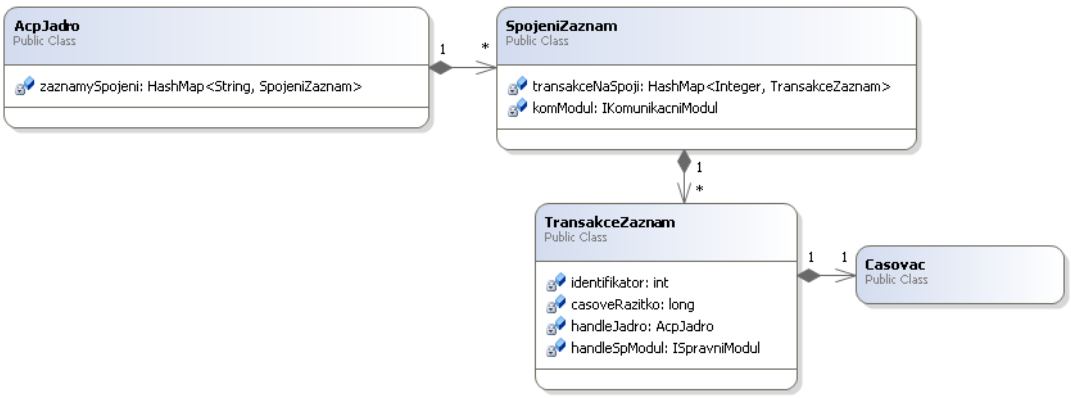
název vlákna realizující daný spoj. Hodnotou je pak záznam spojení popisující daný spoj, realizovaný třídou `SpojeniZaznam.class`.

Každý záznam o spojení obsahuje ukazatel na vlákno komunikačního modulu realizující daný spoj a záznamy o transakcích na daném spoji v podobě Hash mapy `transakceNaSpoji`. Klíčem je zde identifikátor transakce a hodnotou záznam o transakci. Aby bylo možné generovat identifikátor transakce nezávisle na hodnotách identifikátoru zvoleného na protější straně a korektně přiřadit zprávu transakci na spoji, byl zvolen způsob označení transakce identifikátorem složeného z bitu M0 v kódu zprávy a identifikátoru transakce, čímž dojde k jednoznačné specifikaci transakce na daném spoji viz Obr. 38.



Obr. 38 Jedinečný identifikátor transakce na spoji

Záznam o transakci pak obsahuje identifikátor transakce, časové razítko označující čas příchodu zprávy s aktualizací vždy s příchodem další zprávy této transakce, ukazatel na správní modul, pomocí kterého jsou předávány příchozí zprávy od jádra portálu a také ukazatel pro komunikaci s jádrem portálu. Dále pak každý záznam transakce obsahuje vnořenou třídu `Casovac`, která realizuje časovač, který je spuštěn příchodem zprávy START a nulován s příchodem následující správy patřící do stejné transakce. V případě vypršení časovače dojde k ukončení transakce a vymazání všech provozních informací spojených s touto transakcí. Časovač je nyní nastaven na dobu trvaní 10s, jelikož při autentizaci dochází k zapojení uživatele do procesu autentizace zadáním pinu. Výše popsaná hierarchie je zobrazena Obr. 39.



Obr. 39 Hierarchie záznamů obsažená v jádru portálu ACP

Jádro dále obsahuje ukazatel na uživatelské rozhraní, díky kterému je možné poskytovat zpětnou vazbu aplikace směrem k uživateli.

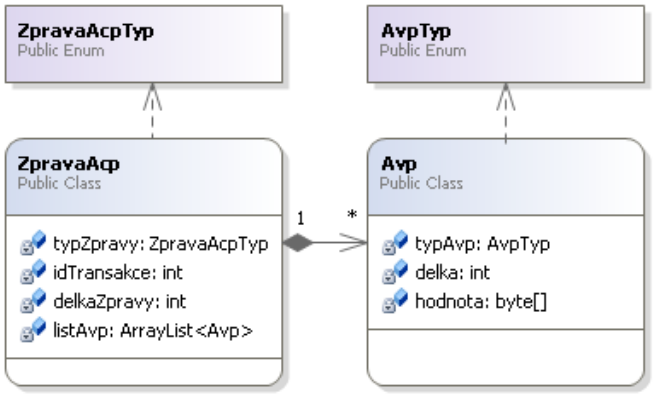
Jádro se stará, kromě udržování informací o otevřených spojích a transakcích na nich realizovaných, také o vytváření nových záznamů o spojení, v případě, že došla zpráva ze spoje, který není evidován v mapě spojů. Dále o přidávání nových záznamů transakce v případě, že byla přijata zpráva START a transakce v daném spoji neexistuje. V případě přijetí zprávy existující transakce, je jí tato zpráva předána, v opačném případě, na tuto zprávu terminál nereaguje. Dojde-li zpráva START s identifikátorem transakce, která již existuje, dojde k zrušení transakce, vymazáním záznamu transakce ze seznamu transakcí a ukončením časovače pro tuto transakci. Tímto smazáním dojde k smazání ukazatele na objekt záznamu transakce a tudíž se o smazání objektu z paměti a všech objektů sním spojených postará Garbage collector, čímž dochází ke smazání všech provozních informací souvisejících s touto transakcí. ACP portál pak tedy nereaguje na žádné další zprávy související s touto transakcí.

Pro komunikaci jádra a komunikačního modulu se používá metoda `odesliZpravuAcp`, která je využita správním modulem k předání ACP zprávy. Jádro na základě přijatého záznamu transakce zjistí, kterému spoji zprávu předat, aby došlo ke korektnímu odeslání zprávy žadateli, kterému zpráva náleží.

Je zde také definována metoda pro nastavení tranzitování zpráv mezi portály a také metoda pro načtení nastavení portálu z XML souboru, který je z důvodu bezpečnosti šifrován blokovou šifrou AES.

Dále se v balíku `acp` nachází třídy `ZpravaAcp` definující strukturu ACP zprávy dle specifikace, jako je typ zprávy, identifikátor transakce, délka zprávy a AVP obsažených v této zprávě. Třída se stará také o vytvoření struktury zprávy z příchozího proudu dat, pomocí metody `sestavStrukturu` a o opětovné sestavení proudu dat pro odeslání ze struktury. Za předpokladu, že dojde při sestavení struktury k chybě, např. nesouhlasí délka zprávy, nebo je zpráva kratší než 7B (povinná délka hlavičky), je vyhozena výjimka, kterou zachytává objekt, který o zpracování zprávy zažádal a na základě detekované výjimky zažádá jádro o zrušení transakce. `ZpravaAcpTyp` pak definuje enumeraci typů zpráv.

Třída `Avp` definuje strukturu AVP dle specifikace. Obsahuje typ AVP, délku a hodnotu, tzn. přenášená data. Definuje metodu pro vytvoření struktury AVP. `AvpTyp` pak definuje enumeraci všech možných typů AVP. Objekt zprávy a v ní obsažených AVP je zobrazena na Obr. 40.



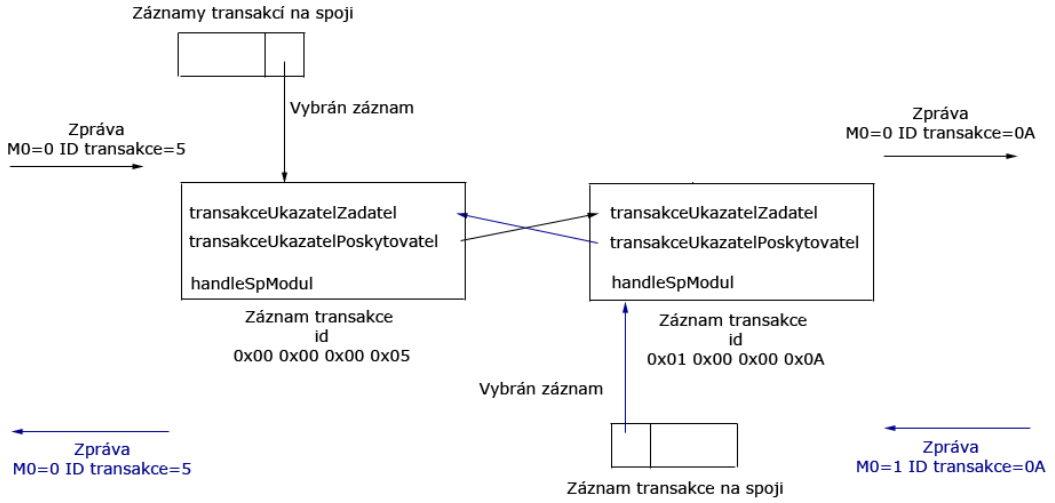
Obr. 40 Struktura objektu zprávy ACP a AVP

Třída `Util` obsahuje potřebné dodatečné metody související s obecným zpracováním potřebných procesů. Např. Převod části proudu dat na datový typ `integer`, využívaný kupříkladu u zjišťování z proudu dat identifikátor transakce nebo délku zprávy. Připojení APDU hlavičky k zprávě ACP pro korektní odeslání zprávy kartě apod.

`SpravceTransakce` je třída rozšiřující objekt `Thread` a tudíž realizuje samotné vlákno. Instance této třídy je vytvořena a následně spuštěna v jádru portálu, při příchodu nové zprávy od vzdáleného portálu. Vytvořený objekt pak slouží k obsluze správního modulu pro danou transakci.

9.5.1 Tranzitování

Jednou z vlastností navrženého portálu je také tranzitování zpráv mezi dvěma portály. Za předpokladu, že portál obdrží zprávu, která mu není určena, předá ji cílovému portálu viz Obr. 41. Princip tranzitování je takový, že portál vyhledá na základě hodnoty bitu M0 a identifikátoru zprávy příslušný záznam transakce v mapě záznamů pro daný spoj. Záznam transakce pak obsahuje křížový odkaz na záznam transakce pro následující transakci, kde portál vystupuje jako žadatel. Dále záznam obsahuje třídní proměnou `handleSpModul`, která je typu `ISpravniModul` a obsahuje správní modul, jehož úkolem je nahradit identifikátor zprávy identifikátorem záznamem transakce a následně tuto zprávu zaslat poskytovateli. Stejný postup pak probíhá, přichází-li odpověď od poskytovatele. Opět se přes křížový odkaz získá záznam transakce, nahradí se identifikátor ve zprávě a odešle se žadateli.



Obr. 41 Tranzitování transakcí

K tranzitování zpráv může dojít za předpokladu, že žadatel ve zprávě START uvede AVP typu ADDR_PRO_G, které obsahuje ip adresu poskytovatele, která není adresou portálu, pak portál pro tuto transakci přebírá funkci tranzitního portálu a předává zprávy od žadatele poskytovateli na dané adrese na spoji TCP. Pokud ve zprávě není uveden konkrétní port je volen port nastavený na portálu. Toto se provede tak, že správní modul volá přes ukazatel metodu `setTranzit` obsaženou ve třídě `AcpJadro`. Jako parametry je předána samotná zpráva, záznam transakce a adresa poskytovatele obsažená ve třídě `EndPoint`. Na základě těchto parametrů jádro vytvoří

nový záznam transakce, vygeneruje pro něj jedinečný identifikátor a klienta zajišťujícího přijímání a odesílání zpráv pro danou transakci na daném spoji. Nakonec jádro vytvoří mezi těmito dvěma záznamy křížové odkazy.

9.6 Uživatelské rozhraní

Uživatelské rozhraní, slouží pro komunikaci uživatele s portálem. Předává požadavky od uživatele jádru portálu a zpětně zobrazuje stavy portálu. Implementace je tvořena sadou tříd, obsažených v balíčku `acpGui`, přičemž vstupní třídou je třída `AcpGuiMain.class`. K programování uživatelského rozhraní, byla použita knihovna `swing`.

10 IMPLEMENTACE ACP PORTÁLU NA ČIPOVOU KARTU

K dispozici byla karta Convego Join 4.0/Sm@rtCafe Expert 4.0 od společnosti Giesecke&Devrient. Pro konpilaci byl použit JC konverter jcdk 3.0.1.

Na čipovou kartu byl vytvořen applet, obsahující zjednodušenou verzi ACP portálu. Jednak z důvodu, že některé funkce ACP portálu nejsou potřeba, např. tranzitování komunikace. A jednak je to dáno omezenými možnostmi čipové karty a samotné technologie Java Card.

Applet reprezentuje ACP portál žadatele a jeho hlavním úkolem je vygenerovat zprávu START, rozpoznat příchozí zprávy protokolu ACP a reagovat na ně. `Applet` se stává z jádra, reprezentovaného třídou `AcpAplet` a z příslušných modulů, reprezentovaných stejnojmennými třídami, stejně jako tomu bylo u návrhu uživatelské aplikace. Komunikační modul zajišťuje komunikaci mezi kartou a čtecím zařízením pomocí APDU zpráv. Správní modul zajišťuje přístupovou politiku, přijímá a odesílá zprávy ACP, obsahuje přístupová práva uživatele. Autentizační modul zajišťuje provedení samotné autentizace. Program je jednovláknový, jelikož Java Card nemá podporu pro tvorbu vícevláknových aplikací.

10.1 AcpAplet

Třída `AcpAplet` rozšiřuje třídu `Applet` a jedná se o ústřední bod celé aplikace. Reprezentuje jádro portálu žadatele. Přijímá požadavky od JCRE a reaguje na ně.

Jeli applet vybrán, tj. přijde-li příkaz APDU s požadavkem o výběr appletu s daným AID do JCRE, vytvoří se správní modul, zajišťující korektního průběhu transakce, tj. zahájení transakce, sjednávání požadovaných aktiv, sjednání autentizační metody apod. Jakmile se správní modul vytvoří, ihned sestaví a odešle zprávu START, dle specifikace protokolu ACP.

Třída dále kontroluje hlavičku APDU příchozí zprávy, na jejímž základě se rozhodne, zda a jakým způsobem s ADPU bude zacházet. Nyní je pro ACP protokol definováno `CLA=0x80` a `INS=0x00`. Jeli obdržen APDU příkaz s výše uvedenými parametry, dochází k zpracovávání APDU příkazu, předáním zprávy komunikačnímu modulu, který příchozí zprávu dále zpracuje.

Ve třídě jsou dále obsaženy metody potřebné k vytvoření záznamů o jednotlivých transakcích, zrušení transakcí, metody pro předávání zpráv od správního modulu příslušnému komunikačnímu modulu, za účelem odeslání a naopak z komunikačního modulu předává zprávy příslušnému správnímu modulu.

10.2 Komunikační modul

Jelikož Java Card nepodporuje více vláken, není možné, aby komunikační modul pracoval nezávisle a jakmile přijde nová zpráva APDU automaticky ji předal jádru ke zpracování. Jelikož však komunikace s kartou probíhá typem výzva-odpověď, a tudíž příchozí zpráva může přijít teprve až po odeslání odpovědi na předešlou zprávu, je problém nepodstatný.

Hlavním úkolem komunikačního modulu je zkontrolovat, zda při přenosu APDU zprávy nedošlo k chybě, načíst přenášená data tj. zprávu ACP a předat ji jádru portálu ke zpracování. Dále pak je zde implementována metoda pro odeslání APDU odpovědi.

10.3 Správní modul

Tato třída se zabývá samotným řízením průběhu komunikace. Je založena na principu stavového automatu, jehož stav závisí na typu přijaté zprávy. Obsahuje metody pro vytvoření zprávy START (vytvoří se při výběru) a následně ji odešle. Dále se pak jedná o metodu `prislaZprava`, kdy v závislosti na typu zprávy se vykoná příslušná operace např. REQUEST, kdy je předána výzva od poskytovatele aktiva autentizačnímu modulu, který tuto výzvu zpracuje a vrátí příslušnou odpověď.

10.4 Autentizační modul

Tato třída se stará o průběh samotné autentizace. Přijímá parametry od správního modulu a vrací zpracovaný výsledek autentizace. V případě REQUEST, kdy poskytovatel aktiva zašle náhodné číslo, autentizační modul vypočte heš SHA1 z tohoto čísla a hesla žadatele. Tento heš vrátí správnímu modulu, který sestaví odpovídající zprávu RESPONSE a zašle ji poskytovateli. Pokud obdrží typ AVP který není schopný zpracovat, dojde ke zrušení transakce.

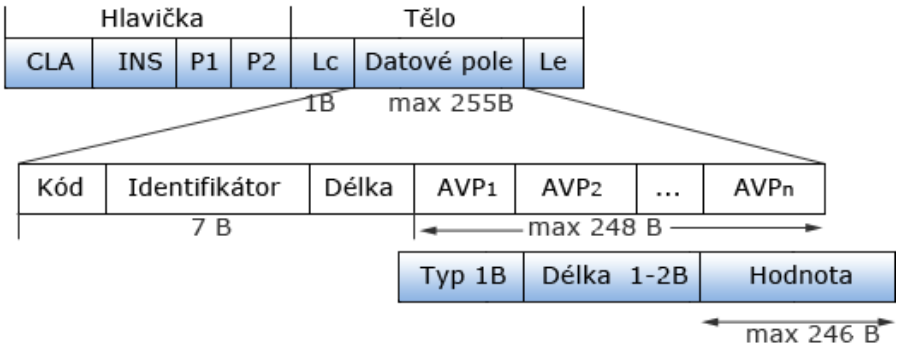
Další autetizační modul (třída `AutentizacniModulRSA`), souvisí s druhým scénářem, který vyžaduje verifikaci pinu. Za tímto účelem bylo využito instance třídy `OwnerPIN`, která je zodpovědná za ověření správnosti pinu a případné snížení počtu pokusů. Jako vstupní parametr se používá dešifrovaný heš pinu obdržený v AVP typu 3DES.

10.5 Další třídy appletu

Třída **`ZprávaAcp`** slouží k reprezentaci zprávy protokolu ACP, obsahuje typ zprávy, identifikátor transakce, délku zprávy a jednotlivá AVP. Jejím úkolem je z příchozího řetězce bajtů sestavit zprávu dle specifikace ACP a naopak ze zprávy vytvořit řetězec bajtů pro odeslání. Jelikož Java Card platforma nepodporuje enumerace, jsou jednotlivé typy zprávy definovány ve třídě **`ZpravaAcpTyp`**, kde je vyjádřena jejich bajtová hodnota.

Z důvodu použití technologie Java Card, která používá pro přenos APDU jednotky, jenž mohou přenášet data o maximální délce 255B (tj. jeden bajt pro Lc). Je možné

v této zprávě přenášet zprávy ACP o maximální délce 255B tzn. přenášená data mohou mít velikost maximálně 248B. Z tohoto důvodu je toto potřeba brát na zřetel a tedy posílat na kartu zprávy s maximální délkou přenášených dat 246B viz. Obr. 42. Pokud je obdržena zpráva se špatnou délkou, je zrušena příslušná transakce a karta vrátí příslušný návratový kód.



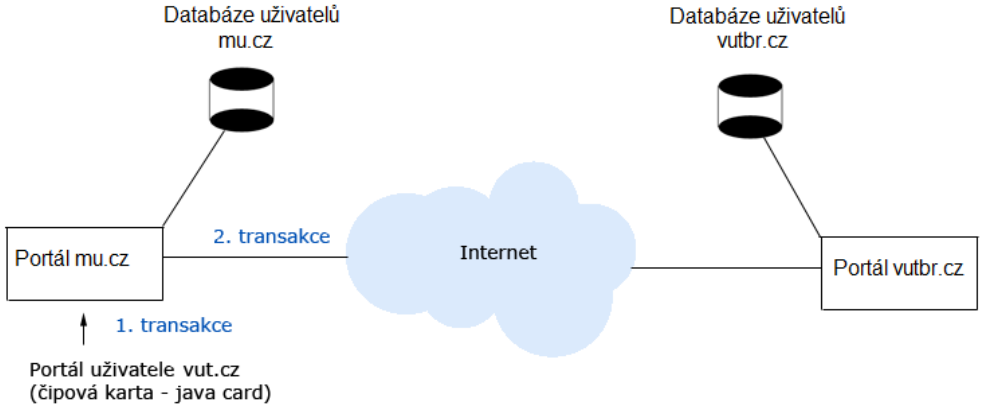
Obr. 42 Maximální délka ACP zprávy

Obdobně třída **`Avp`**, reprezentuje jedno konkrétní AVP. Jeho typ, délku a hodnotu (přenášená data). Pokud zpráva obsahuje AVP, které karta není schopna zpracovat, dojde ke zrušení transakce a poté vrátí návratovou hodnotu `SW =0x9605`. Všechny typy návratových hodnot jsou obsaženy ve třídě **`NavratoveKody`**.

Třída **`Transakce`** slouží k vedení vlastností dané transakce. Obsahuje Identifikátor transakce, který ji jednoznačně specifikuje, ukazatel na správní modul, který řídí danou komunikaci a ukazatel na komunikační modul, který zajišťuje správné odeslání a přijetí zpráv od protistrany. Dále je zde obsažen objekt zprávy, reprezentující zprávu související s touto transakcí. Z důvodu absence vícevláknového programování či integrovaného časovače, nebylo možné implementovat časovač jednotlivých transakcí. K smazání transakce dochází tedy při přijetí zprávy FINISH, případně při opětovném výběru appletu.

11 TESTOVÁNÍ NAVRŽENÉHO SYSTÉMU

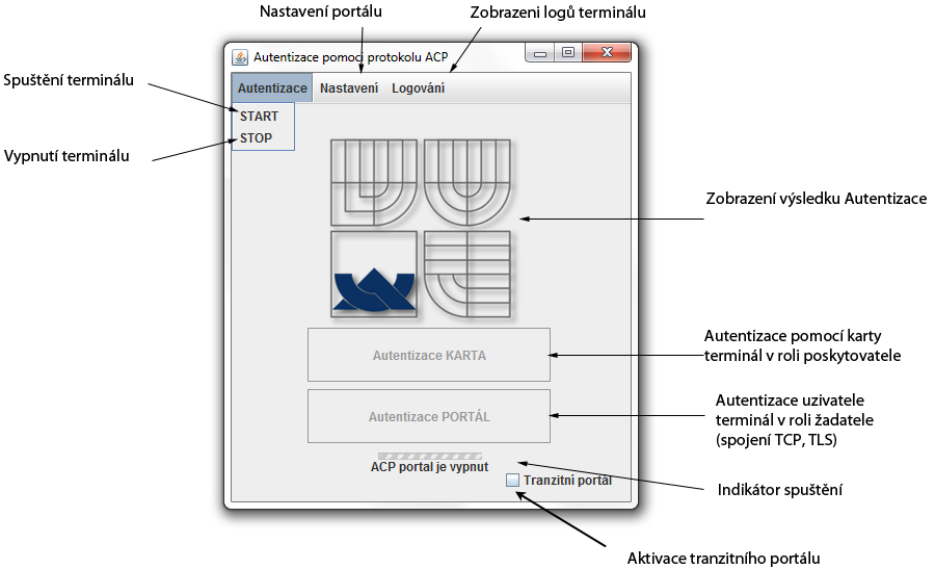
Tato kapitola se zabývá, samotným testováním a bezpečností navrženého systému autentizace. Testován je scénář, kdy uživatel z cizí organizace např. Vysokého učení technického (vutbr.cz) žádá o přístup k určitému aktivu např. otevření dveří v lokální organizaci Masarykovy univerzity (mu.cz). Uživatel organizace vutbr.cz disponuje kartou java Card, na které je nainstalován applet s portálem žadatele, který byl v této práci vytvořen (xdzure00_DP_ACP_Aplet). V jednotlivých organizacích jsou spuštěny ACP portály poskytovatele (xdzure00_DP_ACP_Terminal). Portály mají připojení k SQL databázi uživatelů, nacházející se přímo v daném zařízení, v lokální síti, nebo se také může jednat o vzdáleny server na kterém je databáze nainstalována. Obr. 43 popisuje výše popsané schéma komunikace.



Obr. 43 Testovací scénář navrženého autentizačního systému

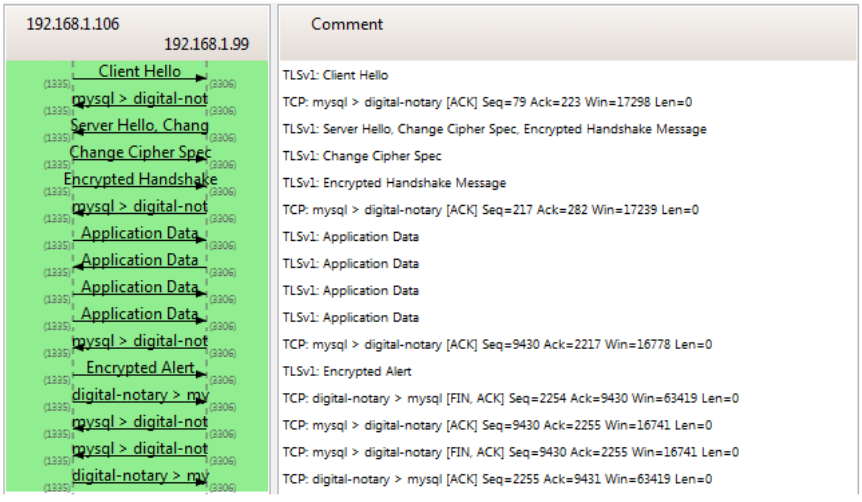
Portály disponují uživatelským rozhraním, pomocí kterého je možné portál konfigurovat, sledovat průběhy komunikace a hlavně také umožňuje interakci uživatele a přístupového systému. ACP portál poskytovatele (dále terminál) je kompilován do spustitelného souboru *acp_uziv_aplikace.jar*. Spuštění aplikace je možné jednak přímo, nebo pomocí příkazového řádku, příkazem `java -jar acp_uziv_aplikace.jar`, kdy je možné sledovat kompletní chování terminálu. Výchozí okno aplikace je zobrazeno na Obr. 44.

Konfigurace terminálu je možná přes položku menu Nastavení. Toto nastavení obsahuje parametry potřebné pro korektní běh portálu. Nastavují se zde čísla portů pro naslouchání na spojích TCP a TLS, uvádí se název lokální organizace, na jejímž základě se zjišťuje zda žadatel patří do této organizace, či nikoli. Dále se zde uvádí cesta k certifikátům portálu a důvěryhodných autorit. Tyto certifikáty musí být bezpečně uloženy v úložišti certifikátů, to je v Javě realizováno pomocí tzv. keystore (*.jks). Přístup k tomuto úložišti je chráněn heslem (zde je heslo *VxDPk12*).



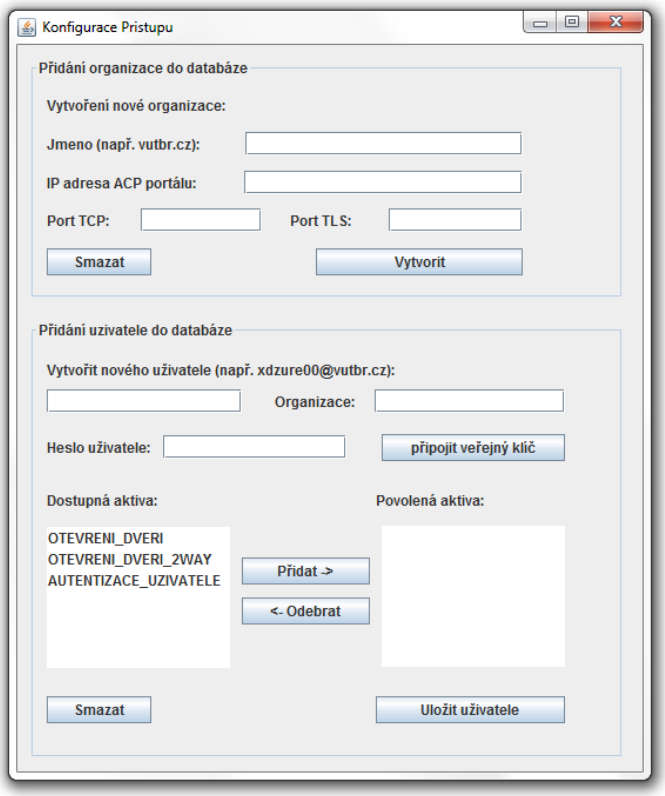
Obr. 44 ACP portál poskytovatele (terminál)

V nastavení je dále potřeba uvést adresu k SQL databázi a název databáze obsahující informace o uživatelích a organizacích zapojených do systému autentizace. Je potřeba také vyplnit uživatelské jméno a heslo pro přístup do databáze. Pomocí tlačítka konektivita, je možné ověřit spojení s databází. Přístup do databáze je zabezpečen heslem. Standardně je spojení s databází nezabezpečené a hrozí odposlechnutí citlivých informací získávaných z databáze. Z tohoto důvodu vyžaduje terminál zabezpečené spojení pomocí protokolu TLS. Terminál vlastní certifikát, pomocí kterého je schopen ověřit předkládaný certifikát SQL serverem. Pokud je správný, dojde k sestavení spojení a případnému načítání dat z databáze. V opačném případě spojení není sestaveno a uživatel je upozorněn na nedůvěryhodnost certifikátu. Na Obr. 45 je vidět průběh komunikace terminálu a MYSQL databáze, zachycené pomocí programu *Wireshark*, od samotného začátku inicializace spojení zahájené na straně klienta zprávou Client Hello (terminál - ip adresa 192.168.1.106) přes šifrovaný přenos až k ukončení komunikace.



Obr. 45 Komunikace terminálu s databází uživatelů

Dále je v nastavení možné přidat novou organizaci, která by se zapojila do procesu systému autentizace a samotného uživatele. Povinné hodnoty organizace jsou název organizace, ip adresa ACP portálu, s kterým bude lokální portál komunikovat a čísla portů na kterých portál naslouchá. U uživatele je pak potřeba vyplnit jméno např. ve formě emailové adresy nebo telefonního čísla a také organizaci ze které pochází, aby jej bylo možné autentizovat. Dále je možné uživateli přiřadit heslo, veřejný klíč a aktiva se kterými smí nakládat viz Obr. 46.

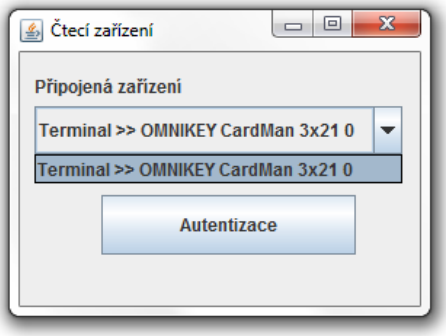


Obr. 46 Vytváření jednotlivých entit do systému autentizace

Z výše uvedeného je patrné, že velkou míru bezpečnosti terminálu nese právě nastavení. Přistup ke konfiguraci je proto chráněn heslem, které uživatel musí zadat (zde 1234). Z tohoto hesla je vytvořen heš SHA-1, který je porovnán s hešem uloženým v terminálu. Samotné nastavení je uloženo v souboru XML, konkrétně nastaveniAES.xml. Aby nebylo možné tento soubor nijak modifikovat a tím narušit bezpečnost celého systému, je tento soubor zašifrován blokovou šifrou AES s délkou klíče 128 bitů. Klíč je vytvořen pomocí hešovací funkce SHA-256, z hesla terminálu, přičemž prvních 16 bajtů je použito jako klíč pro algoritmu AES. Inicializační vektor je vytvářen náhodně a umísťován na začátek bitového streamu.

Jeli-portál správně nakonfigurován, je možné jej spustit pomocí položky Start a následně provést samotnou autentizaci uživatele. Terminál se spustí, zinicializuje a poté je připraven k použití.

Pomocí tlačítka Autentizace karta, se zobrazí okno s připojenými terminály, kde se vybere konkrétní terminál, s přístupovou kartu viz Obr. 47.



Obr. 47 Terminál Autentizace pomocí čipové karty

Tlačítkem autentizace je vybrán applet na kartě, jenž má AID 78 64 7A 75 72 12 34 56, ten odpovídá portálu žadatele. Struktura APDU příkazu pro výběr tohoto appletu je zobrazena na Obr. 48.

Hlavička				Tělo		
CLA	INS	P1	P2	Lc	Datové pole	Le
00	A4	04	00	08	78647A7572123456	00

Obr. 48 Výběr appletu na kartě

Jakmile je applet vybrán, spustí se proces autentizace. Karta zašle terminálu zprávu START, kde uvádí své jméno NAME_SUP_G, které je *xdzure00@vutbr.cz*, volbu aktiva ASSET_L, čímž je požadavek na otevření dveří a také uvádí autentizační metodu LAM, která je konkrétně typu výzva odpověď, kdy terminál zasílá náhodné číslo, žadatel toto číslo podepíše svým soukromým klíčem a zašle zpět.

Po přijetí zprávy terminálem, ji terminál zkontroluje, zda obsahuje všechny náležité informace. Pokud ano, zahájí se proces autentizace. Vyžádá si od uživatele pin, který spolu s náhodným číslem (AVP typu INIT zašle kartě). Aby byl zajištěn bezpečný přenos pinu, je pin šifrován. Ze zadaného pinu je vytvořen heš SHA-1. Tento heš měl být původně uložen do AVP typu AES. Jelikož však disponovaná karta šifrování AES nepodporovala, bylo jako alternativa k němu vytvořeno AVP typu 3DES, které nese data zašifrovaná šifrou 3DES v režimu CBC pomocí třech klíčů o délce 56 bitů. Pro toto AVP byl vybrán typ 176, který spadá do rezervovaných typů LAVP. Toto AVP je přenášeno v kontejnérovém AVP typu ENC spolu s inicializačním vektorem (AVP typu INIT). Jednotlivé klíče jsou vypočteny z heše pinu. Heš má délku 20B, první klíč 1-8B, druhý klíč 6-14B a třetí klíč 12-20B. Terminál poté zašle tato AVP ve zprávě REQUEST na kartu. Tímto způsobem je zajištěno, že nedojde k odposlechnutí pinu, během komunikace.

Pin se v procesu autentizace používá pro zvýšení stupně bezpečnosti. Je zde vytvořena dvoufaktorová autentizace, kdy se žadatel prokazuje jednak kartou (tokenem) a jednak znalostí (pinem).

Po příjmu zprávy karta dešifruje data v AVP 3DES. Je na ní uložen heš pinu, a následně porovná tyto dva heše, shodují-li se, znamená to, že s kartou manipuluje oprávněná osoba. Z AVP INIT zjistí hodnotu náhodného čísla výzvy a pomocí svého soukromého klíče tuto výzvu podepíše a výsledek zašle zpět portálu v AVP HMAC.

Neproběhla-li validace pinu úspěšně, dojde ke sníží počtu pokusů o jeden a karta dále již nereaguje.

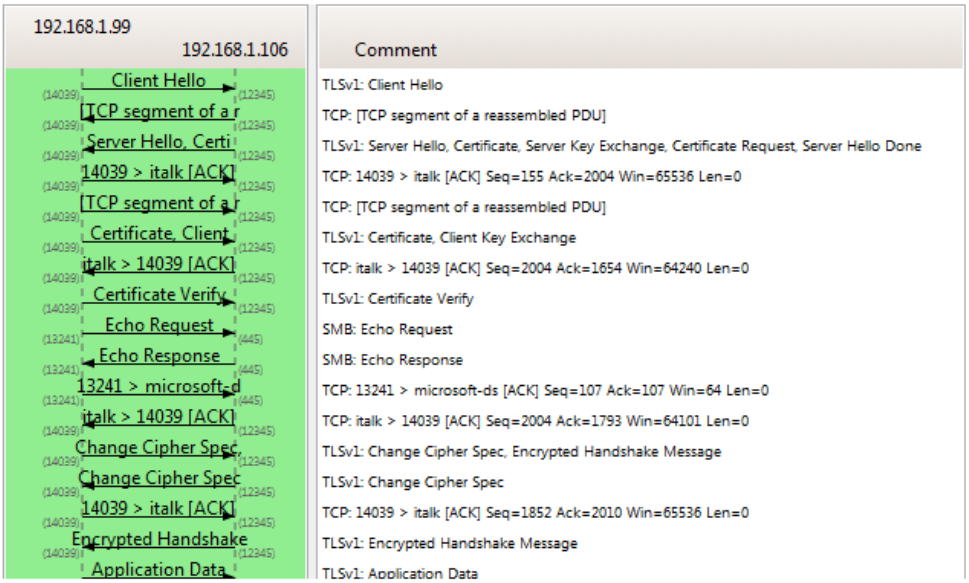
Pin by teoreticky nemusel být hešován, jelikož je přenášen šifrovaně. Při ověřování pinu by však bylo možné zjistit, na jaké pozici pinu došlo k chybě ověření, což by dalo útočníkovy užitečnou informaci. Při použití heše však útočník není schopen zjistit, kde došlo k chybě.

Přenášené náhodné číslo podepsané soukromým klíčem karty není nijak šifrováno. Je pouze podepsáno a tím je splněn požadavek pro jednoznačnou identifikaci uživatele tím, že je opravdu držitelem soukromého klíče.

Jelikož uživatel nespadá do místní organizace a terminál nemá přístup k veřejnému klíči uživatele, není schopen jej autentizovat. Z databáze uživatelů však ví, že uživatel spadá do organizace *vutbr.cz*. Načte si potřebné informace pro spojení tj. ip adresu portálu vutbr.cz a číslo portu TLS. Vytvoří zprávu START, do které vloží AVP typu NAME_SUP_G tj. název portálu *xdzure00_Terminal_vutbr.cz*, NAME_ENT_G specifikující jméno autentizovaného *xdzure00@vutbr.cz*, ASSET_G výběr aktiva tj. ověření uživatele, INIT náhodné číslo výzvy, HMAC podepsané náhodné číslo výzvy kartou.

Následně je vytvořeno nové zabezpečené spojení s portálem vutbr.cz a vytvořena nová transakce a zpráva START je odeslána. Portál vutbr.cz příjme zprávu a přidělí ji nové transakci. Zkontroluje zda obsahuje všechny potřebné parametry a následně zahájí ověření uživatele. Jelikož vlastní veřejný klíč držitele karty zkontroluje podpis a výsledek autentizace odešle zpět portálu mu.cz.

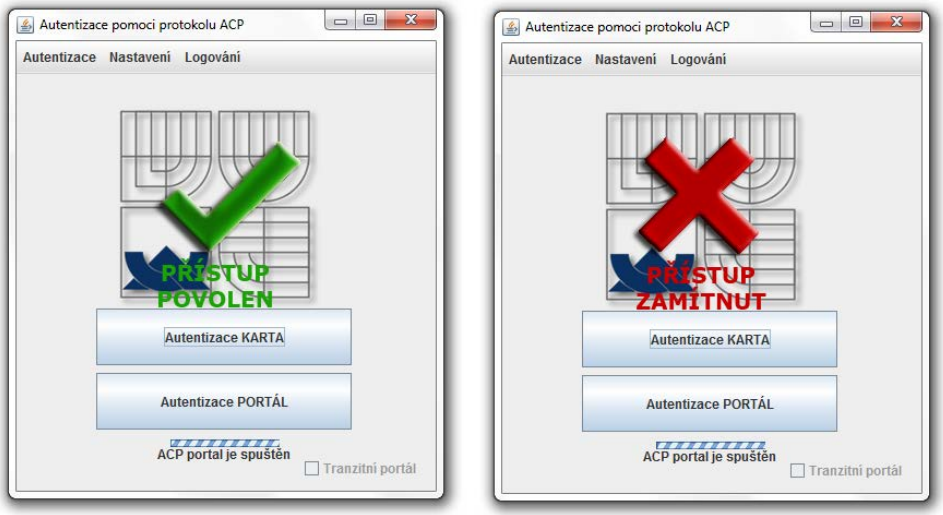
Jelikož je spojení vytvořeno pomocí zabezpečeného spoje TLS, kdy jsou obě strany vzájemně autentizovány, neprobíhá zde fáze autentizace. V případě aktiva autentizace uživatele je vždy vyžadována zpráva START se všemi potřebnými parametry a proto neprobíhá ani fáze sjednávání parametrů. Dochází tedy pouze k výměně zpráv START a FINISH. Na Obr. 49. je zobrazeno spojení mezi oběma ACP portály, zachycené programem Wireshark.



Obr. 49 Zabezpečené spojení mezi ACP portály

Na výše uvedeném obrázku je opět vidět průběh sestavení spojení. Oproti sestavování spojení s databází, je zde oboustranná autentizace (2WAY authentication), tzn. svoji identitu prokazují obě strany tzn. certifikát předkládá jak klient tak server. Na obrázku ACP portál vutbr.cz reprezentuje IP adresa 192.168.1.99 a portál mu.cz IP adresa 192.168.1.106.

Na základě příchozí zprávy FINISH, pak portál umožní, popřípadě zamítne, přístup uživateli k jemu zvolenému aktivu. Dochází k zrušení spojení mezi portály a také zrušení transakce. Terminál o výsledku autentizace upozorní uživatele skrz uživatelské rozhraní viz. Obr. 50.



Obr. 50 Zobrazení výsledku autentizace

Celý výše popsaný průběh autentizačního schématu, je také možné pozorovat v logu Terminálu. Jsou zde obsaženy informace o stavech portálu, o příchozích a odchozích zprávách a v nich obsažených AVP. Logy lze filtrovat na základě typu a dosáhnout tím lepší přehlednosti, viz. Obr. 51.

Index	Čas ▲	Urov...	Typ	Informace
1	7:25:38	INFO	PORTAL_INFO	Načtení uživatelského rozhraní
7	7:25:41	INFO	PORTAL_INFO	Připojená karta: PC/SC card in OMNIKEY CardMan 3x21 0, protocol T=1, state OK
6	7:25:41	INFO	PORTAL_INFO	Pripojen terminal: PC/SC terminal OMNIKEY CardMan 3x21 0
5	7:25:41	INFO	PORTAL_INFO	Aktivace posluchače TLS na čísle portu 43215
4	7:25:41	INFO	PORTAL_INFO	Aktivace posluchače TCP na čísle portu 12345
3	7:25:41	INFO	PORTAL_INFO	Rozhraní ISO7816, Kontrola připojených terminálů
2	7:25:41	INFO	PORTAL_INFO	ACP portál spuštěn
9	7:25:45	INFO	DATA_DO	[128, 0, 0, 1, 0, 0, 32, 0, 17, 120, 100, 122, 117, 114, 101, 48, 48, 64, 118, 117, 116, 98, 114, 46, 99, 122, 37, 1, 1, 33, 1, 0]
8	7:25:45	INFO	DATA_Z	[120, 100, 122, 117, 114, 18, 52, 86]
12	7:25:45	INFO	TRANSAKCE	Vytvořen nový záznam transakce s id =1
11	7:25:45	INFO	PORTAL_INFO	Vytvořen nový spojThread[Terminal >> OMNIKEY CardMan 3x21 0,6,main]
10	7:25:45	INFO	ACP_PRIJEM	Zprava = START, ID transakce = 1, obsažená AVP = NAME_SUP_G ASSET_L LAM
13	7:25:51	INFO	DATA_Z	[133, 0, 0, 1, 0, 0, 53, 48, 4, 40, 247, 212, 226, 224, 0, 37, 48, 8, 76, 72, 55, 200, 7, 12, 178, 144, 176, 0, 24, 15, 208, 57, 207, ...
16	7:25:53	INFO	ACP_ODESLANI	Zprava = REQUEST, ID transakce = 1, obsažená AVP = INIT ENC
15	7:25:53	INFO	ACP_PRIJEM	Zprava = RESPONSE, ID transakce = 1, obsažená AVP = HMAC
14	7:25:53	INFO	DATA_DO	[130, 0, 0, 1, 0, 0, 137, 50, 128, 37, 36, 66, 61, 208, 103, 83, 21, 68, 122, 231, 15, 122, 207, 179, 210, 227, 8, 4, 219, 134, 146...
20	7:25:54	INFO	DATA_Z	[135, 0, 0, 1, 0, 0, 10, 38, 1, 1]
19	7:25:54	INFO	TRANSAKCE	Vytvořen nový záznam transakce s id =30985334
18	7:25:54	INFO	PORTAL_INFO	Vytvořen nový spoj192.168.1.106
17	7:25:54	INFO	PORTAL_INFO	Obdržený certifikát: CN=www.xdzure00.cz, OU=FEKT, O=VUT, L=Brno, ST=Ceska Republika, C=CZEMAILADDRESS=xdzure...
21	7:25:55	INFO	ACP_ODESLANI	Zprava = FINISH, ID transakce = 1, obsažená AVP = RESULT
22	7:25:59	INFO	PORTAL_INFO	Smazán spoj = Thread[Terminal >> OMNIKEY CardMan 3x21 0,6,main]
25	7:26:0	INFO	ACP_ODESLANI	Zprava = START, ID transakce = 14208118, obsažená AVP = NAME_SUP_G NAME_ENT_G ASSET_G LAM INIT HMAC
24	7:26:0	INFO	DATA_Z	[128, 216, 204, 118, 0, 0, 193, 0, 23, 120, 100, 122, 117, 114, 101, 48, 48, 95, 84, 101, 114, 109, 105, 110, 97, 108, 95, 109, ...
23	7:26:0	INFO	TRANSAKCE	Smazán záznam transakce s id =1
29	7:26:1	INFO	TRANSAKCE	Smazán záznam transakce s id =30985334
28	7:26:1	INFO	PORTAL_INFO	Smazán spoj = 192.168.1.106
27	7:26:1	INFO	ACP_PRIJEM	Zprava = FINISH, ID transakce = 14208118, obsažená AVP = RESULT
26	7:26:1	INFO	DATA_DO	[135, 216, 204, 118, 0, 0, 10, 38, 1, 0]

Obr. 51 Logy terminálu (průběh komunikace)

Dále byl testován 1. scénář z kapitoly 8.1, kdy se uživatel prokazuje znalostí hesla. Ve zprávě START nejsou uvedena všechna potřebná AVP a dochází tedy k procesu sjednávání parametrů transakce. Následně je zahájen proces autentizace. Testován byl případ, kdy uživatel použil nesprávné heslo. Průběh autentizace je zobrazen na Obr. 52.

Index	Čas	Uroveň	Typ	Informace
1	7:44:2	INFO	PORTAL_INFO	Načtení uživatelského rozhraní
2	7:44:6	INFO	PORTAL_INFO	ACP portál spuštěn
3	7:44:6	INFO	PORTAL_INFO	Rozhraní ISO7816, Kontrola připojených terminálů
4	7:44:6	INFO	PORTAL_INFO	Aktivace posluchače TCP na čísle portu 12345
5	7:44:7	INFO	PORTAL_INFO	Aktivace posluchače TLS na čísle portu 43215
6	7:44:35	INFO	PORTAL_INFO	Vytvořen nový spoj192.168.1.99
7	7:44:35	INFO	TRANSAKCE	Vytvořen nový záznam transakce s id =31656741
8	7:44:35	INFO	DATA_Z	[128, 227, 11, 37, 0, 0, 26, 0, 17, 120, 100, 122, 117, 114, 101, 48, 48, 64, 118, 117, 116, 98, 114, 46, 99, 122]
9	7:44:35	INFO	ACP_ODESLANI	Zprava = START, ID transakce = 14879525, obsažená AVP = NAME_SUP_G
10	7:44:36	INFO	DATA_DO	[129, 227, 11, 37, 0, 0, 9, 0, 0]
11	7:44:36	INFO	ACP_PRIJEM	Zprava = OFFER, ID transakce = 14879525, obsažená AVP = NAME_SUP_G
12	7:44:36	INFO	DATA_Z	[134, 227, 11, 37, 0, 0, 26, 0, 17, 120, 100, 122, 117, 114, 101, 48, 48, 64, 118, 117, 116, 98, 114, 46, 99, 122]
13	7:44:36	INFO	ACP_ODESLANI	Zprava = SPECIFICATION, ID transakce = 14879525, obsažená AVP = NAME_SUP_G
14	7:44:36	INFO	DATA_DO	[129, 227, 11, 37, 0, 0, 16, 37, 1, 0, 37, 1, 1, 37, 1, 2]
15	7:44:36	INFO	ACP_PRIJEM	Zprava = OFFER, ID transakce = 14879525, obsažená AVP = ASSET_L ASSET_L ASSET_L
16	7:44:36	INFO	DATA_Z	[134, 227, 11, 37, 0, 0, 10, 37, 1, 2]
17	7:44:36	INFO	ACP_ODESLANI	Zprava = SPECIFICATION, ID transakce = 14879525, obsažená AVP = ASSET_L
18	7:44:37	INFO	DATA_DO	[129, 227, 11, 37, 0, 0, 13, 33, 1, 0, 33, 1, 1]
19	7:44:37	INFO	ACP_PRIJEM	Zprava = OFFER, ID transakce = 14879525, obsažená AVP = LAM LAM
20	7:44:37	INFO	DATA_Z	[134, 227, 11, 37, 0, 0, 10, 33, 1, 1]
21	7:44:37	INFO	ACP_ODESLANI	Zprava = SPECIFICATION, ID transakce = 14879525, obsažená AVP = LAM
22	7:44:38	INFO	DATA_DO	[133, 227, 11, 37, 0, 0, 13, 48, 4, 4, 83, 133, 30]
23	7:44:38	INFO	ACP_PRIJEM	Zprava = REQUEST, ID transakce = 14879525, obsažená AVP = INIT
24	7:44:38	INFO	DATA_Z	[130, 227, 11, 37, 0, 0, 29, 50, 20, 159, 92, 253, 58, 170, 52, 225, 78, 122, 68, 97, 249, 109, 249, 59, 142, 160, 219, 217, 19]
25	7:44:38	INFO	ACP_ODESLANI	Zprava = RESPONSE, ID transakce = 14879525, obsažená AVP = HMAC
26	7:44:38	INFO	DATA_DO	[135, 227, 11, 37, 0, 0, 10, 38, 1, 2]
27	7:44:38	INFO	ACP_PRIJEM	Zprava = FINISH, ID transakce = 14879525, obsažená AVP = RESULT
28	7:44:38	INFO	PORTAL_INFO	Smazán spoj = 192.168.1.99
29	7:44:38	INFO	TRANSAKCE	Smazán záznam transakce s id =31656741

Obr. 52 Logy terminálu prvního scénáře (chybná autentizace)

Na Obr. 53 je pak zobrazeno chování portálu, které je v tranzitním režimu a předává tak zprávy od jednoho portálu ke druhému, čímž plní pouze roli prostředníka.

Index	Čas ▲	Uroveň	Typ	Informace
1	7:59:22	INFO	PORTAL_INFO	Načtení uživatelského rozhraní
2	7:59:39	INFO	PORTAL_INFO	ACP portál spuštěn
3	7:59:39	INFO	PORTAL_INFO	Rozhraní ISO7816, Kontrola připojených terminálů
4	7:59:39	INFO	PORTAL_INFO	Pripojen terminal: PC/SC terminal OMNIKEY CardMan 3x21 0
5	7:59:39	INFO	PORTAL_INFO	Aktivace posluchače TCP na čísle portu 12345
6	7:59:39	INFO	PORTAL_INFO	Aktivace posluchače TLS na čísle portu 43215
7	7:59:39	INFO	PORTAL_INFO	Připojená karta: PC/SC card in OMNIKEY CardMan 3x21 0, protocol T=1, state OK
8	7:59:45	INFO	DATA_Z	[120, 100, 122, 117, 114, 18, 52, 86]
9	7:59:46	INFO	DATA_DO	[128, 0, 0, 1, 0, 0, 32, 0, 17, 120, 100, 122, 117, 114, 101, 48, 48, 64, 118, 117, 116, 98, 114, 46, 99, 122, 37, 1, 1, 33, 1, 0]
10	7:59:46	INFO	ACP_PRIJEM	Zprava = START, ID transakce = 1, obsažená AVP = NAME_SUP_G ASSET_L LAM
11	7:59:46	INFO	PORTAL_INFO	Vytvořen nový spojThread[Terminal >> OMNIKEY CardMan 3x21 0,6,main]
12	7:59:46	INFO	TRANSAKCE	Vytvořen nový záznam transakce s id =1
13	7:59:46	INFO	PORTAL_INFO	Vytvořen nový spoj192.168.1.106
14	7:59:46	INFO	TRANSAKCE	Vytvořen nový záznam transakce s id =32997997
15	7:59:46	INFO	DATA_Z	[128, 247, 130, 109, 0, 0, 32, 0, 17, 120, 100, 122, 117, 114, 101, 48, 48, 64, 118, 117, 116, 98, 114, 46, 99, 122, 37, 1, 1, 33, 1, 0]
16	7:59:46	INFO	ACP_ODESLANI	Zprava = START, ID transakce = 16220781, obsažená AVP = NAME_SUP_G ASSET_L LAM
17	8:0:0	INFO	DATA_DO	[133, 247, 130, 109, 0, 0, 53, 48, 4, 64, 33, 155, 239, 224, 0, 37, 48, 8, 50, 134, 37, 23, 189, 82, 131, 133, 176, 0, 24, 14, 79, 168, 198, ...
18	8:0:0	INFO	ACP_PRIJEM	Zprava = REQUEST, ID transakce = 16220781, obsažená AVP = INIT ENC
19	8:0:0	INFO	DATA_Z	[133, 0, 0, 1, 0, 0, 53, 48, 4, 64, 33, 155, 239, 224, 0, 37, 48, 8, 50, 134, 37, 23, 189, 82, 131, 133, 176, 0, 24, 14, 79, 168, 198, 74, 213,...
26	8:0:18	INFO	PORTAL_INFO	Smazán spoj = 192.168.1.106
27	8:0:18	INFO	TRANSAKCE	Smazán záznam transakce s id =32997997
20	8:0:2	INFO	DATA_DO	[130, 0, 0, 1, 0, 0, 137, 50, 128, 21, 106, 10, 136, 167, 221, 3, 221, 126, 212, 81, 80, 72, 154, 209, 96, 68, 154, 167, 4, 73, 174, 79, 123...
21	8:0:2	INFO	ACP_PRIJEM	Zprava = RESPONSE, ID transakce = 1, obsažená AVP = HMAC
22	8:0:2	INFO	DATA_Z	[130, 247, 130, 109, 0, 0, 137, 50, 128, 21, 106, 10, 136, 167, 221, 3, 221, 126, 212, 81, 80, 72, 154, 209, 96, 68, 154, 167, 4, 73, 174, ...
23	8:0:3	INFO	DATA_DO	[135, 247, 130, 109, 0, 0, 10, 38, 1, 0]
24	8:0:3	INFO	ACP_PRIJEM	Zprava = FINISH, ID transakce = 16220781, obsažená AVP = RESULT
25	8:0:3	INFO	DATA_Z	[135, 0, 0, 1, 0, 0, 10, 38, 1, 0]

Obr. 53 Logy portálu v tranzitním režimu

12 ZÁVĚR

V souladu se zadáním, řeší diplomová práce problematiku bezpečnosti současných autentizačních metod a rizik, které s nimi souvisí. Byl vytvořen a realizován návrh systému autentizace založeném na protokolu ACP, kde se autentizuje uživatel disponující čipovou kartou vůči aplikaci spuštěné na uživatelském počítači. Práce sestává ze tří hlavních bloků, rozebrání bezpečnosti autentizačních metod, popisem protokolu ACP a následným návrhem.

V první části jsou rozebrány současné autentizační metody založené na znalosti, vlastnictví tokenu a vlastnostech žadatele. Jsou zde popsány jejich rizika a možné útoky a demonstrační ukázky bezpečnostních chyb. Ze zde popsané problematiky, je patrné, že nejvyšší míru bezpečnosti zajišťují tokeny, konkrétně pak autentizátory. Naopak jako nejméně bezpečné (přesné) se jeví autentizace na základě vlastnosti. S vývojem nových technologií se však i tento způsob autentizace uživatele zdokonaluje, avšak za cenu vyšší složitosti měření a tím i cenu zařízení. V této části je pak také rozebrána problematika certifikátů, která slouží k jejich pochopení, jelikož na nich stojí bezpečnost navrženého systému autentizace.

Další část práce popisuje protokol ACP využitého v návrhu. Jeho účel, strukturu zpráv a způsob komunikace. Popis typů zpráv a jejich účel stejně tak jako typů AVP v nich obsažených. Následující kapitola je pak věnována problematice technologie Java Card, jelikož je tato technologie v návrhu využívána.

Poslední část práce se již zabývá konkrétním návrhem autentizačního systému založeného na protokolu ACP. Je zde rozebrána struktura a funkcionalita appletu vytvořeného pro tuto práci, který byl nainstalován na kartu Sm@rtCafé Expert 4, která byla k dispozici pro testování.

Je zde také rozebrána struktura navržené aplikace realizující ACP portál poskytovatele. Jeho struktura a způsob zpracovávání požadavků. Dále je popsán scénář komunikace, na kterém systém pracuje. Jsou zde rozebrány jednotlivé bezpečnostní prvky, které byly do systému zapojeny. Pro zajištění bezpečnosti komunikace mezi portálem a databází uživatelů byl použit šifrovaný spoj TLS, kdy se musí server prokázat platným certifikátem, podepsaným důvěryhodnou certifikační autoritou. Následný přenos dat je pak šifrován a je tedy eliminován problém zjištění citlivých informací přenášených z databáze. Komunikace mezi jednotlivými ACP portály zapojených v systému je opět zabezpečena pomocí zabezpečeného spoje, s tím rozdílem, že se autentizují obě strany. Bezpečnost konfiguračního souboru je pak zajištěna jeho šifrováním a není tedy možné jeho vlastnosti měnit bez znalosti hesla.

Co se týče bezpečnosti protokolu ACP, protokol ACP sám o sobě nezajišťuje důvěrnost a autentičnost. Jedná se o otevřenou strukturu, kdy úroveň bezpečnosti závisí na konkrétním průběhu transakce a navrženém bezpečnostním schématu. Zajištění důvěrnosti a autentičnosti zpráv lze provést u protokolu ACP externě nebo interně. V navrženém systému byla použita externí varianta pomocí šifrovaného spoje TLS mezi jednotlivými zúčastněnými portály organizací. Interní varianta byla pak použita při komunikaci karty a aplikace na počítači, kdy bylo potřeba bezpečně přenést zadaný pin z aplikace na kartu.

V systému autentizace, byl pak navržen scénář realizující dvoufaktorovou autentizaci, kdy se uživatel prokazuje kartou a znalostí pinu. Systém je navržen tak, že se do procesu autentizace může zapojit více portálů. Konkrétní navržené schéma autentizace založené na ověřování podpisu žadatele, přináší výhodu vtom, že jednotlivé portály mohou obsahovat veřejné klíče žadatele, jelikož se nejedná o bezpečnostní riziko a s jeho pomocí jsou schopni uživatele autentizovat a nemusí tedy o autentizaci žádat portál domácí organizace uživatele.

LITERATURA

[1] TODOROV, Dobromir. Mechanics of User Identification and Authentication : Fundamentals of Identity Management. [s.l.] : Auerbach Publications, 2007. 760 s. ISBN 1420052195

[2] SHAFFER, George. Good and Bad Passwords How-To. GeodSoft [online]. [cit. 2011-12-12]. Dostupný z WWW: <http://geodsoft.com/howto/password/cracking_passwords.htm>

[3] How to Avoid the Most Common and Dangerous Passwords. Zonealarm [online]. [cit. 2011-12-12]. Dostupný z WWW: <http://blog.zonealarm.com/2011/01/securing-yourself-from-a-world-of-hackers.html>

[4] How Rainbow Tables work. [online]. 2006, [cit. 2011-12-12]. Dostupný z WWW: <http://kestas.kuliukas.com/RainbowTables>

[5] ČÍŽEK, Jakub. Heslo „this is fun" je desetkrát bezpečnější než j4fS<2. Živě [online]. 2011, -, [cit. 2011-12-12]. Dostupný z WWW: <http://www.zive.cz/clanky/heslo-this-is-fun-je-desetkrat-bezpecnejsi-nez-j4fs2/sc-3-a-156806/default.aspx>

[6] BURDA, Karel. Bezpečnost informačních systémů. Brno, 2005. 104 s. Skriptum. FEKT VUT Brno

[7] RFID - technologie pro internet věcí. Pandatron [online]. 2009, [cit. 2011-12-12]. Dostupný z WWW: < http://pandatron.cz/?733&rfid_-_technologie_pro_internet_veci>

[8] PUST, Radim. Čipové karty Mifare a jejich bezpečnost. Elektrorevue. 2009, 27.

[9] MUIJRERS, Ruben. Dismantling MIFARE Classic [online]. The Netherlands. Radboud University Nijmegen. Dostupné z WWW: <http://www.sos.cs.ru.nl/applications/rfid/2008-esorics.pdf>

[10] Attacks on and Countermeasures for USB Hardware Token Devices. In - [online]. Cambridge : [cit. 2011-12-12]. Dostupné z WWW: <http://web.eecs.utk.edu/~dunigan/cns06/ usb_hardware_token.pdf>

[11] KOUŘIL, Daniel. Certifikáty veřejných klíčů. Zpravodaj ÚVT MU. 2000, 4, s. 5-9

[12] DOSTÁLEK, Libor. Velký průvodce protokoly TCP/IP: bezpečnost. Praha : Computer Press, 2003. 571 s. ISBN 80-7226-849-X

[13] BENEŠ, R. Autentizační metody založené na biometrických informacích. [online]. 2011, [cit. 2011-12-12]. Dostupný z WWW: <http://access.feld.cvut.cz/view.php?cisloclanku= 2010110002>

[14] ŘÍHA, Zdeněk; MATYÁŠ, Václav. Biometric Authentication Systems. Brno, 2000. 46 s. -. FI MU. Dostupné z WWW: <http://www.fi.muni.cz/reports/files/older/FIMU-RS-2000-08.pdf>

[15] ŠČUREK, Radomír. Biometrické metody identifikace osob v bezpečnostní praxi. Ostrava, 2008. 58 s. Studijní text. VŠB TU Ostrava.

[16] HANÁČEK, Petr. Čipové karty. Computerworld [online]. 1998, [cit. 2011-12-12]. Dostupný z WWW: <http://computerworld.cz/archiv/cipove-karty-9778>

[17] Mikrokontroléry a čipové karty. Pandatron [online]. 2011, -, [cit. 2011-12-12]. Dostupný z WWW: <http://pandatron.cz/?2631&mikrokontrolery_a_cipove_karty>

[18] WITTEMAN, Marc. Advances in Smardcards security. Information security bulletin [online]. 2002, 11, [cit. 2011-12-12]. Dostupný z WWW: <http://profs.info.uaic.ro/~fltiplea/IS/Witt2002.pdf>

[19] KUČERA, Jan. Courtoisův útok na MIFARE. Praha, 2010. 53 s. Bakalářská práce. Katedra algebry

[20] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. *Lecture Notes in Computer Science*,1109:104–113, 1996

[21] Paul Kocher, Joshua Jaffe, Benjamin Jun: Differential power analysis. V Advances in Cryptology - CRYPTO'99 (M. J. Wiener ed.), Lecture Notes in Computer Science no. 1666, Springer Verlag, 1999, strana 388-397.

[22] BURDA, K., STRASIL, I., PELKA T., STANCIL, P. Access Control Protocol (ACP). Brno: Vysoké učení technické v Brně, 2011. 25 s. Dostupné z WWW: <http://tools.ietf.org/html/draft-kaaps-acp-01>.

[23] BURDA, Karel, Ležák Petr. Aplikace univerzálního rámce řízení přístupu. Elektrorevue. 2012, 37, s. 1-5.

[24] BURDA, Karel. Univerzální rámec pro řízení přístupu v počítačových sítích. Elektrorevue. 2011, 9, s. 1-6

[25] ORACLE : Java Card Technology [online]. Oracle Corporation, 500 Oracle Parkway, Red Wood Shores : 2010 [cit. 2010-12-05]. Dostupné z WWW: <http://www.oracle.com/technetwork/java/javacard/overview/index.html>

[26] RANKL, Wolfgang; EFFING, Wolfgang. Smart Card Handbook. 3. vydání. Munich : John Wiley & Sons Ltd, 2003. 1043 s. ISBN 0470856688.

[27] CHEN, Zhigun. Java Card Technology for Smart Cards : Architecture and Programmer´s Guide. 2. vydání. Messachusetts : Addison Wesley, 2004. 368 s. ISBN 0201703297

[28] RANKL, Wolfgang. Smart Card Applications. West Sussex (England) : John Wiley & Sons, Ltd., 2007. 217 s. ISBN 978-0-470-05882-4.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

PIN	Personal Identification Number
SIM	Subscriber Identity Module
USB	Universal Serial Bus
GSM	Global System for Mobile Communications
UMTS	Universal Mobile Telecommunication System
Telnet	Telecommunication Network
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
SSH	Secure Shell
POP3	Post Office Protocol
SHA-1	Secure Hash Algorithm
MD5	Message-Digest algorithm
RSA	Rivest, Shamir, Adleman
RAM	Read Access Memory
ROM	Read Only Memory
EEPROM	Electrically Erasable Programmable Read Only Memory
RFID	Radio frequency identification
SPA	Simple power analysis
DPA	Differential power analysis
DES	Data Encryption Standard
CRC	Cyclic redundancy check
3DES	Triple Data Encryption Standard
MKEY	Master Key
VK	Veřejný klíč
SK	Soukromý klíč
MITM	Man in the middle
DER	Distinguished Encoding Rules
PEM	Privacy Enhanced Mai
CA	Certifikační autorita
HSM	Hardware Security Module

RA	Registrační autorita
CRL	Certificate Revocation List
URL	Uniform Resource Locator
FRR	False rejection rate
FAR	False acceptance rate
ACP	Access Control Protocol
RADIUS	Remote Authentication Dial In User Service
EAP	Extensible Authentication Protocol
AVP	Attribute Value Pair
SAVP	Short AVP
LAVP	Long AVP
CAVP	Container AVP
ACP-VSA	ACP Variant of Single Answers
APDU	Application Protocol Data Unit
AID	Application identifier
TPDU	Transport Protocol Data Unit
TCP	Transmission Control Protocol
TLS	Transport Layer Security