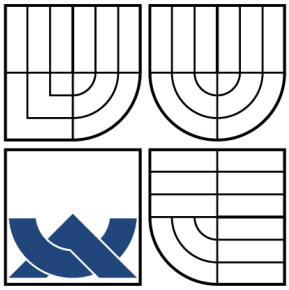
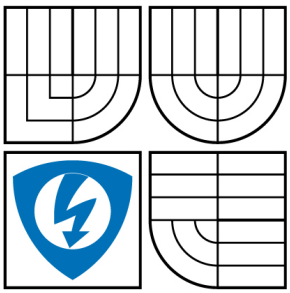


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

MHP APLIKACE

MHP APPLICATION

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

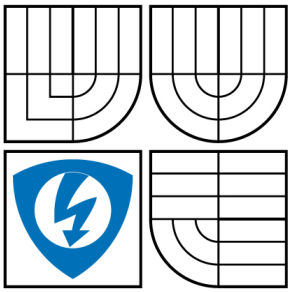
AUTOR PRÁCE — Bc. TOMÁŠ HOLÍK

AUTHOR

VEDOUCÍ PRÁCE — Ing. PETR ČÍKA

SUPERVISOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor

Telekomunikační a informační technika

Student: Holík Tomáš Bc. ***ID:*** 88502

Ročník: 2 ***Akademický rok:*** 2007/2008

NÁZEV TÉMATU:

MHP aplikace

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s platformou MHP pro vytváření interaktivních aplikací pro DVB. Navrhněte aplikaci umožňující sázení na sportovní zápasy. Společně s aplikací navrhněte databázi, která bude obsahovat uživatelské informace společně s výší kreditu pro sázení. Komunikace MHP aplikace s databází bude probíhat po zabezpečeném kanále. Základem celé aplikace bude uživatelsky přístupné grafické rozhraní.

DOPORUČENÁ LITERATURA:

[1] Schwalb, E. M. ITV Handbook: Technologies and Standarts, London: Prentice Hall, 2004. ISBN 978-0131003125

[2] STEVEN, M. Interactive TV standards: A Guide to MHP, OCAP, and JavaTV. Elsevier: Focal Press. 2005, ISBN 978-0240806662

Termín zadání: 11.2.2008 ***Termín odevzdání:*** 28.5.2008

Vedoucí práce: Ing. Petr Číka

prof. Ing. Kamil Vrba, CSc.

předseda oborové rady

UPOZORNĚNÍ:

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení: Bc. Tomáš Holík
Bytem: Hranická 84/25, 75124, Přerov - Přerov II-Předmostí
Narozen/a (datum a místo): 26.9.1983, Přerov

(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta elektrotechniky a komunikačních technologií
se sídlem Údolní 244/53, 60200 Brno 2
jejímž jménem jedná na základě písemného pověření děkanem fakulty:
prof. Ing. Kamil Vrba, CSc.

(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
   - ☐ disertační práce
   - ☑ diplomová práce
   - ☐ bakalářská práce

   jiná práce, jejíž druh je specifikován jako .....

(dále jen VŠKP nebo dílo)

Název VŠKP: MHP aplikace
Vedoucí/školitel VŠKP: Ing. Petr Číka
Ústav: Ústav telekomunikací
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:

- ☑ tištěné formě - počet exemplářů 1
- ☑ elektronické formě - počet exemplářů 1

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2

Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizovaní výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti
   - ☑ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3

Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

....

Nabyvatel Autor

ABSTRAKT

Tato práce se zabývá návrhem interaktivní aplikace pro online sázení v prostředí digitální televize DVB-T. Aplikace je určena pro běh na multimediální platformě MHP a má umožnit sázení na sportovní zápasy vysílané prostřednictvím digitální televize. Celá aplikace je napsána v jazyce Java. Veškeré údaje o uživatelích a zápase, na který je možno sázet, jsou uloženy v MySQL databázi na serveru, která je přístupná prostřednictvím PHP skriptu. Aplikace komunikuje se skriptem prostřednictvím zpětného kanálu set-top boxu, na kterém je provozována. Komunikace mezi aplikací a serverem probíhá nezabezpečeným kanálem, proto je celá komunikace šifrována. Samotné šifrování probíhá symetrickou AES šifrou, šifrování a dešifrování se provádí se znalostí soukromého klíče. Tento soukromý klíč si vygeneruje každá aplikace při spuštění a na server ho zašle zašifrovaný veřejným klíčem serveru. K zašifrování klíče je využita asymetrická kryptografická metoda RSA.

KLÍČOVÁ SLOVA

digitální televize, MHP, Java, PHP, kryptografie

ABSTRACT

This Master Thesis is engaged in designing an interactive application for online betting in the DVB-T environment. The application is determined to running at Multimedia Home Plattform MHP and it allowes digital television viewers to make a bet on a sport match broadcasted on the television. The whole application is written in the Java language. All the information about users and the match are stored in a MySQL database, which is accessible through a php script placed on the server with the database. The applicaton is running in a set-top box and communicates with the script through the set-top box return channel. The channel between the server and the script is unsecure, so the communication has to be encrypted. The encryption process uses a symmetric-key cipher AES, which uses a shared secret key for encryption and decryption. Each application generates its own pseuodorandom and crypthographicaly strong AES key and sends it to the server. To secure the secret key transsmision is used the public-key cipher RSA.

KEY WORDS

digital television, MHP, Java, PHP, cryptography

HOLÍK, T. *MHP aplikace*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. 58 s. Vedoucí diplomové práce Ing. Petr Číka.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma "MHP aplikace" 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 jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.“

V Brně dne …… ……..

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM ZKRATEK

DTV	Digitální televize
NTSC	National Television Systems Committee
PAL	Phase Alternation Line
HDTV	High Definition Television
MPEG	Motion Picture Experts Group
VSB	Vestigial Sideband Modulation
DVB	Digital Video Broadcasting
OFDM	Orthogonal Frequency Division Multiplexing
EPG	Electronic Program Guide
QPSK	Quadrature Phase-Shift Keying
QAM	Quadrature Amplitude Modulation
SDTV	Standard Definition Television
LDTV	Low Definition Television
PVR	Personal Video Recorder
MHP	Multimedia Home Platform
API	Application Programming Interface
JVM	Java Virtual Machine
PiP	Picture in Picture
AWT	Abstract Window Toolkit
HAVi	Home Audio / Video Interoperability
PHP	Personal Homepage
SQL	Structured Query Language
RSA	Rivest, Shamir, Adleman
AES	Advanced Encryption Standard
DH	Diffie-Hellman
DES	Data Encryption Standard
ECB	Electronic CodeBook
CBC	Cipher-Block Chaining
SHA	Secure Hash Algorithm
GUI	Graphical User Interface

1 Úvod

V dnešní době dochází k stále masovějšímu přechodu od analogových technologií k technologiím digitálním. Postup digitalizace je nezadržitelný, zasahuje prakticky do všech oborů, ať už se jedná o audio techniku a přechodu od gramofonových desek až k CD nosičům, nebo třeba o fotoaparáty a jejich přeměnu do digitální podoby. Ani televizní technika není výjimkou. Analogová televize u nás sice stále vysílá, ale její dny jsou již sečteny. Analogová technika má bezesporu své kouzlo, ale technika digitální přináší spoustu nových možností a lepší kvalitu.

Standardem pro digitální televizi se v Evropě stal projekt DVB, který zahrnuje standardy pro kabelové (DVB-C), satelitní (DVB-S) a pozemní (DVB-T) televizní vysílaní. Vysílané programy jsou komprimovány, což umožňuje daleko lepší využití frekvenčního spektra než tomu bylo u analogové televize. Na jednom televizním kanále, který je u analogové televize využíván pro přenos jediného programu, je přenášen tzv. multiplex, který může obsahovat několik televizních nebo rozhlasových stanic a doplňkových služeb, ke kterým patří zejména EPG (Electronic Program Guide) což je elektronická verze televizního programu, superteletext a interaktivní aplikace, umožňující on-line nákupy, hlasování, e-mail nebo jednoduché hry.

Spouštění interaktivních aplikací umožňuje platforma MHP, evropský standard domácí multimediální platformy. Interaktivní aplikace jsou vysílány společně s audio a video signálem, jsou součástí transportního toku. Tyto aplikace mohou být například hry, interaktivní hlasování, e-mail nebo SMS. Pro zajištění interaktivity aplikací je ale zapotřebí zpětný kanál realizující spojení od diváka k vysílateli.

Tato práce se zabývá návrhem interaktivní aplikace pro platformu MHP, která umožní sázení na sportovní zápasy z pohodlí domova, prostřednictvím digitální televize. Divákovi je prostřednictvím aplikace umožněno vsadit si na výsledek vysílaného zápasu. Poté může nerušeně sledovat průběh zápasu a po jeho skončení ověřit výsledek své sázky opětovným spuštěním aplikace.

Vlastní návrh aplikace je podrobně rozpracován včetně teorie o digitální televizi, platformě MHP, databázích, kryptografii a jazycích Java a PHP.

2 Úvod do digitální televize

V současné době stále roste zájem o digitální televizi, která postupem času zcela jistě nahradí stávající analogovou televizi. Hlavním důvodem tohoto zájmu jsou velmi výhodné vlastnosti, které digitální televize (DTV) nabízí. Digitální televize umožňuje přenos obrovského množství informací k maximálnímu počtu diváků za velmi nízkou cenu a může být plně integrována do digitálních přenosových síti.

Základem digitální televize je digitální signál, ze kterého pramení většina výhodných vlastností digitální televize. I když při převodu signálu z analogové formy do digitální dochází ke ztrátě informace a ke vzniku kvantovacího šumu, který znehodnocuje původní zdrojový signál, má digitální vyjádření signálu významné výhody, díky kterým je digitální signál v současné době výhodnější než analogový. Digitální signál může být upravován mnoha způsoby, které u analogového signálu nejsou možné. Mezi tyto úpravy patří například aplikace moderních kompresních algoritmů. Tato úprava umožňuje lepší využití kmitočtového spektra a tím je umožněn přenos více programů na stávajících kmitočtech, což je další velká výhoda digitální televize. Analogová televize může v jednom přenosovém kanálu přenášet pouze jediný program a k dobrému pokrytí České republiky nestačilo ani 48 stávajících kanálů, pro vysílání čtyř programů v digitální formě postačí hypoteticky tři televizní kanály.

2.1 Historické pozadí pro vývoj digitální televize

Současné analogové televizní systémy jsou založeny na systému NTSC (National Television Systems Committee) vyvinutém v USA v roce 1953. Tento systém je používán v Americe, Kanadě, Mexiku a Japonsku a Koreji od roku 1954. V Evropě, Austrálii a na Dálném východě je používána jiná varianta NTSC systému s názvem PAL (Phase Alternation Line). Dalším systémem je například SECAM, který je využíván ve Francii, Egyptě a Bulharsku. Tyto rozdílné standardy vytvořily bariéru, která zabránila vzniku mezinárodního televizního vysílání. Nicméně tyto analogové standardy budou nahrazeny standardy digitálního televizního vysílání, které jsou méně omezující při vysílání a které jsou schopny zajistit rozmanité služby [3].

2.2 DTV v USA

V roce 1987 začal v USA proces výběru vhodného standardu pro HDTV (High Definition Television), který by byl kompatibilní s existujícím analogovým televizním standardem NTSC. Vznikl Dolby standard nazvaný AC-3 pro vícekanálové kódování audia a MPEG standard známý jako MPEG-2 pro kódování zdrojového videa, poskytování systémových informací a multiplexování. Byla také specifikována osmistavová modulace (8 VSB – Vestigial Sideband Modulation) a systém pro kabelové televizní systémy [3].

2.3 DTV v Evropě

O výběr vhodného HDTV standardu pro Evropu se pokoušelo začátkem 90.let mnoho organizací, které v roce 1992 vytvořili skupinu ELG (European Launching Group). Díky tomuto uskupení organizací, výrobců a telekomunikačních společností vznikl v roce 1993 standard DVB (Digital Video Broadcasting). Projekt DVB vyvinul standardy pro kabelové (DVB-C), satelitní (DVB-S) a pozemní (DVB-T) vysíláni digitální televize, specifikoval MPEG-2 jako standard pro zdrojové kódování pro video a audio, systémové informace a multiplexování. Projekt DVB také specifikoval modulaci pro pozemní vysílání digitální televize. Touto modulací je COFDM (Coded Orthogonal Frequency Division Multiplexing) [3].

3 DVB-T

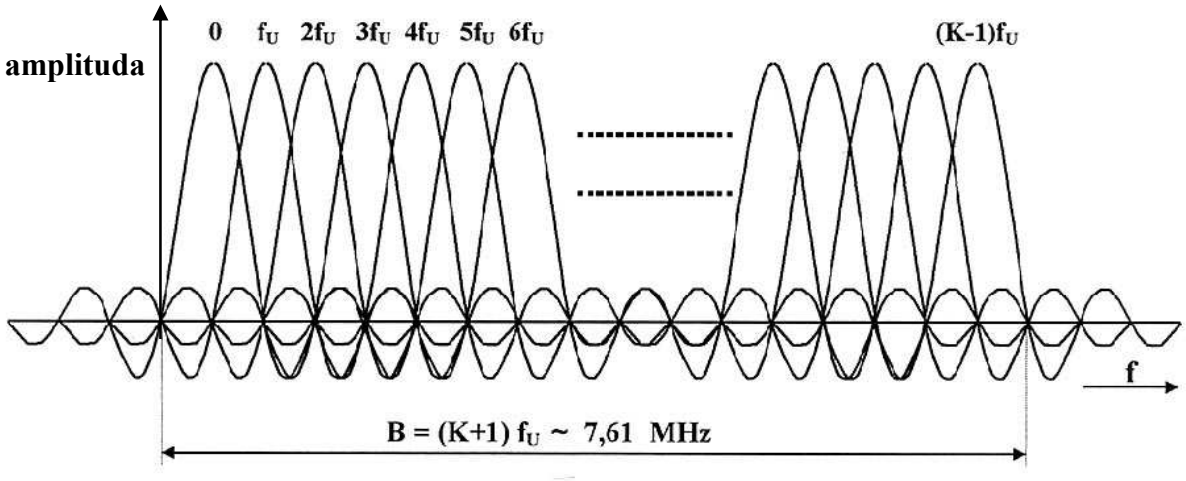
DVB-T je standard pro digitální televizní vysílání přes pozemní vysílače. Vysílané programy jsou komprimovány, což umožňuje daleko lepší využití frekvenčního spektra než tomu bylo u analogové televize. Na jednom televizním kanále, který je u analogové televize využíván pro přenos jediného programu, je přenášen tzv. multiplex, který může obsahovat několik televizních nebo rozhlasových stanic a doplňkových služeb, ke kterým patří zejména EPG (Electronic Program Guide) což je elektronická verze televizního programu, superteletext, popř. další interaktivní služby jako jsou on-line nákupy, hlasování, e-mail nebo jednoduché hry [3].

3.1 Modulace pozemního televizního vysílání

Standard DVB-T využívá pro digitální pozemní vysílání ortogonálně dělený multiplex OFDM (Orthogonal Frequency Division Multiplex). Modulační systém OFDM je založený na použití velkého počtu dílčích digitálně modulovaných subnosných vln umístěných rovnoměrně v přenosovém kanálu 8 MHz (případně 7 nebo 6 MHz). Standard DVB-T připouští dvě základní varianty OFDM s rozdílným počtem nosných vln, odvozeným od „počítačových“ hodnot mocnin dvou – 2k (2048) a 8k (8192).

Každá dílčí vlna (o frekvenci f_U) je digitálně modulována s využitím čtyřstavové kvadraturní modulace QPSK (Quadrature Phase Shift Keying), 16-stavové kvadraturní modulace 16-QAM (Quadrature Amplitude Modulation) nebo 64-stavové kvadraturní modulace 64-QAM, tzn. přenáší současně 2, 4 nebo 6 bitů. Spektrum jednotlivých subnosných je znázorněno na obrázku 3.1. Celkové spektrum OFDM je dáno součtem všech dílčích spekter [2].

Podstatnou výhodou OFDM je optimální využití přenosového kanálu. Spektrum signálu modulovaného OFDM je rovnoměrné v celém kanálu, to má za důsledek to, že signál má charakter šumu. Ve skutečnosti je ale tento „šumový“ signál přesně definovaný a velmi důkladně zabezpečený proti přenosovým chybám. Původní digitální signál se získá postupnou realizací všech operací v dekodéru v obráceném sledu.



Obr. 3.1: Spektrum dílčích subnosných COFDM v systému DVB-T

K modulaci dílčích subnosných vln se v kodéru s výhodou používá algoritmus inverzní rychlé Fourierovy transformace IFFT a v dekodéru pak algoritmus přímé rychlé Fourierovy transformace (FFT).

Rozdělením celkového bitového toku mezi *K* nosných vln dojde k prodloužení doby trvání každého bitu právě *K*-krát. V podstatě se jedná o změnu charakteru přenosu, ze sériového na paralelní. Místo jedné nosné vlny je informace přenášena mnoha nosnými současně. Velmi důležitou roli v systému OFDM hraje protichybové zabezpečení, proto se často přesněji označuje jako COFDM (Coded OFDM) [3].

Standard DVB-T umožňuje volit ze tří modulačních schémat a pěti kódových poměrů vnitřního konvolučního kódování. Podle stupně zabezpečení je pak možno volit užitečný datový tok digitální pozemní televize v rozmezí 5 až 32 Mbit/s.

3.2 Kanálové kódování

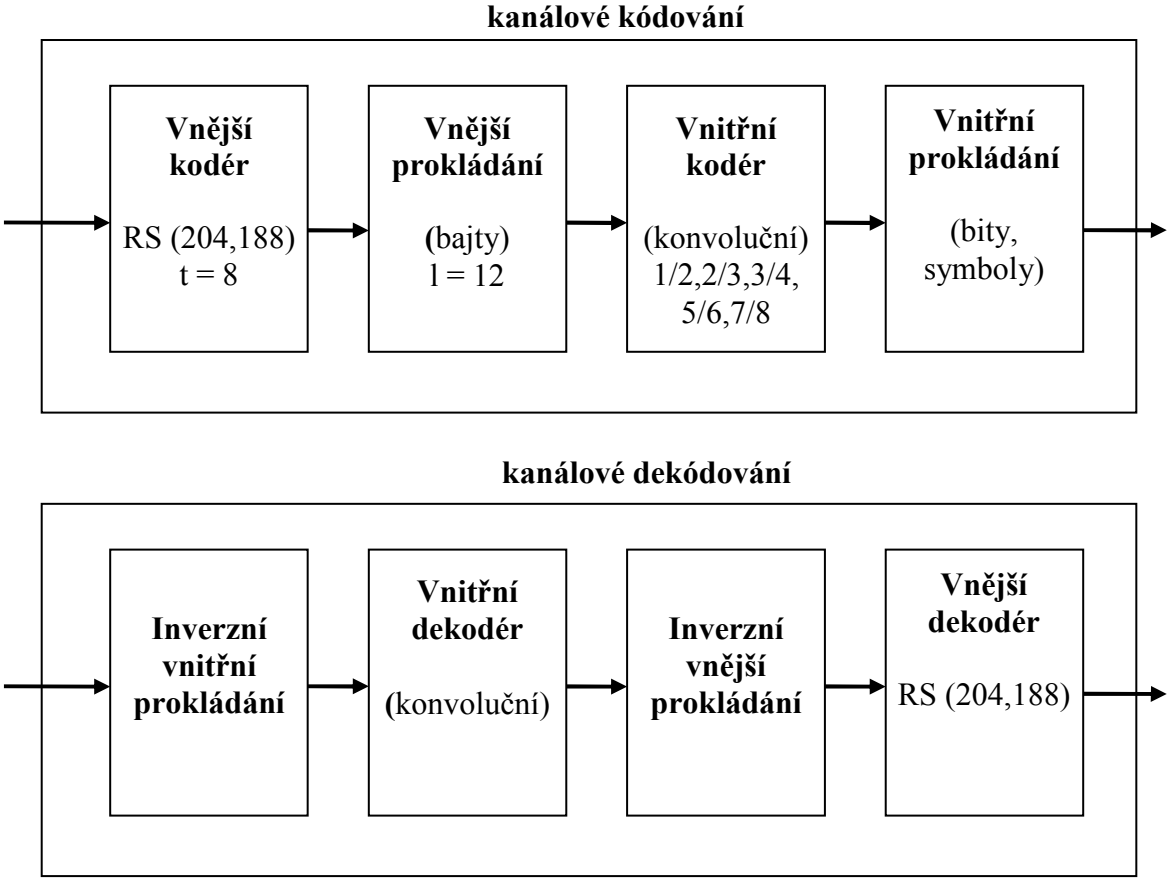
Na jeden aktivní snímek připadá při 576 aktivních řádcích ve snímku a 720 obrazových prvcích na řádku zhruba 415 000 obrazových prvků. Při osmibitovém kódování spotřebujeme na zakódování jednoho obrazového prvku průměrně 16 bitů (8 bitů pro každý jasový vzorek a 16 bitů pro chrominanci, ale jen v každém druhém obrazovém prvku). Aktivní bitový tok nekomprimovaného obrazového signálu 4:2:2 je tedy při 25 snímcích/s zhruba 165Mbit/s, aktivní bitový tok signálu 4:2:0 asi 124 Mbit/s. S využitím kódování pomocí standardu MPEG-2 je možné dosáhnout výrazné redukce bitového toku obrazového signálu.

Užitečným bitovým tokem rozumíme tok odpovídající obrazovému signálu a příslušným zvukovým i datovým signálům včetně nezbytných synchronizačních informací a např. pohybových vektorů přenášených pro správné dekódování. Bity pro korekci přenosových chyb se do užitečného signálu nezahrnují. Výsledné užitečné bitové toky zdrojového kódování určené pro distribuci televizního signálu závisejí na mnoha okolnostech. Jako orientační hodnoty můžeme uvažovat bitové toky 3 až 6 Mbit/s pro standardní kvalitu SDTV (Standard Definition Television) a kolem 20 Mbit/s pro kvalitu blížící se HDTV (High Definition Television). Kvalita LDTV (Low Definition Television) vyžaduje bitový tok kolem 1,5Mbit/s. S vývojem techniky dochází postupně k redukci datových toků obrazových signálů při zachování stejné kvality.

Klíčovými problémy kanálového kódování jsou převedení užitečného bitového toku s kvalitou HDTV, SDTV nebo LDTV (s příslušnými zvukovými a doplňkovými digitálními signály) do přenosového kanálu, dostatečné protichybové zabezpečení a doprava k divákovi. Protichybovým zabezpečením dochází ke zvýšení datového toku. K přenosu zabezpečených bitových toků do přenosového kanálu charakterizovaného šířkou pásma (MHz) slouží při kanálovém kódování vícestavové digitální modulace.

V pozemní digitální televizi se využívá kompletní zabezpečení bitového toku zahrnující vnější kódování, vnější prokládání, vnitřní kódování a vnitřní prokládání. Nejmenší zabezpečení se používá v kabelové televizi s ohledem na nízkou úroveň rušení v kabelech. Popsaný řetězec zabezpečení je zobrazen na obr. 3.2. Na výstupu uvedeného řetězce

dostáváme celkový hrubý bitový tok, který se pomocí digitální vícestavové modulace přenáší v daném kmitočtovém pásmu.



Obr. 3.2: Protichybové zabezpečení DVB-T

Ještě před korekcí chyb se v kodéru v rámci kanálového kódování provádí skramblování dat. Vnější kodér zajišťuje korekci chyb pomocí Reed-Solomonova kódu RS (204, 188, t =8). Ke každému skramblovanému paketu přidává 16 zabezpečovacích bajtů, takže v každém paketu lze opravit až 8 chybných bajtů. Vnější korekce chyb zvyšuje užitečný bitový tok v poměru 204/188.

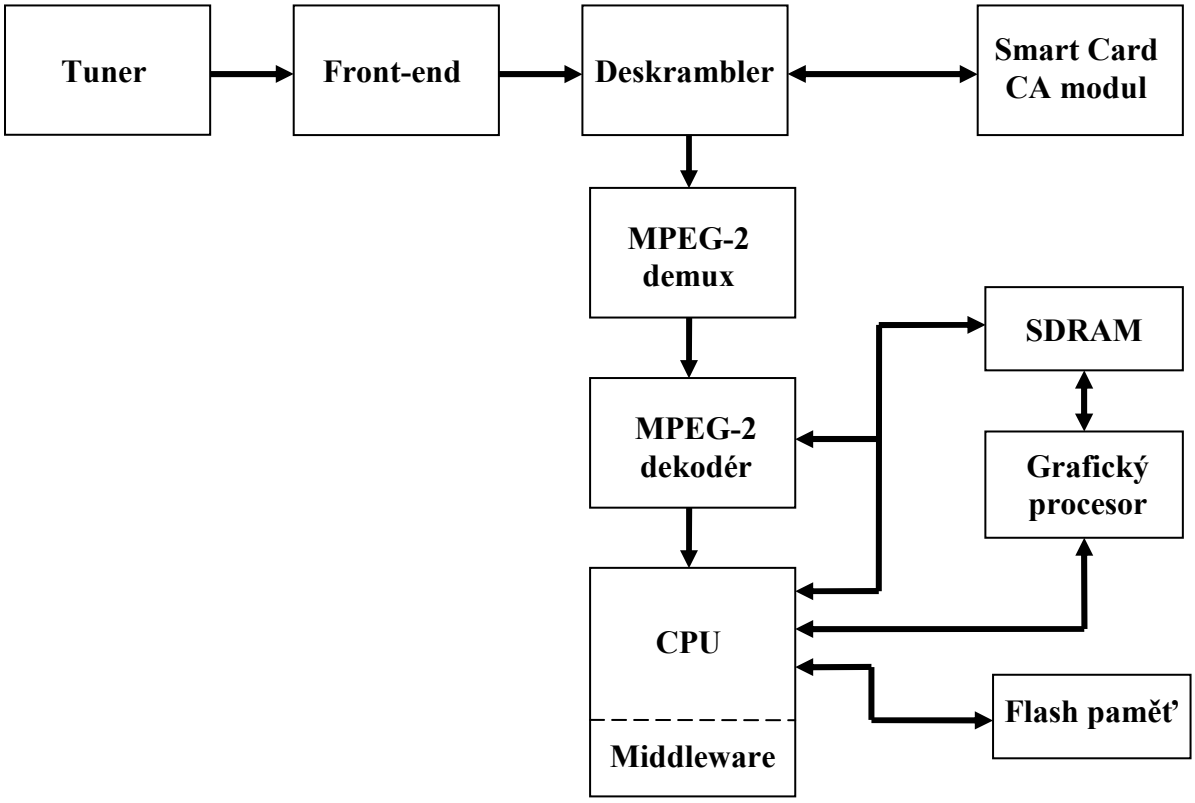
Účelem prokládání (interleaving) je zajištění přenosu proti shlukům chyb. Vnější konvoluční prokládání s hloubkou $l = 12$ zajišťuje, že i v případě přenosové chyby o celkové délce 12 bajtů se na vstupu dekodéru nevyskytnou bezprostředně za sebou dva chybné bajty a chyba může být vykorigována vnějším kódováním. Podobně vnitřní prokládání zajišťuje dostatečné rozházení sousedních bitů a symbolů OFDM při pozemním vysílání digitální televize. Vnější ani vnitřní prokládání přitom celkový bitový tok neovlivňují, protože jednotlivé bity, bajty a symboly pouze časově „rozhází“ aniž by samy přidávaly nějaké korekční bity.

Vnitřní konvoluční kódování transformuje k zdrojových bitů na n zakódovaných bitů, kódový poměr je k/n. V důsledku vnitřního kódování se bitový tok zvyšuje v obráceném poměru n/k. Normy DVB připouštějí kódové poměry 1/2, 2/3, 3/4, 5/6 a 7/8. [3].

4 Set-top box (STB)

Set-top box je zařízení které převádí digitální televizní signál do formátu, který může být interpretovaný existujícími analogovými televizními přijímači. Někdy bývají označovány jako digitální set-top boxy (DSTB). Set-top box zahrnuje všechen hardware, middleware a přístupový software sloužící uživateli k dekódování všech příchozích videí, audia i dat. Umožňuje přístup k veškerému digitálnímu obsahu a službám s využitím analogového televizního přijímače jako zobrazovací jednotky. Set top boxy umožňující záznam videa na pevný disk, nebo jiné médium bávají označovány jako PVR (Personal Video Recorder). V některých případech se STB vyrábí přímo jako součástí televizního přijímače.

Set-top boxy pro DVB-T systém podporují obě varianty počtu nosných u COFDM, 2k i 8k. Standardně se STB k analogové televizi připojuje prostřednictvím SCART konektoru, popřípadě přímo do anténního vstupu, pokud televizor neobsahuje videovstup. STB ale musí být pro tuto variantu připojení vybaven modulátorem.



Obr. 4.1: Bloková struktura přijímače DTV

Na obr. 4.1 je znázorněna bloková struktura typického přijímače pro DTV. Nemusí se jednat pouze o STB, ale například o přijímač integrovaný přímo v televizním přijímači nebo o televizní kartu v počítači. Tuner přijímá signál na frekvenci určené set-top boxem a namoduluje ho. Součást nazvaná Front-end zajišťuje první úroveň korekce chyb a první úroveň depaketizace, jeho výstupem je transportní tok MPEG-2 v digitální podobě. Demultiplexer od sebe odděluje jednotlivé signálové toky - video a audio, datový tok a informace o službách a předává příslušné toky dalším blokům v STB - audio a video do

MPEG-2 dekodéru, datový tok a informace o službách do procesoru (CPU) pokud o ně má zájem. MPEG-2 dekodér se stará o dekódování audia a videa a o správné zobrazení videa na obrazovce. Může zahrnovat podporu pro změnu rozměru videa, změnu pozice videa na obrazovce, nebo například zobrazení kurzoru na obrazovce. Často bývá integrován přímo do procesoru. Procesor neboli CPU zajišťuje dekódování a zpracování servisních informací, dekódování datových toků, interakci s uživatelem, spouštění stažených nebo zabudovaných aplikací. Procesor nemusí mít integrovaný pouze MPEG-2 dekodér, často jsou v procesoru integrovány i další bloky set-top boxu – MPEG-2 demultiplexer, deskrambler nebo grafický procesor. Blok CA modul (Conditional Acces) představuje protipirátský systém pro placené televize [2].

5 Multimedia Home Platform (MHP)

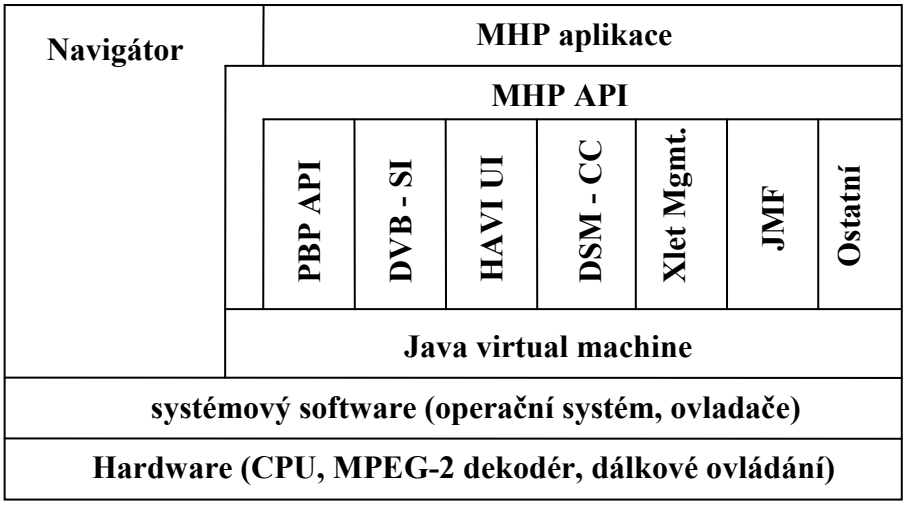
Aby bylo možné na televizním přijímači spouštět interaktivní aplikace, musí podporovat platformu umožňující jejich běh. Jednou z těchto platforem je MHP, evropský standard domácí multimediální platformy.

Zjednodušeně můžeme MHP popsat jako soubor instrukcí, které říkají operačnímu systému běžícímu na přijímači DTV, jak pracovat s interaktivními aplikacemi, které přijímá. MHP také definuje formu, ve které jsou aplikace doručeny do přijímače, včetně servisních informací, které signalizují, že se tyto interaktivní aplikace nacházejí v příchozím transportním toku (TS – Transport Stream).

MHP standard byl vytvořen s využitím specifikace programovacího jazyka Java, rozšířeného o funkce a instrukce určené pro uživatelské řízení domácích multimediálních aplikací. Jeho ustanovení předcházel vznik řešení jednotlivých společností, které byly navzájem nekompatibilní [2].

Interaktivní aplikace jsou vysílány společně s audio a video signálem, jsou součástí transportního toku. Tyto aplikace mohou být například hry, interaktivní hlasování, e-mail nebo SMS. Pro zajištění interaktivity aplikací je ale zapotřebí zpětný kanál realizující spojení od diváka k vysílateli. MHP platforma je svázána se standardem DVB, není ji proto možné provozovat na žádné jiné implementaci digitální televize než na zmíněné DVB.

Struktura platformy MHP je na obr. 5.1. MHP specifikuje rozsáhlé prostředí pro spouštění aplikací digitální televize, nezávislé na výchozím výrobcem specifikovaném hardwaru a softwaru. Toto prostředí je založeno na využití tzv. virtuálního stroje JVM (Java Virtual Machine), jehož funkcí je zajistit vazbu na hardware a interpretovat javovský bajtkód, a definic API (Application Programming Interface), které představují knihovní třídy. Jak je patrné z obrázku, MHP aplikace se nachází na pomyslném vrcholu těchto definic API. Aplikace, která umožňuje uživateli přístup k MHP aplikacím a ostatním službám DVB (např. TV nebo rádio) se nazývá navigátor [2].



Obr. 5.1: Struktura platformy MHP

V současné době existuje několik verzí platformy MHP, původní verze 1.0 byla nahrazena postupně verzí 1.1, kterou budeme využívat pro vytvoření požadované aplikace. V nedávné době vznikly také verze 1.1.3 a 1.2, které ale nejsou zatím set-top boxy podporovány.

5.1 Middleware

Aby mohl STB poskytovat datové služby, potřebuje softwarovou vrstvu, která se nachází mezi hardwarem STB a aplikací, která je právě zpracovávána. Tato softwarová vrstva se nazývá middleware. Set top boxy mající různé middleware vrstvy nemusí být schopny zpracovat stejná data nebo interaktivní služby. Z toho vyplývá, že každá služba musí být navržena pro konkrétní middleware systém na kterém má běžet. Příkladem middleware systému (standardu) je už zmíněná MHP platforma [2].

6 Jazyk Java

Java je programovací jazyk a prostředí, které bylo navrženo k řešení mnoha problémů v moderní programátorské praxi. Vznikl jako součást většího projektu, který se zabýval vývojem pokročilého softwaru pro zařízení spotřební elektroniky. Jedná se o objektově orientovaný programovací jazyk, který vyvinula firma Sun Microsystems. Java je jedním z nejpoužívanějších programovacích jazyků na světě. Díky své přenositelnosti je používána pro programy, které mají pracovat na různých systémech počínaje čipovými kartami, přes mobilní telefony a různá zabudovaná zařízení (platforma Java ME), aplikace pro desktopové počítače (platforma Java SE) až po rozsáhlé distribuované systémy (platforma Java EE). Tyto technologie se jako celek nazývají platforma Java [4].

Mezi základní výhody jazyka Java patří jeho jednoduchost, jedná se o zjednodušenou a drobně upravenou verzi syntaxe jazyka C a C++, bezpečnost, nezávislost na architektuře, přenositelnost, víceúlohovost, a v neposlední řadě také výkonnost. Jazyk Java patří mezi interpretované jazyky, tzn. místo skutečného strojového kódu se vytváří pouze tzv. mezikód (bajtkód). Tento formát je nezávislý na architektuře počítače nebo zařízení. Program pak může pracovat na libovolném počítači nebo zařízení, který má k dispozici interpret Javy, tzv. virtuální stroj Javy – Java Virtual Machine (JVM). Další vlastností programu napsaného v jazyce Java je jeho robustnost. Java je určena pro psaní vysoce spolehlivého softwaru a z tohoto důvodu neumožňuje některé programátorské konstrukce, které jsou častou příčinou chyb (např. správa paměti, používání ukazatelů). Správa paměti je realizována pomocí automatického Garbage collectoru, který automaticky vyhledává již nepoužívané části paměti a uvolňuje je pro další použití [4].

6.1 JavaTV

JavaTV je specifikace API, od firmy Sun Microsystems pro platformy digitální televize. Sdružuje dohromady všechny elementy, které jsou potřebné pro platformy DTV, jako je například přístup k televiznímu vysílání nebo servisním informacím. Pro přístup k televiznímu vysílání DTV je možno kromě JavaTV API využít také JMF (Java Media Framework), která je popsána dále v textu. Na rozdíl od standardu MHP není JavaTV svázána s žádným specifickým standardem pro digitální televizi [1].

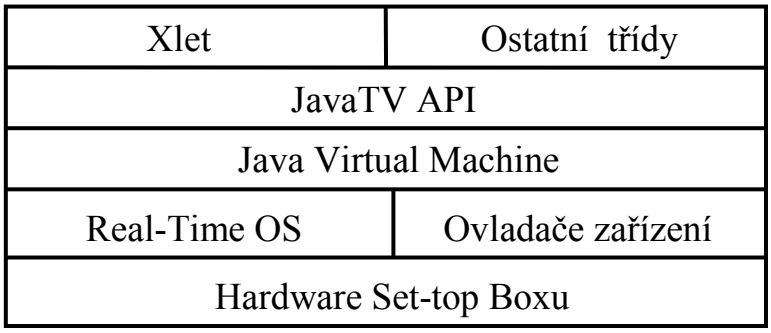
Stejně jako všechny ostatní knihovní funkce v jazyce Java jsou JavaTV API organizovány do balíků, výchozím balíkem je `javax.tv`. Některé základní balíky se stručným popisem jsou uvedeny v následující tabulce.

Architektura JavaTV je zobrazena na obr. 6.1. Aplikace JavaTV se skládá ze vstupního bodu reprezentovaným Xletem (viz dále) a případných dalších tříd. Kód v Xletu a ostatních třídách provádí svou funkci prostřednictvím volání JavaTV API a je vykonáván javovským virtuálním strojem (JVM), který běží pod operačním systémem a rozhraními s ovladači zařízení [1].

Tab. 6.1: Některé balíky specifikace JavaTV

Balík	Popis
`javax.tv.xlet`	Model životního cyklu aplikace a podpůrné třídy
`javax.tv.locator`	Reference na vysílané média a služby, na bázi URL
`javax.tv.net`	Mechanismus pro přístup k IP datagramům obsažených ve vysílání
`javax.tv.graphics`	Podpora pro práci s videem a grafikou
`javax.tv.media`	Přidává podporu pro funkčnost TV do JMF
`javax.tv.media.protocol`	Přidává do JMF podporu pro streamované TV vysílání
`javax.tv.service`	Koncepty pro popis služeb digitální televize

JavaTV není pouze soubor knihovních funkcí, jedná se o specifikaci API, proto není možné napsat program pro JavaTV bez této specifikace.



Obr. 6.1: Architektura JavaTV

6.2 Java Media Framework (JMF)

K řízení správného dekódování a zobrazování audia a videa využívá JavaTV a MHP Java Media Framework od SunMicrosystems. I když je JMF využíváno hlavně pro řízení toho co vysílané video prezentuje a jak to prezentuje, aplikace mohou JMF využívat k přehrávání zvukových souborů a zobrazování speciálních obrazových formátů.

JMF je standardizované API pro řízení médií a existovalo již před vznikem MHP a JavaTV. Ačkoliv se API používané v MHP od originální JMF specifikace příliš neliší, existují jistá omezení, co funguje a co ne, liší se návratové hodnoty některých funkcí.

JMF API jsou umístěny v balíku `javax.media`, ale některá rozšíření pro DTV se vyskytují i některých dalších balících [2].

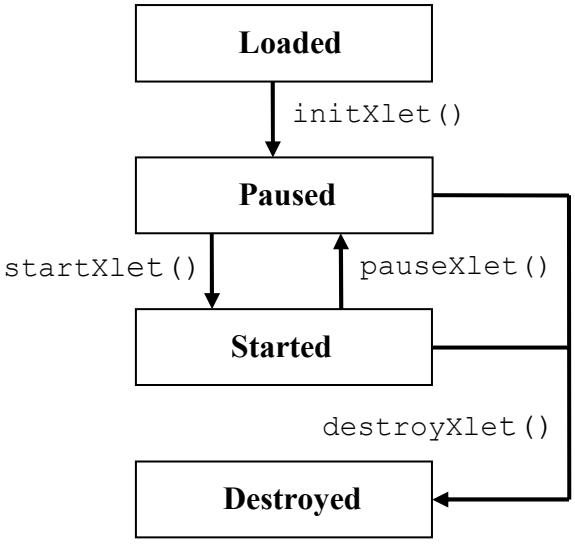
6.3 Xlet

V prostředí digitální televize nelze použít klasický model Java aplikace. Klasický model předpokládá, že daný virtuální stroj zpracovává pouze jednu aplikaci a že tato aplikace má plnou kontrolu nad vlastním životním cyklem (což zahrnuje i možnost ukončení běhu JVM). Tento přístup není pro přijímače DTV vůbec vhodný, protože je potřeba zajistit současný běh více aplikací najednou a také zajistit oddělení jednotlivých aplikací. K tomuto účelu nám nejlépe poslouží applet. Na rozdíl od klasické aplikace mohou být applety spouštěny i jiným

mechanismem než jen z příkazové řádky a může jich běžet několik současně na jedné webové stránce. Prostředí DTV samozřejmě není stejné jako prostředí Webu, a proto je třeba applet přizpůsobit podle potřeb digitálního přijímače, výsledkem je rozhraní Xlet [2].

Stejně jako applet, rozhraní Xlet umožňuje spouštění a pozastavování aplikace externím zdrojem, externí zdroj umožňuje i další řízení aplikace (v prostředí DTV se tento zdroj nazývá application manager). Toto rozhraní nalezneme v balíku `javax.tv.xlet`.

Stejně jako třída applet, má Xlet metody, které umožní aby jej application manager mohl inicializovat, spouštět a ukončovat. Mezi appletem a Xletem existuje samozřejmě celá řada rozdílů. Největší z nich je fakt, že Xlet může být pozastaven a znovu obnoven. Tato rozdílná vlastnost Xletu je potřebná pro prostředí DTV, ve kterém je umožněn současný běh více aplikací, přičemž ale hardware umožňuje zobrazovat pouze jednu z těchto aplikací. Ostatní aplikace, které zrovna nejsou viditelné musí být pozastaveny, aby neplýtvaly systémovými prostředky, které jsou potřebné pro běh viditelné, tedy právě zobrazované aplikace [2].



Obr. 6.2: Stavový diagram Xletu

Xlet se může nacházet v jednom ze čtyř stavů – *Loaded*, *Paused*, *Started* nebo *Destroyed*.

Životní cyklus Xletu:

1. Application manager načte hlavní třídu Xletu a vytvoří jeho instanci voláním defaultního konstruktoru, v tomto okamžiku se Xlet nachází ve stavu *Loaded*
2. Pokud se uživatel rozhodne, že spustí Xlet, application manager v přijímači zavolá metodu `initXlet()`. Po proběhnutí inicializace je Xlet ve stavu *Paused* a je připraven ke spuštění.
3. Po skončení metody `initXlet()` zavolá application manager metodu `startXlet()` a díky tomu přejde Xlet ze stavu *Paused* do stavu *Started*. Od tohoto okamžiku je Xlet schopen komunikovat s uživatelem.

4. Během vykonávání Xletu může application manager volat metody `pauseXlet()` a `startXlet()` a měnit tak stav Xletu na *Paused* resp. *Started*.
5. Na konci životního cyklu Xletu zavolá application manager metodu `destroyXlet()`, která změní stav Xletu na *Destroyed* a uvolní systémové prostředky. Po provedení tohoto bodu už Xlet nemůže být znovu spuštěn.

7 Práce s videem a obrazem v STB

Při zpracování videa a obrazu na Set-top boxu je třeba mít na paměti, že drtivá většina STB má omezené možnosti při práci s videem a obrazem. Na počítači je video a grafika renderována softwarem, dokonce i když je video dekódováno hardwarem, renderování provádí software. Díky tomu, že vše je prováděno softwarem, video může být přesouváno nebo může být prováděna změna jeho rozměrů stejně jako u jakéhokoliv jiného obsahu. Protože se jedná pouze o další část dat k renderování, je snadné ji integrovat do systému oken a video a grafika můžou současně koexistovat tak dlouho, dokud je procesor v systému dostatečně rychlý.

Set-top box bohužel ale nepracuje tímto způsobem. Moderní Set-top boxy jsou přístroje s přesně definovanou konstrukcí a hardwarovou specifikací, která je ale značně limitována požadavky na cenu STB od operátorů a zákazníků. Důsledkem tohoto cenového omezení je integrace mnoha komponent do jediného čipu. Obvykle u STB bývá na jednom čipu integrován grafický procesor, MPEG demultiplexer a dekodér a procesor. Tento čip bývá obvykle navržen pouze přímo pro potřeby digitální televize, a proto využití tohoto čipu pro jiné účely je podporováno méně.

Grafický subsystém STB se skládá z několika grafických rovin:

- Rovina pozadí (*Background plane*) – zobrazuje jedinou barvu na pozadí všech ostatních grafických rovin
- Rovina statického obrazu (*Still image plane*) – poskytuje možnost zobrazení statického obrázku na pozadí videa nebo grafiky. Jsou podporovány jen některé obrazové formáty, typicky MPEG-2 I-snímek.
- Video rovina (*Video plane*) – v této rovině je zobrazeno dekódované video. Některé systémy mohou mít více než jednu video rovinu pro podporu funkce obraz v obraze (PiP – Picture In Picture) nebo pro podporu dekódování několika video toků.
- Grafická rovina (*Graphics plane*) – Určena pro grafiku zobrazované aplikace nebo pro titulky. Některé systémy mohou mít více než jednu grafickou rovinu, tyto systémy poskytují různé barevné hloubky a rozlišení.
- Kurzorová rovina (*Cursor plane*) – využívána pro zobrazení hardwarového kurzoru.

Každý přijímač nemusí obsahovat všechny uvedené roviny, některé naopak mohou mít rovin více. Nejnovější čipsety podporují celkem až 5-13 rovin v závislosti na jejich schopnostech a ceně.

Mezi jednotlivými vrstvami existují různá omezení a závislosti. Jedním z těchto omezení je rozdílnost mezi vlastnostmi jednotlivých vrstev na různých hardwarových platformách. Některé hardwarové platformy mohou podporovat více barev, některé pouze základní sadu barev. Tvar pixelů se může mezi jednotlivými vrstvami také měnit, video pixely obvykle nejsou čtvercové jak je tomu u počítačové grafiky. Poměr stran 4:3 představuje u počítačové grafiky rozlišení 800 x 600 pixelů, zatímco rozlišení TV obrazovky s poměrem stran 4:3 v normě PAL je 720 x 625 pixelů. Důsledkem toho je, že grafika nemusí

být přesně zarovnána s videem. Změna parametrů jedné vrstvy může ovlivnit parametry vrstvy jiné na což je nutné myslet při návrhu aplikace.

Pro správné zobrazení aplikace je třeba brát v úvahu i poměr stran obrazu. V dnešní době stále větší počet televizních přijímačů nabízí poměr stran 16:9 oproti stávajícím 4:3. To je další problém pro vytváření aplikace, změna poměru stran má velmi nepěkný dopad na grafickou stránku aplikace. Jednou z možností jak se tomuto problému vyhnout je sledování výstupu Set-top boxu, pokud dojde ke změně poměru stranu aplikace je třeba dynamicky reagovat na tuto změnu a přizpůsobit vzhled aplikace. Toto je extrémní přístup u většiny aplikací je grafický obsah navržen prostě tak, aby nevypadal příliš špatně ani při přepnutí na jiný poměr stran než pro který je aplikace navržena.

7.1 Grafický model v MHP

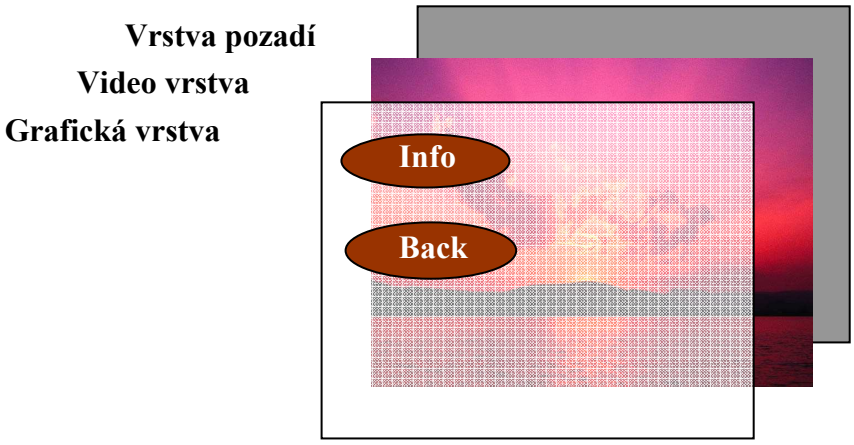
Žádná aplikace není kompletní (nebo dokonce ani funkční) bez možnosti vykreslování na obrazovku. MHP specifikace využívá ořezanou verzi Java AWT (*Abstrakt Windows Toolkit*), což je část Java Core API, která umožňuje tvorbu grafického uživatelského rozhraní v graficky orientovaných systémech (např. Windows), a HAVi (*Home Audio Video Interoperability*) standard, definující rozšíření Javy pro tvorbu uživatelských rozhraní, umožňující aplikacím sdílet obrazovku, i když není přítomen žádný manažer oken. Tyto rozšíření jsou uloženy v balíku `org.havi.ui`.

Grafický model je jednou s nejsložitějších a nejrozsáhlejších částí MHP a to z dobrého důvodu. Existuje totiž velmi mnoho situací a problémů, které je potřeba řešit při práci s grafikou v prostředí DTV. Zde jsou některé z nich, některé už byly zmíněny výše.

- Tvar pixelů. Televizní a video aplikace využívají pixely, nečtvercového tvaru, zatímco počítačová grafika předpokládá pixely čtvercového tvaru. Kombinací těchto dvou modelů vzniká řada problémů pro vývojáře.
- Změna poměru stran z 4:3 na 16:9 nebo 14:9, ať už způsobená televizním signálem nebo uživatelem samotným, může mít velmi nepříjemný dopad na grafiku a obrázky aplikace.
- Průhlednost. Jak udělat grafiku částečně průhlednou, tak aby divák mohl vidět co je pod ní?
- Mapování barevných signálů. Java používá barevný model RGB, kdežto televize YUV.
- Neexistence žádného manažera oken. Manažer oken, je příliš náročný na to, aby mohl být využit na většině dnešních přijímačích DTV, takže aplikace potřebuje jiný způsob jak si zajistit oblast, do které bude moci vykreslovat. To také znamená, že aplikace musí koexistovat současně s ostatními aplikacemi, ale jiným způsobem než je tomu u standardních javovských nebo počítačových aplikací.

7.2 Model zobrazení v MHP

Interaktivní televizní obrazovka může být rozdělena na tři základní části. Vrstva pozadí (*background layer*) umožňuje zobrazovat jednu barvu, někdy i statický obraz. Nad touto vrstvou je video vrstva (*video layer*), která jak již název napovídá zobrazuje video. Video dekodéry v STB mají obvykle velmi omezené možnosti, a proto umožňují zobrazení videa pouze na celou obrazovku, na čtvrtinu obrazovky a na omezené sadě souřadnic. Nad popsanými dvěmi vrstvami je grafická vrstva (*graphics layer*). To je vrstva, do které jsou vykreslovány veškeré grafické operace v MHP. Tato vrstva může mít jiné rozlišení než video vrstva a vrstva pozadí a může používat jiný tvar pixelů (video pixely jsou obdélníkové, grafické pixely obvykle čtvercové). Ačkoliv je každá z těchto vrstev konfigurovatelná samostatně, existuje mezi vrstvami určitá interakce a nakonfigurování jedné vrstvy určitým způsobem může ovlivnit způsob jakým budou nakonfigurovány ostatní vrstvy.

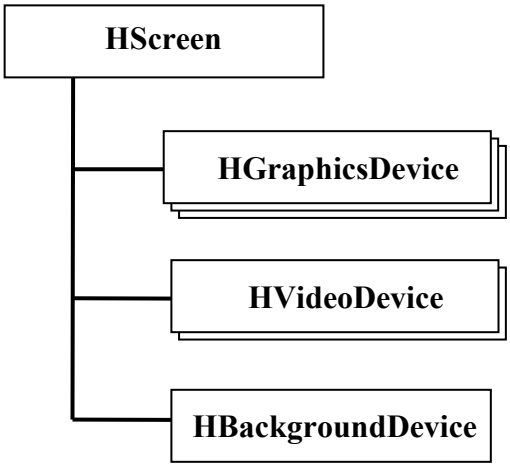


Obr. 7.1: Vrstvy grafického subsystému v MHP

K odstranění výše popsaného problému s interakcí jednotlivých vrstev definuje HAVi (a MHP) třídu `HScreen`. Tato třída reprezentuje fyzické zobrazovací zařízení a každý MHP přijímač musí obsahovat jednu instanci této třídy pro každé připojené zobrazovací zařízení, typicky tedy má každý přijímač jednu instanci třídy `HScreen`. Každá tato třída má několik objektů typu `HScreenDevice`. Tyto objekty představují jednotlivé vrstvy grafického subsystému (viz obr. 7.1). Třída `HScreen` obsahuje tyto podtřídy:

- `HBackgroundDevice`, která reprezentuje vrstvu pozadí
- `HVideoDevice`, reprezentuje video vrstvu
- `HGraphicsDevice`, reprezentuje grafickou vrstvu

Pozadí může být vždy jen jedno, a proto může existovat pouze jediný objekt podtřídy `HBackGroundDevice`, ale funkce jako obraz v obraze umožňují existenci několika objektů podtřídy `HVideoDevice`. Jestliže je podporováno více grafických vrstev, může existovat jeden objekt `HGraphicsDevice` pro každou z nich.



Obr. 7.2: Zařízení tvořící MHP obrazovku

7.2.1 Konfigurace jednotlivých zařízení

Jakmile máme vytvořen objekt některého zařízení pro některou z vrstev, např. `HGraphicsDevice`, můžeme jej konfigurovat využitím instance třídy `HScreenConfiguration` a jejích podtříd. Tímto způsobem můžeme nastavit formát pixelů, poměr stran obrazovky, rozlišení obrazovky a ostatní parametry, které si přejeme změnit. Konfigurační parametry jsou nastaveny pomocí podtřídy třídy `HScreenConfigTemplate`. Zmíněná třída poskytuje mechanismus, který umožní nastavit parametry pro určité zařízení a zároveň se dotáže, zda je uvedená konfigurace možná s ohledem na konfiguraci ostatních zařízení. Třída `HScreenConfigTemplate` obsahuje tři podtřídy, `HBackgroundConfigTemplate`, `HGraphicsConfigTemplate` a `HVideoConfigTemplate`, každá z nich pro odpovídající zařízení MHP obrazovky.

HAVi API využívá třídu `HScreenConfigTemplate` pro definování různých sad preferencí (předvoleb) pro konfiguraci. Každý parametr zařízení se skládá ze dvou částí, hodnoty předvolby a její priority. Jednotlivé předvolby jsou definovány jako konstantní hodnoty ve třídě HScreenConfigTemplate a jejích podtřídách. Příkladem můžou být předvolby ZERO_GRAPHICS_IMPACT a ZERO_VIDEO_IMPACT, které specifikují, že žádná konfigurace by neměla mít vliv na již běžící grafické aplikace nebo přehrávané video. Dalším příkladem předvolby je VIDEO_GRAPHICS_PIXEL_ALIGNED, předvolba která indikuje, zda by pixely v grafické vrstvě a video vrstvě měly být perfektně zarovnány.

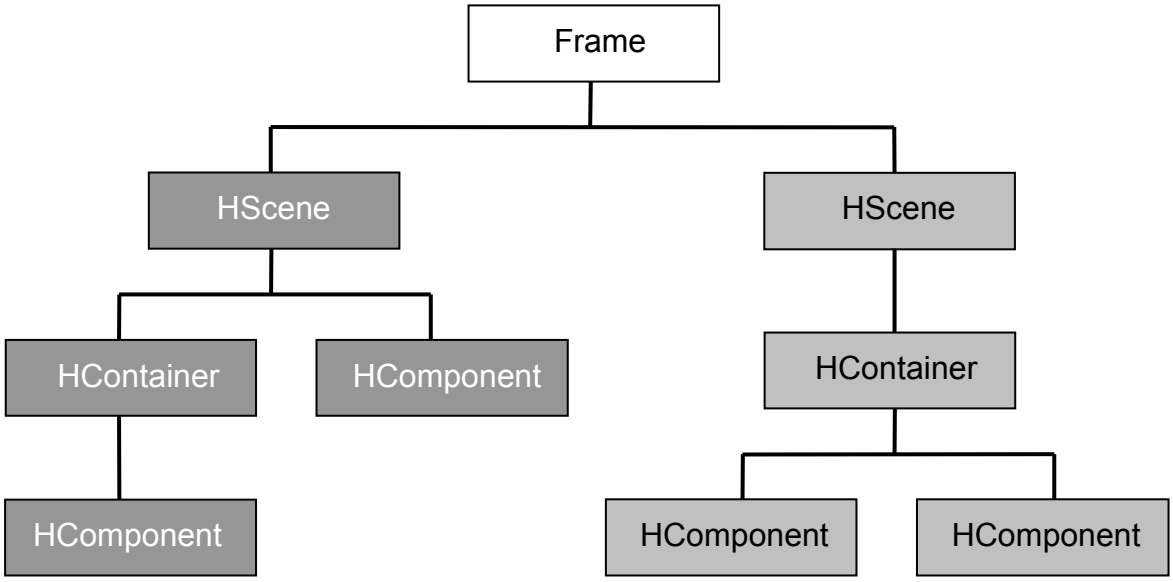
Zajímavější částí parametrů zařízení je priorita, která je přiřazena ke každé předvolbě. Priorita může nabývat jedné z následujících hodnot:

- REQUIRED – tato předvolba musí být dodržena
- PREFERRED – tato předvolba by měla být dodržena, ale je ignorována, pokud je to nezbytné
- UNNECESSARY – aplikace nemá žádnu preferovanou hodnotu pro tuto předvolbu
- PREFERRED_NOT – tato předvolba by neměla mít specifikovanou hodnotu, ale může ji mít, pokud je to nezbytné
- REQUIRED_NOT – předvolba nesmí mít specifikovanou hodnotu

Když aplikace zavolá metodu `HGraphicsDevice.getBestConfiguration()` (nebo ekvivalentní metodu s jiným zařízením), přijímač zkontroluje předvolby specifikované v `HScreenconfigTemplate` a zkusí nalézt konfiguraci, která splňuje všechny předvolby s prioritou REQUIRED a REQUIRED_NOT. Pokud se takovou konfiguraci podaří nalézt, metoda vrátí objekt `HGraphicsConfiguration` (v případě konfigurace grafického zařízení) reprezentující novou konfiguraci, pokud se nalézt nepodaří, vrací metoda NULL.

Jak je patrné z výše uvedeného, konfigurační třídy náležející jednotlivým zařízením obrazovky mají jména odpovídající jménům jednotlivých zařízení (`HBackgroundConfiguration`, `HVideoConfiguration` a `HGraphicsConfiguration`). Je zřejmé, že konfigurace jednotlivých zařízení obrazovky se může měnit za běhu aplikace. Aplikace často potřebuje vědět, že došlo ke změně konfigurace. Třída `HScreenDevice` proto umožňuje aplikaci, aby přidala sama sebe jako posluchače událostí třídy `HScreenConfigurationEvent`, což se provádí zavoláním metody `addScreenConfigurationListener()`. Tyto události informují aplikaci o změnách konfigurace zařízení, které nejsou kompatibilní s předvolbami uvedenými jako argument funkce `addScreenConfigurationListener()`.

7.2.2 Třídy Hscene a HsceneTemplate



Obr. 7.3: Organizace grafických komponent

Protože STB disponuje omezenou pamětí a relativně nízkým výkonem procesoru, není příliš pravděpodobné, že by STB mohl provozovat plně funkčního manažera oken. Pro vývojáře MHP aplikací to znamená, že jako hlavní okno aplikací nemůže být využita třída java.awt.Frame, protože je příliš závislá na manažeru oken na to, aby mohla fungovat bez něj.

Místo této třídy využijeme třídu `org.havi.ui.Hscene`, která je velmi podobná třídě `Frame`, není ale tak rozsáhlá a má některé bezpečnostní omezení, které třída `Frame` nemá. Jedním z největších omezení je fakt, že aplikace nevidí celou AWT hierarchii, nýbrž

pouze vlastní třídu `Hscene` a všechny komponenty v ní umístěné. Třída `Hscene` jedné aplikace není viditelná pro ostatní aplikace a stejně tak tato aplikace nevidí třídy `Hscene` ostatních aplikací. Díky tomuto přístupu je každá aplikace téměř úplně izolovaná od grafických aktivit ostatních aplikací. Na diagramu v obr. aplikace, které vlastní komponenty světlé šedé barvy nemůže manipulovat s tmavě šedými komponentami, protože tyto patří jiným aplikacím. Ve skutečnosti tato aplikace ani neví, že tyto komponenty existují.

Jedním z dalších rozdílů mezi třídami `Hscene` a `Frame` je ten, že aplikace může vytvořit pouze jednu instanci třídy `Hscene`. Jedinou výjimkou, kdy může mít aplikace více než jednu `Hscene` je případ, kdy každá `Hscene` patří do jiné `HScreen` (tzn. na různých zobrazovacích zařízeních).

Vytvoření objektu třídy `Hscene` je prakticky stejné jako u třídy `Frame`, pouze vycházíme z odlišné třídy, `org.havi.ui.HSceneFactory`. Tato třída vytvoří požadovaný objekt `HScene` tak, aby vzniklý objekt byl co nejblíže našim požadavkům a zároveň aby splňoval všechny požadavky dané platformy. Způsob jakým je prováděno vytvoření objektu třídy `HScene` pomocí třídy `HSceneFactory` je skoro stejný jako způsob vytváření objektů zobrazovacích zařízení (viz výše).

Třída `HSceneTemplate` umožňuje specifikovat sadu omezení a parametrů pro vytvářený objekt `HScene`, například rozměry, pozici na obrazovce a plno dalších parametrů a zároveň umožňuje zadat jakou prioritu tyto parametry mají. Každý parametr objektu `Hscene` může mít jednu ze tří priorit – REQUIRED, PREFERRED nebo UNNECESSARY, což umožňuje specifikovat relativní důležitost jednotlivých parametrů. Parametry s prioritou UNNECESSARY jsou při pokusu třídy `HSceneFactory` o vytvoření objektu `HScene` ignorovány. Parametry s prioritou PREFFERED mohou být ignorovány, pokud je to jediný způsob jak vytvořit `HScene` objekt a vyhovět ostatním parametrům, které musí být dodrženy. Parametry s prioritou REQUIRED nesmí být nikdy ignorovány a pokud nemohou být splněny, objekt `HScene` není vytvořen.

Vlastní vytvoření objektu `HScene` probíhá voláním funkce `HSceneFactory.getBestScene()`, která má jako argument objekt `HSceneTemplate`. Pokud mohou být splněny všechny parametry specifikované v `HSceneTemplate`, funkce vrací objekt `HScene`, pokud ne, vrací funkce null. S vytvořeným objektem `HScene` je možno pracovat jako s každým jiným objektem třídy AWT `Container`. Jediným rozdílem je nutnost explicitně odstranit objekt `HScene` po ukončení práce s ním voláním funkce `dispose()`.

Ačkoliv může MHP přijímač samozřejmě provozovat více aplikací současně, neexistuje záruka, že bude možno vždy vytvořit objekt `HScene`, který bude vyhovovat „nejlepší“ sadě parametrů definovaných v `HSceneTemplate`, protože jiná aplikace může vytvořit objekt `HScene` nebo provést nějakou jinou operaci tak, že změní nastavení grafických parametrů nekompatibilním způsobem.

7.3 Souřadnicový systém v MHP

MHP používá tři rozdílné souřadnicové systémy, které poskytují několik rozdílných způsobů jak přesně umístit objekt na obrazovce. Různé API funkce podporují různé souřadnicové systémy, proto by aplikace měla vědět o všech souřadnicových systémech, i když je sama všechny nevyužívá.

7.3.1 Normalizované souřadnice

Normalizovaný souřadnicový systém má počátek v levém horním rohu obrazovky a svoje maximum (1,1) v pravém dolním rohu. Tento systém je abstraktní a umožňuje umísťování objektů relativně vzhledem k poloze ostatních objektů, bez nutnosti specifikace absolutních souřadnic, tudíž je možno umístit objekt doprostřed obrazovky bez znalosti rozlišení obrazovky. Normalizované souřadnice jsou obvykle využívány třídami HAVi.

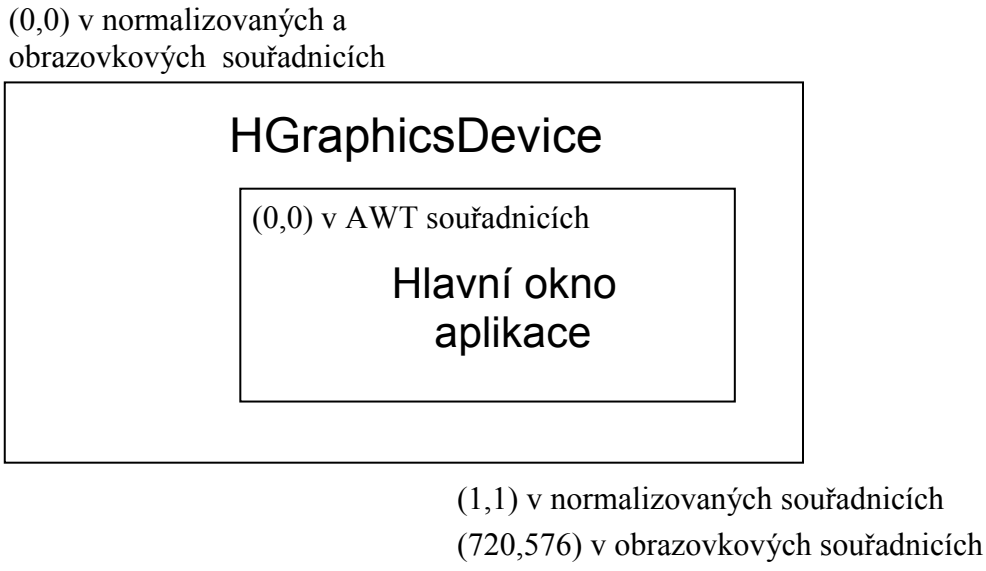
7.3.2 Souřadnice obrazovky

Souřadnicový systém obrazovky má také svůj počátek v levém horním rohu a má svoje maximální souřadnice X a Y v pravém dolním rohu. Tyto maximální hodnoty nejsou pevně dány, ale záleží na zobrazovacím zařízení. Podle MHP specifikace, by maximální hodnoty souřadnic X a Y měly být nejméně (720,576).

Prostor souřadnic obrazovky je definován třídou `HGraphicsDevice`. Tyto souřadnice jsou využívány k určování polohy objektů třídy `Hscene` (viz níže) a k nastavení rozlišení objektů `HScreenDevice`.

7.3.3 AWT souřadnice

Souřadnicový systém AWT je, jak již název napovídá, standardním systémem využívaným třídami Java AWT. Podobně jako souřadnice obrazovky je založen na pixelech, ale oproti práci s celou obrazovkou pracuje pouze s hlavním oknem aplikace. Počátek je v levém horním rohu hlavního AWT kontejneru aplikace (kterým je objekt `Hscene`, viz dále), maximum je v pravém dolním rohu tohoto okna, využívaného AWT.

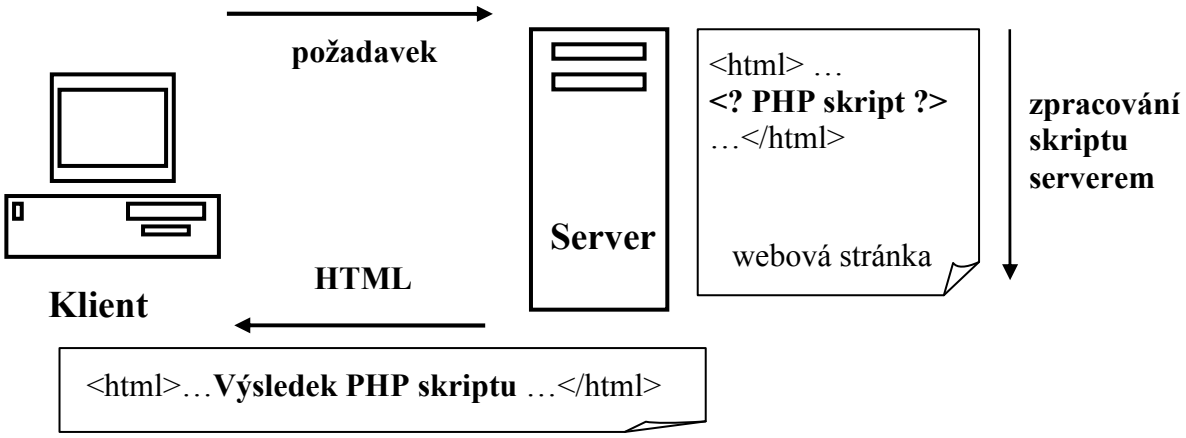


Obr. 7.4: Souřadnicové systémy v MHP

8 PHP (Personal Home Page)

PHP je programovací jazyk, který byl původně vyvinut pro tvorbu dynamických webových stránek. Tento jazyk vznikl v roce 1996 a od té doby prošel značnými změnami. Zkratka PHP dneska znamená Hypertext Preprocesor, místo původního Personal Home Page. PHP pracuje na straně serveru a dovede ukládat a měnit data stránek přímo prostřednictvím prohlížeče. Jak již bylo řečeno, vše se odehrává na straně serveru, kde běží PHP a kde jsou uloženy data webových stránek. Skript napsaný v PHP se nejprve provede na serveru a potom odešle výsledek prohlížeči, tzn. že prohlížeč na straně klienta obdrží pouze „čisté“ HTML. V jazyce HTML chybí oproti PHP akce a dynamika. Tento nedostatek byl kompenzován vznikem skriptovacího jazyku JavaScript, ale i tento jazyk má velká omezení. Jeho největší nevýhodou je nemožnost změny dat a práce s databázemi, což je dáno tím, že JavaScript pracuje na straně webového prohlížeče. PHP umožňuje jak změnu dat, tak práci s databázemi, např MySQL a dalšími [7].

Webová stránka s PHP má nejčastěji příponu php, ale je možné použít i php3, php4, php5 a phtml. Bohužel si PHP nevystačí jen s prohlížečem určité verze (třeba jako HTML nebo JavaScript), a proto je potřeba ho na počítač nainstalovat. Jak již bylo řečeno výše, PHP pracuje na straně serveru. Pro podporu PHP na klientském počítači je proto nezbytné mít nainstalovaný a správně nakonfigurovaný webový server, např. Apache.



Obr. 7.1: Princip funkce PHP skriptů

9 Databáze

Databází rozumíme určitou uspořádanou množinu informací (dat), uloženou na paměťovém médiu. Konkrétně ve světě počítačů a Internetu si lze databázi představit jako množinu dat srovnaných do několika tabulek podle určených pravidel. Tato pravidla a způsob manipulace s nimi je dán použitým databázovým systémem [7].

9.1 Typy databází

Databáze můžeme v zásadě rozdělit na dva základní typy, relační a objektové. V osmdesátých letech způsobily revoluci relační databáze, v letech devadesátých s mohutným nástupem objektově orientovaného programování začaly vznikat i objektově orientované databáze, které si kladou za cíl ulehčit a zrychlit práci s daty. Relační a objektový přístup je zcela rozdílný, ale oba způsoby mají mnoho výhod i nevýhod. V současné době se na databázovém trhu vyskytují tři základní typy databází – relační databáze, objektově-relační a objektové [7].

9.1.1 Relační databáze

Tento typ databáze uchovává data v databázi skládající se z řádků a sloupců. Řádek odpovídá záznamu, sloupce odpovídají atributům (polím v záznamu). Každý sloupec má určen datový typ. Každý atribut (pole) záznamu může uchovávat jedinou hodnotu. Vztahy nejsou explicitní, ale spíše vyplývají z hodnot ve speciálních polích, tzv. cizí klíče (foreign keys) v jedné tabulce, který se rovná hodnotám v jiné tabulce. Relační databáze využívá pro definici dat, řízení dat, přístup k datům a získávání dat jazyk SQL (Structured Query Language) [7].

9.1.2 Objektové databáze

Objektové databáze využívají datového modelu, který má objektově orientované aspekty jako třídy s atributy a metodami, podporují zapouzdření, násobnou dědičnost a podporují abstraktní datové typy.

Objektové databáze kombinují prvky objektově orientovaného programování s databázovými schopnostmi. Dotazovacím jazykem těchto databází bývá např. C++, Java nebo Smalltalk [7].

9.1.3 Objektově-relační databáze

Databázové systémy těchto typů, se snaží sjednotit rysy jak relačních, tak objektových databází. Využívají datový model tak, že „přidávají objektovost do tabulek“. Všechny trvalé informace jsou stále v tabulkách, ale některé položky mohou mít bohatší datovou strukturu, nazývanou abstraktní datové typy (ADT). ADT je datový typ, který vznikne zkombinováním základních datových typů. Tyto databáze podporují rozšířenou verzi SQL – SQL3 [7].

9.2 Databázové systémy

Jedním z nejznámějších databázových systémů je MySQL, multiplatformní databáze, komunikující s okolím pomocí jazyka SQL. MySQL systém je snadno implementovatelný,

lze jej instalovat v MS Windows, Linuxu i dalších operačních systémech, má vysoký výkon a jedná se o volně šířitelný software. Především díky těmto přednostem má vysoký podíl na v současné době používaných databázích. Velmi oblíbená a často nasazovaná je kombinace MySQL, PHP a Apache jako základní software webového serveru.

9.3 SQL

SQL je standardizovaný dotazovací jazyk používaný pro práci s daty v relačních databázích (databázový systém založený na relačním modelu, data jsou uspořádána do tabulek – relací, nad kterými jsou definovány přípustné operace). V době vzniku SQL nabývaly relační databáze stále více na významu a proto bylo nutné jejich jazyk standardizovat, vznikl tak SQL standard označovaný jako SQL-86, označovaný podle roku vzniku. V dalších letech se ukázalo, že SQL-86 obsahuje některé nedostatky a proto vznikl nová standard SQL-93, který reaguje na potřeby nejmodernějších databází s objektovými prvky [7].

10 Kryptografie

Kryptografie neboli šifrování je nauka o metodách utajování smyslu zpráv převodem do podoby, která je čitelná jen se speciální znalostí. Někdy je pojem obecněji používán pro vědu o čemkoli spojeném se šiframi jako alternativa k pojmu kryptologie. Kryptologie zahrnuje kryptografii a kryptoanalýzu, neboli luštění zašifrovaných zpráv. Kryptoanalýza je věda zabývající se metodami získávání obsahu šifrovaných informací bez přístupu k tajným informacím, které jsou za normálních okolností potřeba. Tzn. především získání tajného klíče. V netechnickém kontextu je používán tento termín obecně pro prolamování kódu. Je vlastně opakem kryptografie, která šifry vytváří.

Kryptografické metody obecně využívají tzv. "klíč", pomocí kterého tajná data zašifrují a posléze opět dešifrují. Současně některé metody umožňují nebo i vynucují použití více klíčů různých pro šifrování a dešifrování.

Utajení dokumentu se skládá z dvou částí. Utajení šifrovací metody a utajení klíče. Zásadní je zejména utajení klíče, jelikož metod není takové množství, aby nemohlo dojít k jejímu odhalení. Často se tedy ani k utajení vlastní metody nepřistupuje a utajení zajišťuje jen klíč. Podle použití způsobu práce s klíči se kryptografické metody dělí na symetrické a asymetrické [10].

10.1 Asymetrické metody kryptografie

První skupinou kryptografických metod je asymetrická kryptografie. Název těchto metod je odvozen od principu jejich funkce. Pro šifrování a dešifrování je použit jiný klíč. Šifrování je prováděno veřejným klíčem a následné dešifrování se provádí klíčem soukromým. Tyto dva klíče se souhrnně nazývají párem klíčů a bývají generovány současně.

Pokud majitel těchto klíčů chce, aby mu někdo mohl posílat šifrované zprávy, musí dát k dispozici svůj veřejný klíč. Odesílatel zašifruje zprávu tímto klíčem a příjemce si zašifrovanou zprávu dekóduje pomocí soukromého klíče, který uchovává v tajnosti.

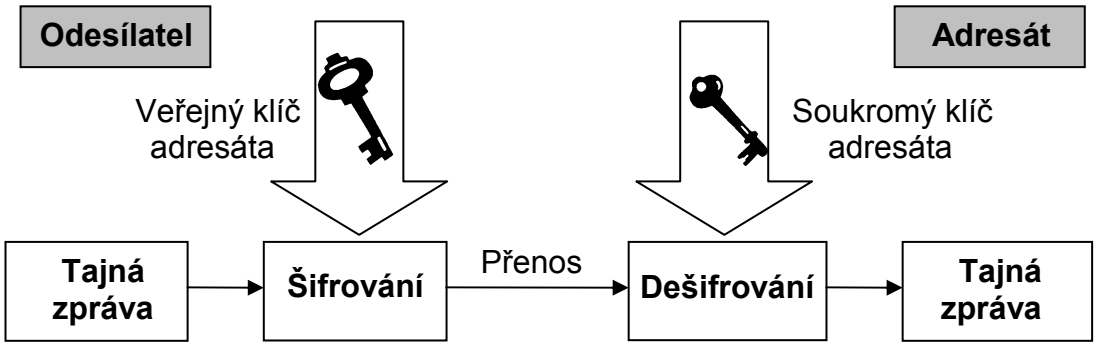
Je zřejmé, že veřejný a soukromý klíč spolu musí být matematicky svázány, nezbytnou podmínkou však je nemožnost spočítat soukromý klíč z veřejného. To je zajištěno mechanismem, kterým asymetrické šifry pracují. Ten je založen na jednocestných funkcích, což jsou operace, které lze snadno provést jedním směrem, ze vstupu spočítat výstup, z výstupu je však velmi obtížné nalézt vstup. Příkladem může být například násobení, které je relativně snadno proveditelné i pro velká čísla, a faktorizace (rozklad součinu na činitele), která je velmi obtížná.

Mezi nejznámější asymetrické metody patří algoritmy DH (Diffie-Hellman), RSA (Rivest Shamir-Aleman) a DSA (Digital Signature Algorithm).

Výhody a nevýhody asymetrické kryptografie:

- Hlavní výhodou je to, že není třeba nikam posílat soukromý klíč, a tak nemůže dojít k jeho vyzrazení
- Pro komunikaci mezi více osobami stačí jeden pár klíčů pro každou osobu

- Nevýhodou je rychlost, tyto metody jsou řádově 1000x pomalejší než metody symetrické (viz níže)
- Další nevýhodou je nutnost ověřit pravost klíče, tzn. bezpečně identifikovat majitele klíče. Pro tyto účely slouží certifikační úřady



Obr. 10.1: Princip funkce asymetrických metod kryptografie

10.1.1 RSA

Jednou z nejznámějších asymetrických metod kryptografie je RSA (iniciály autorů, Rivest, Shamir, Adleman). Jedná se o první algoritmus určený k „podepisování“, tzn. ověřování pravosti dokumentu, a šifrování. Používá se dodnes. Při použití klíče s dostatečnou délkou je tato asymetrická metoda považována za bezpečnou. Za dostatečnou délku klíče se v dnešní době považuje 1024b a více [10].

10.1.1.1 Princip funkce

Jak již bylo zmíněno výše asymetrické metody jsou založeny na jednocestných funkcích, které lze snadno provést pouze v jednom směru, opačný postup je výpočetně velmi náročný.

Bezpečnost RSA je založena na předpokladu, že faktorizace (rozložení čísla na součin prvočísel) je velmi obtížná úloha. Například násobení dvou velkých prvočísel $p.q = n$ je elementární úloha. Oproti tomu zjištění činitelů p a q z čísla n je v rozumném čase prakticky nemožné zjistit, neboť není znám žádný algoritmus faktorizace, který by pracoval v polynomiálním čase vůči velikosti zápisu čísla n.

10.1.1.2 Tvorba páru klíčů

1) Zvolí se dvě různá velká prvočísla p a q. Za velká se považují čísla alespoň patnáctimístná.
2) Spočítá se jejich součin $n = p \,.\, q$.
3) Spočítá se hodnota Eulerovy funkce $\varphi(n) = (p - 1) \,.\, (q - 1)$.
4) Zvolí se celé číslo e menší než $\varphi(n)$, které je s $\varphi(n)$ nesoudělné.
5) Nalezneme číslo d tak, aby platilo $d \,.\, e \equiv 1 \pmod{\varphi(n)}$.
6) Jestli e je prvočíslo, tak $d = (1 + r * \varphi(n) / e)$, kde $r = [(e - 1) \,.\, \varphi(n) \,\hat{}\, (e - 2)]$

Dvojice (n, e) tvoří veřejný klíč, přičemž n se označuje jako modul, e jako šifrovací či veřejný exponent. Dvojice (n, d) tvoří klíč soukromý, d se označuje jako dešifrovací či

soukromý exponent. V praxi se klíče uchovávají v upravené formě, která umožňuje rychlejší zpracování.

Veřejný klíč, jak již z názvu napovídá, může být zveřejněn, soukromý zůstává uchován v tajnosti u příjemce šifrovaných zpráv. Každý, kdo potřebuje šifrovaně komunikovat s majitelem soukromého klíče, zašifruje data odpovídajícím veřejným klíčem, příjemce data dekóduje svým soukromým klíčem [10].

10.1.1.3 Šifrování zprávy

Zprávu k odeslání M je nejprve potřeba převést nějakým předem dohodnutým postupem na číslo m ($m < n$). Šifrovaným textem odpovídajícím této zprávě pak je číslo

$$c = m^e \bmod n \tag{10.1}$$

Takto zašifrovaná zpráva (kryptogram) může být poslána nezabezpečeným kanálem.

10.1.1.4 Dešifrování zprávy

Původní zpráva m se z kryptogramu získá následujícím výpočtem

$$m = c^d \bmod n \tag{10.2}$$

10.2 Symetrická kryptografie

Všechny metody symetrické kryptografie (někdy nazývaná konvenční kryptografie) využívají stejný klíč k šifrování i dešifrování tajné informace.

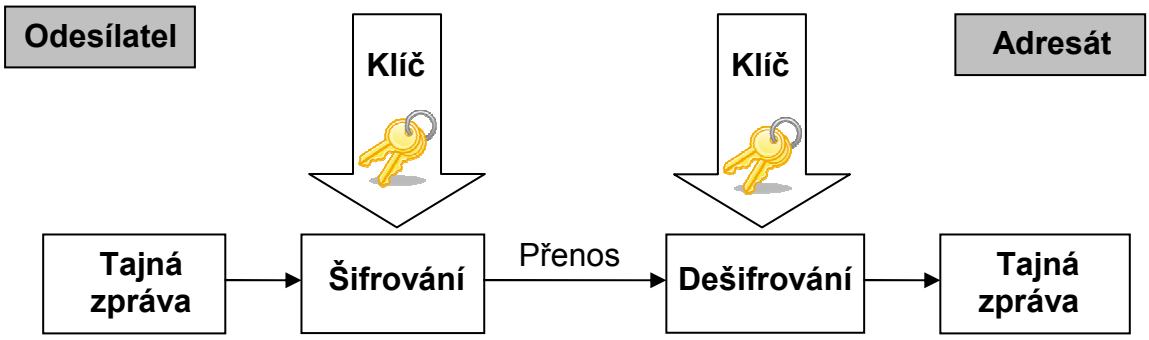
Vstupem je tajný text ze stanovené abecedy a klíč. Šifrovací funkcí se pomocí klíče tajný text zašifruje a je odeslán adresátovi. Ten přijatý text dešifruje stejným klíčem, jakým byl text zašifrován. Je tedy důležité zajistit bezpečný přenos klíče tak, aby se nedostal do nepovolaných rukou.

Pro šifrování se používají funkce, u kterých platí, že při znalosti vstupního a šifrovaného textu je velmi obtížné vygenerovat klíč. Vlastní šifrování a dešifrování pomocí tohoto klíče je rychlá záležitost. Obtížnost případného zjištění klíče záleží zejména na délce klíče samotného. Šifrovaná zpráva musí odolat útoku hrubou silou, při kterém jsou postupně zkoušeny všechny možné klíče. Pokud je tedy klíč dlouhý například 8 bitů existuje 256 možných klíčů. V dnešní době se používají běžně klíče délky 128 nebo 192 bitů.

Mezi symetrické metody kryptografie patří algoritmus DES (délka klíče 56 bitů), jeho nástupce 3DES (délky klíče 3 x 56 b = 168 bitů), Blowfish (proměnná délka klíče až 256 bitů), AES (délka klíče 128, 192 nebo 256 bitů) a další.

Výhody a nevýhody symetrické kryptografie:

- Hlavní výhodou je vysoká rychlost šifrování a dešifrování
- Největší nevýhodou je, že pokud chceme s někým tajně komunikovat, musíme si předem bezpečným kanálem předat klíč.
- Další nevýhodou je počet klíčů. Chceme-li zajistit, aby mohli tajně spolu komunikovat dvě osoby, potřebují jeden klíč, pro tři osoby jsou to již tři klíče, pro čtyři osoby šest klíčů, obecně počet klíčů = n * (n-1) / 2, kde n je počet osob.



Obr. 10.2: Princip funkce symetrických metod kryptografie

10.2.1 AES (Advanced Encryption Standard)

AES je současným průmyslovým standardem pro symetrické šifrování. Někdy bývá nazýván též Rijndael, podle jmen svých autorů: Joan Daemen a Vincent Rijmen. Rijndael je původní název algoritmu, který standard AES používá, byl přijat americkým Národním institutem pro standardizaci a technologie (NIST) v roce 2001. V této roli nahradil již zastaralý algoritmus DES (Data Encryption Standard), který byl v roce 1999 prolomen během necelých 23 hodin. V praxi tedy názvy „Rijndael“ a „AES“ odkazují na totéž.

Protože AES patří mezi symetrické metody kryptografie, je zřejmé, že používá stejný klíč pro šifrování a dešifrování. Délka klíče je 128, 196 nebo 256 bitů [10].

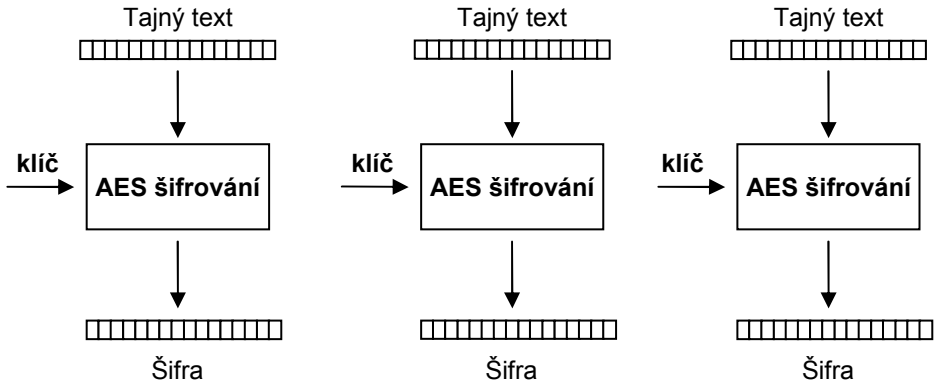
10.2.1.1 Bezpečnost

Pro zajištění bezpečné komunikace je nutná výměna klíče důvěrnou cestou mezi komunikujícími stranami. V praxi se pro tento účel často využívají asymetrické metody, například RSA. Samotná data se zašifrují symetrickým algoritmem a klíč se pro přenos zašifruje algoritmem asymetrickým. Bezpečnost šifry také závisí na kvalitě použitého klíče, musí být dostatečně komplexní a dostatečně náhodný, jinak je šifra snadno prolomitelná.

10.2.1.2 Módy AES

Rijndael je blokový algoritmus, který je aplikován na data s pevně danou délkou. Pokud jsou šifrovaná data delší, zpracovávají se po jednotlivých blocích. Pokud jsou data kratší (většinou se jedná o poslední blok se zbytkem dat), je potřeba tato data doplnit na odpovídající délku. Tomuto doplnění na odpovídající délku se říká „padding“. Existuje pro něj několik algoritmů, od primitivního doplnění nulami po složitější schémata.

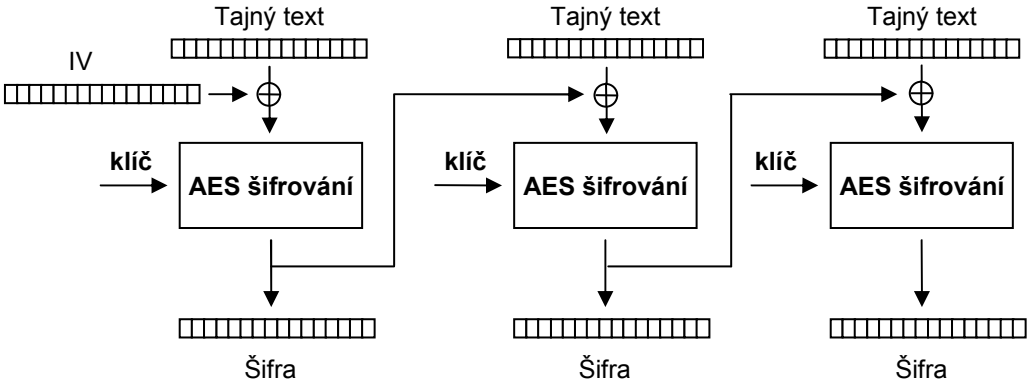
Jedním z módů ve kterém může AES pracovat je ECB (Electronic Codebook). Šifra je aplikována na jednotlivé bloky tak jak jdou za sebou. Výsledkem tohoto přístupu je, že stejné bloky otevřeného textu jsou zašifrovány vždy stejně. Vzhledem k poměrně malé velikosti bloků to může znamenat, že ve výsledné zprávě budou patrná stejná schémata jako v původním dokumentu. Schéma funkce šifry AES v módu ECB je na obrázku 10.3.



Obr. 10.3: AES šifrování v módu ECB

V praxi se spíše používají algoritmy, které toto nebezpečí odstraňují. Nejjednodušším z nich je CBC (Cipher Block Chaining). Princip jeho funkce je, že se před zašifrováním blok otevřeného textu XORuje s předcházejícím blokem zašifrovaného textu. To znamená, že jednotlivé bloky jsou na sobě závislé a při dešifrování konkrétního bloku je třeba dešifrovat i všechny předchozí.

Při šifrování prvního bloku v módu CBC vzniká problém. Čím Xorovat, když neexistuje žádný předcházející blok. Tento problém je vyřešen vygenerováním jednoho bloku náhodně, tento blok je přidán k datům jako „nultý blok“. Tomuto bloku se pak říká inicializační vektor (IV). Tento blok se používá pouze k dešifrování prvního bloku a pak se zahodí.



Obr. 10.4: AES šifrování v módu CBC

Ve vytvářené aplikaci je použit mód EBC, tzn. mód bez inicializačního vektoru a bez závislosti následujícího bloku na bloku předcházejícím. Jak již bylo zmíněno výše, problémem tohoto módu je, že stejné bloky otevřeného textu jsou šifrovány vždy stejně. Tato skutečnost se ale v navrhované aplikaci vůbec neprojeví, protože veškerá šifrovaná data mají maximální velikost odpovídající jednomu bloku (při použití 128 bitového klíče je maximální délka bloku 128 b = 16 B) [10].

11 Aplikace pro online sázení

11.1 Popis funkce

Aplikace je vytvořena pro běh na multimediální platformě MHP, v prostředí DVB. Hlavním účelem aplikace je umožnit interaktivní sázení na sportovní zápasy vysílané prostřednictvím DVB-T. Jak již bylo zmíněno výše, aplikace je součástí televizního vysílání, a divák ji může spustit prostřednictvím dálkového ovladače STB. Po spuštění aplikace je uživateli umožněno vsadit na sportovní zápas, který bude vysílán. Uživatel zadá své uživatelské jméno a heslo a aplikace se prostřednictvím zpětného kanálu připojí k databázovému serveru na kterém je uložena databáze uživatelů. PHP skript, který je uložený na serveru ověří uživatelské jméno a heslo podle údajů v databázi mySQL a odešle aplikaci informaci o výši uživatelova kreditu. Nyní si uživatel může vsadit částku podle svého uvážení a odeslat informaci serveru, kde se informace o výši sázky uloží do databáze. Po skončení sportovního přenosu musí být do databáze uloženy informace o výsledku zápasu. Uživatel si znovu spustí aplikaci, ta se opět připojí k serveru a podle výsledku zápasu uloženého v databázi vyhodnotí všechny sázky. Informace o výši případné výhry se zobrazí v okně aplikace (průběh komunikace viz obr. 11.1).

Pro vytvoření aplikace byl využit programovací nástroj NetBeans 5.5.1. Aplikace byla vyvíjena pomocí softwaru simulujícího prostředí MHP reálného set-top boxu – IRT MHP RI 1.0.2 – 3final, testování probíhalo na STB STRONG SRT 5510.

11.2 Komunikace mezi aplikací a serverem

Aplikace komunikuje se serverem prostřednictvím zpětného kanálu Set-top boxu. Aplikace je navržena pro použití se STB obsahujícími Ethernetový zpětný kanál. Samotné spojení je realizováno pomocí třídy `URLConnection` z balíku `java.net`. Pro správnou komunikaci je na serveru nezbytná podpora http (Hyper Text Transfer Protocol), php a mySQL.

11.2.1 Síťové API Javy – balík `java.net`

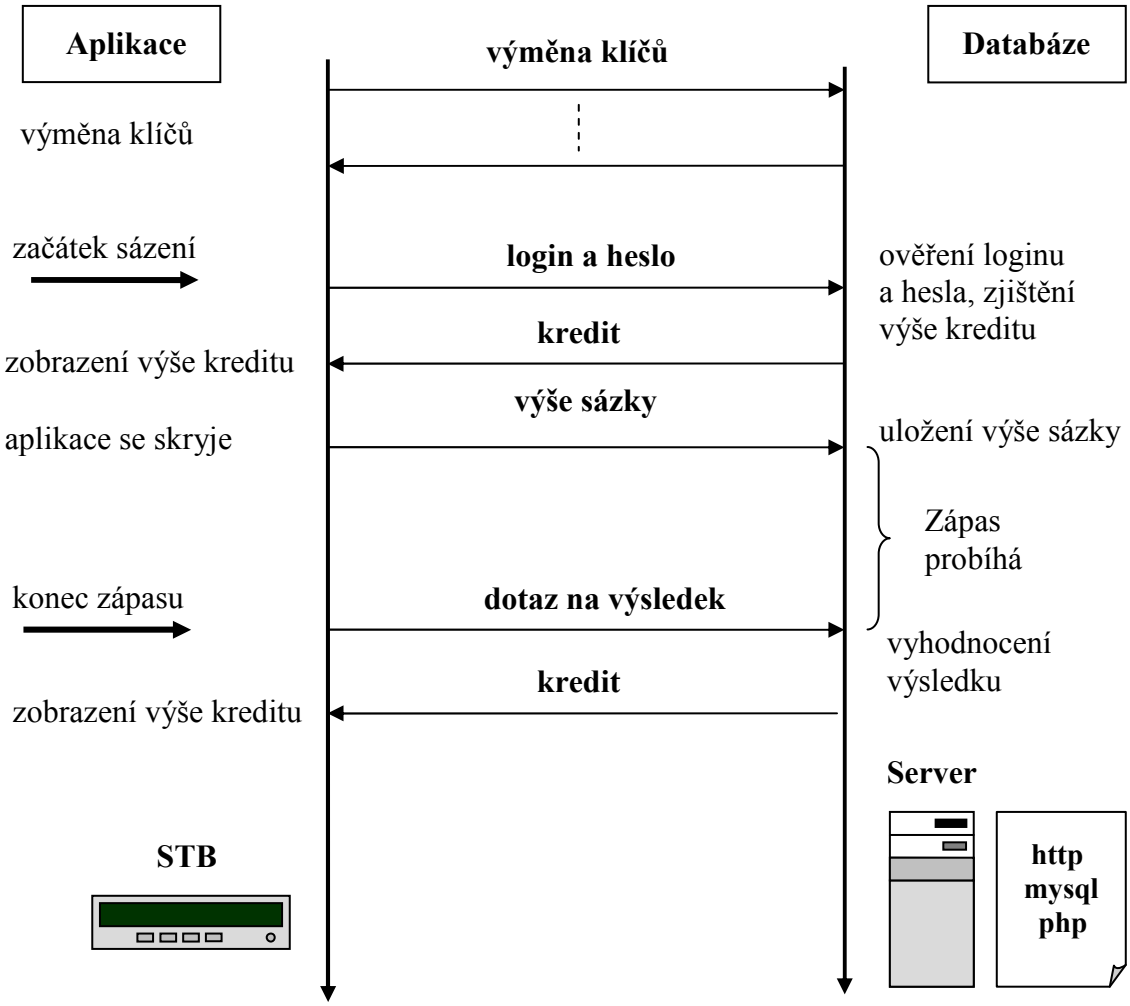
Balík `java.net` obsahuje veškeré třídy potřebné pro tvorbu síťových aplikací. V současné době je podporována komunikace v prostředí Intranet / Internet na bázi rodiny protokolů TCP/IP. Kromě nízkoúrovňové komunikace založené na API používající soketů je k dispozici řada služeb zpřístupňujících obsah síťových zdrojů na základě jejich identifikace pomocí URL (Uniform Resource Locator) [8].

11.2.2 URL

URL reprezentuje ukazatel na konkrétní zdroj v síti. Jeho formát je následující: *služba://uživatel:heslo@host:port/cesta#reference*

Reference není ve skutečnosti součástí URL, interpretuje ji klient jako část dokumentu, o kterou má uživatel zájem. Data odkazována zdroje mohou být statická (v souboru) nebo generována dynamicky programem (např. PHP), jak je tomu v případě naší komunikace. URL

mohou být zadány absolutně nebo relativně vzhledem k síťovému zdroji, ve kterém se vyskytují.



Obr. 11.1: Zjednodušené schéma komunikace aplikace se serverem

11.2.3 Třída `URLConnection`

Základem komunikace mezi aplikací a serverem je třída `URLConnection` z výše popsaného balíku `java.net`. Jedná se o abstraktní nadtřídu všech tříd reprezentujících komunikační spojení mezi aplikací a zdrojem. Získá se voláním `openConnection()` nad instancí příslušného URL. Ve skutečnosti se vrací instance podtřídy specializované na příslušný protokol (např. `HttpURLConnection`). Třída `URLConnection` se používá k zápisu do nebo čtení z přiřazeného zdroje jednoznačně určeného pomocí URL. V rámci jednoho objektu třídy URLConnection je možné realizovat pouze jediný dotaz a odpověď mezi komunikujícími stranami. Výměna informací mezi aplikací a zdrojem probíhá pomocí metod:

- `getInputStream()` – čtení dat aplikací ze zdroje (ze serveru)
- `getOutputStream()` – zápis dat do zdroje (na server)

Z výše uvedeného je patrné, že veškeré řízení komunikace má na starost aplikace běžící v STB. Přenos dat na server provádí aplikace zápisem do výstupního proudu (OutputStream) a k získání dat ze serveru čte ze vstupního proudu (InputStream).

11.3 Zpracování dat serverem

Aplikace běžící v STB neobsahuje žádné uživatelské informace. Všechny informace musí být dostupné odkudkoliv z Internetu, a proto jsou všechny informace sloužící k ověření uživatele včetně jeho kreditu pro sázení uloženy na serveru v databázi mySQL.

Pro zvolenou formu komunikace mezi aplikací a serverem je zapotřebí, aby na serveru byla podpora http protokolu. Pro zpracování příchozích informací od aplikace a interakci s databází mySQL je využíván výše popsaný jazyk PHP, který musí být na serveru také nainstalován. Protože skript komunikující s aplikací obsahuje třídu pro RSA šifrování, je nezbytné, aby na serveru běželo PHP ve verzi 5 a vyšší. Nižší verze PHP nepodporují třídy. Posledním nezbytným požadavkem na PHP pro správný chod skriptu na serveru je existence rozšíření *mcrypt*, které poskytuje kryptografické funkce a je skriptem využíváno.

11.3.1 PHP skript na serveru

Na serveru je uložen skript *kom.php* napsaný v jazyce PHP, který zajišťuje veškerou interakci mezi serverem a aplikací. Zároveň spravuje databázi mySQL ve které jsou uloženy informace o probíhajícím zápase, informace o jednotlivých uživatelích, aplikacích, které se serverem právě komunikují.

Jak již bylo zmíněno výše, aplikace komunikuje se serverem pomocí třídy `URLConnection` respektive pomocí http protokolu. Každý dotaz odeslaný aplikací pomocí http protokolu na server představuje jediný řetězec složený ze jmen jednotlivých proměnných a jejich hodnot. Metoda POST, pomocí které protokol http předává data serveru, naplní pole `$_POST`, z kterého lze ve skriptu jednoduše načíst jednotlivé proměnné z příslušných položek pole , `$_POST['název_proměnné']`.

Skript v PHP představuje posloupnost příkazů, které vykonávají určitý úkol. Při každém http dotazu aplikace (např. přihlášení uživatele, uskutečnění sázky) je skript zavolán znovu, což znamená průchod skriptem opět od začátku a vynulování všech proměnných. Z tohoto poznatku vyplývá nemožnost uložit jakákoliv data do proměnné, kde by byla uchována do dalšího volání skriptu, a také jistá bezestavovost, skript nemůže „počkat“ v určitém stavu, ale proběhne vždy od začátku. Tomu musí být přizpůsobena struktura skriptu.

Data, která je třeba zachovávat mezi jednotlivými voláními skriptu (např. AES klíč aplikace), jsou ukládána do databáze, ze které mohou být kdykoliv opětovně načtena. Stav ve kterém se komunikace s příslušnou aplikací právě nachází (přihlašování, sázení, atd.) je určován podle obsahu pole `$_POST`. Hodnotou jsou totiž naplněny jen proměnné předávané metodou POST. Ostatní jsou nulové.

11.3.1.1 Připojení k databázi

Pro testování aplikace byl zvolen server *php5.cz*, který splňuje všechny požadavky pro správný chod skriptu (viz výše). V každém kroku komunikace mezi aplikací s serverem je nějakým způsobem využívána mySQL databáze, proto je potřeba, aby se skript při každém jeho zavolání k této databázi připojil. Děje se tak prostřednictvím funkce `mysql_pconnect(DB_host, DB_klient, DB_heslo)`. Parametry této funkce představují databázový server, přihlašovací jméno k databázi a heslo. Příslušná databáze je pak vybrána pomocí volání funkce `mysql_select_db(název_databáze)`.

11.3.1.2 Třída RSA

Tato třída představuje implementaci RSA algoritmu. Ze zadaných prvočísel *p* a *q* vygeneruje pár klíčů, pomocí metody `generate_keys (p, q)`, šifrovací exponent je zvolen „napevno“ na běžně užívanou hodnotu 65537. Vygenerované klíče jsou uloženy v poli *$keys*, kde *$keys[0]* obsahuje modul *n*, *$keys[1]* obsahuje šifrovací exponent *e* a *$keys[2]* dešifrovací exponent *d*.

Třída obsahuje metody pro šifrování i dešifrování zpráv RSA algoritmem, z pohledu serveru je ale potřebná pouze funkce na dešifrování `decrypt(c, d, n)`, kde *c* je kryptogram, *d* a *n* představují soukromý klíč. Základním kamenem této funkce je realizace vztahu 10.2, která provádí dešifrování přijaté zprávy. Přijatý kryptogram, který má být dešifrován představuje velké číslo, které je ale uloženo jako číselný řetězec. Pro správné dekódování je tedy nezbytná správná interpretace číselného řetězce a jeho konverze do ASCII znaků.

11.3.1.3 Odeslání veřejného klíče aplikaci

Odeslání veřejného RSA klíče aplikaci je prvním krokem komunikace mezi aplikací a skriptem. Aplikace pošle žádost o odeslání klíče, skript identifikuje tuto žádost díky nenulové hodnotě proměnné *$_POST[`request`]* a pomocí třídy RSA vygeneruje RSA klíče. Parametry *p* a *q* potřebné ke generování klíče jsou zadány „natvrdo“, což vede ke generování jednoho a totéž klíče pro server. Veřejný klíč je aplikaci odeslán v jednom řetězci, který obsahuje šifrovací exponent *e* a modul *n*.

11.3.1.4 Přijetí AES klíče

Dalším krokem komunikace je výměna AES klíče mezi aplikací a serverem. Příchozí AES klíč je zašifrován veřejným klíčem serveru. Skript si samozřejmě nemůže veřejný klíč uchovávat v proměnné mezi jednotlivými voláními a ukládání do databáze by bylo zbytečně komplikované, proto je klíč znovu vygenerován. Díky fixně zadaným parametrům *p* a *q* bude klíč vždy stejný pro všechny aplikace, což je žádoucí. Díky vysokému výpočetnímu výkonu serveru nepředstavuje generování páru RSA klíčů žádné výraznější zpoždění.

Následuje dešifrování přijatého AES klíče a jeho uložení do databáze (tabulka *aplikace*) s číslem příslušné aplikace reprezentovaným proměnnou *$appID*. Čísla slouží k identifikaci AES klíčů jednotlivých aplikací (každá aplikace si vygeneruje vlastní AES klíč, který pošle severu). Tato čísla jsou aplikacím přidělována od 1, vždy nejmenší volné číslo, po skončení

komunikace s příslušnou aplikací je totiž klíč s číslem aplikace odstraněn z databáze. Po uložení AES klíče do databáze je přidělené číslo aplikace oznámeno aplikaci, se kterou právě probíhá komunikace. Od této chvíle je již komunikace obousměrně šifrována AES šifrou.

11.3.1.5 Přihlášení uživatele

Aplikace požadující přihlášení uživatele k databázi předává skriptu proměnné obsahující uživatelské jméno, heslo a číslo aplikace. Skript z databáze načte odpovídající AES klíč a dešifruje přijaté uživatelské jméno a heslo. Dešifrované údaje porovná s údaji v databázi uživatelů (tabulka *hraci*) a oznámí aplikaci, zda byl uživatel nalezen nebo ne. V případě, že ano, odešle aplikaci informaci o výši uživatelova kreditu. Všechny údaje jsou chráněny AES šifrou.

11.3.1.6 Uskutečnění sázky

Skriptu jsou aplikací předány proměnné obsahující uživatelské jméno, výši sázky, uživatelův tip na zápas a číslo aplikace pro načtení příslušného AES klíče. Po dešifrování je do tabulky uživatelů v databázi příslušnému uživateli zaznamenána výše sázky, která je odečtena z uživatelova kreditu, a tip na který vsadil.

11.3.1.7 Odeslání výsledků aplikaci

Posledním krokem v komunikaci je odeslání výsledku zápasu a výsledku sázky aplikaci. Aplikace po skončení zápasu pošle skriptu žádost o výsledky, která je provedena předáním proměnné *AESresult*, uživatelského jména a čísla aplikace do skriptu. Díky nenulové hodnotě proměnné *AESresult* v poli *$_POST* skript pozná, že má aplikaci poslat výsledky. Přijaté údaje jsou dešifrovány pomocí AES klíče načteného z databáze podle čísla aplikace. Z tabulky *zápasy* v databázi jsou načteny výsledky zápasu, kurzy na možné výsledky, skóre a z něho vyplývající výherní tip. Na základě těchto údajů a příslušných kurzů je spočítána výše případné výhry a výsledky jsou v šifrované podobě odeslány aplikaci. Posledním krokem je odstranění AES klíče z tabulky *aplikace* v databázi.

11.4 Databáze

Veškeré údaje, se kterými pracuje skript na serveru a které jsou na server posílány aplikací běžící na STB, jsou ukládány do databáze. Databáze na serveru obsahuje tři tabulky, *aplikace*, *hraci* a *zapasy*.

11.4.1 Tabulka klíčů

Tato tabulka, pojmenovaná *aplikace*, obsahuje AES klíče jednotlivých aplikací, které právě se serverem komunikují. Každé aplikaci je přiřazeno identifikační číslo *ID*, pod kterým je příslušný klíč uložen. Obrázek 11.2 znázorňuje tabulku *aplikace* ve stavu, kdy se serverem komunikují tři aplikace. Klíč každé aplikace je v tabulce uložen, dokud nedojde k odeslání výsledků zápasu a sázek aplikaci, poté je záznam z tabulky odstraněn.

ID	AESkey
1	516375c20353b054
2	1b3f3234e458633f
3	ad3b403a3f3f4911

Obr. 11.2: Tabulka AES klíčů jednotlivých aplikací

11.4.2 Tabulka hráčů

V tabulce *hraci* jsou uloženy informace o jednotlivých uživatelích, o jaké údaje se jedná je patrné z tabulky na obr. 11.3. *Login* a *Heslo* mohou být položky dlouhé maximálně 16 znaků, *Kredit* musí být celočíselný, stejně jako *Sazka*, která musí být samozřejmě nižší nebo rovna výši kreditu. Tip může nabývat pouze hodnot 1, 0, 2, 10 nebo 02. Toto omezení vstupních hodnot, ale řeší přímo aplikace na STB.

ID	Login	Heslo	Jmeno	Prijmeni	Kredit	Sazka	Tip
1	xholik00	mojeheslo	Tomas	Holik	535	100	1
2	xlogin00	hesloxxx	Jan	Novak	515	0	0

Obr. 11.3: Tabulka uživatelů

11.4.3 Tabulka zápasů

V tabulce *zapasy* jsou uloženy informace o aktuálním zápase, na který je možno přes aplikaci na STB sázet. Jednotlivé údaje této tabulky jsou na obr 11.4. Po skončení zápasu dojde k naplnění této tabulky a následnému stažení výsledků aplikací. Nejdůležitější je položka *Result*, která představuje vítězný tip, tato položka je porovnána s položkou *Tip* v tabulce hráčů a na základě výsledku této operace je rozhodnuto o výsledku sázky. Výše výhry je reprezentována součinem odpovídajícího kurzu a vsazené částky.

Id	Home	Away	Score_home	Score_away	Course_1	Course_0	Course_2	Course_10	Course_02	Result
1	Sparta	Slavia	1	0	1.15	1.3	1.4	1.1	1.35	1

Obr. 11.4: Tabulka zápasů

11.5 Bezpečnost komunikace

Veškerá komunikace mezi aplikací pro online sázení a databázovým serverem probíhá prostřednictvím Internetu. Taková komunikace je jednoduše napadnutelná a může dojít k odposlechu přenášené informace nebo k jejímu podvržení. Aby k těmto situacím nemohlo dojít, je potřeba komunikaci zabezpečit.

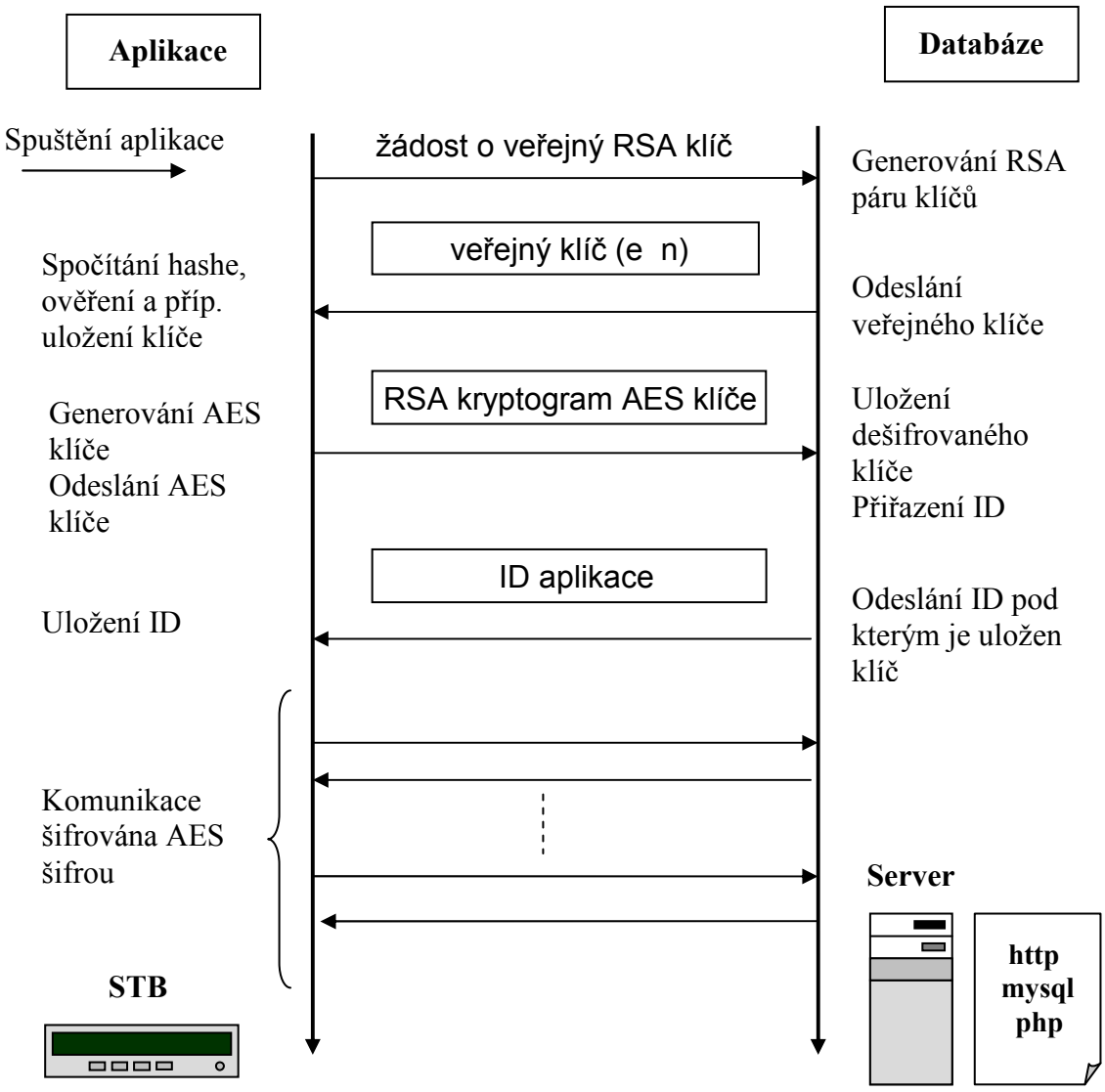
11.5.1 Zabezpečení AES šifrou

Kryptografie nabízí dva druhy metody šifrování, symetrické a asymetrické. Pro komunikace je využita kombinace obou druhů. Pro komunikaci mezi serverem a aplikací (v obou směrech) je využita symetrická šifra AES. Šifrování i dešifrování je prováděno tajným klíčem, který je na obou stranách stejný. Každá aplikace má svůj AES klíč, kterým komunikuje se serverem. Server samozřejmě musí identifikovat příslušnou aplikaci podle jejího čísla a použít AES klíč příslušné aplikace, který má uložený v databázi.

Každá aplikace si vygeneruje vlastní AES klíč, ten je ale potřeba bezpečným způsobem předat serveru. To vyžaduje bezpečný kanál nebo jiný způsob předání klíče (osobně, paměťové zařízení atd..). Kanál, kterým probíhá komunikace mezi serverem a aplikací, ale za bezpečný rozhodně považovat nelze, a proto je potřeba výměnu klíče nějakým způsobem zabezpečit.

11.5.2 Výměna AES klíče

Aby se zabránilo odposlechu posílaného AES klíče, je klíč zašifrován pomocí RSA algoritmu. RSA je asymetrická metoda, která využívá pár klíčů (viz kap.10.1.1), veřejný a soukromý klíč. Veřejný klíč, jak již název napovídá, není žádným tajemstvím, může být aplikaci zaslán nezabezpečeným kanálem. Tímto klíčem je zašifrován AES klíč a odeslán serveru, ten ho dešifruje pomocí svého privátního klíče. RSA pár klíčů je stále stejný, server má jeden soukromý klíč, který nezveřejňuje a jeden veřejný, který posílá každé aplikaci, která s ním chce šifrovaně komunikovat.



Obr. 11.5: Podrobné schéma výměny klíčů mezi serverem a aplikací

11.5.3 Ověření pravosti klíče

I když je veřejný klíč opravdu veřejný a tudíž ho není potřeba utajovat a nehrozí nebezpečí vyzrazení, stále hrozí nebezpečí podvržení klíče. Pokud útočník nacházející se mezi aplikací a serverem odchytí posílaný veřejný klíč, může ho vyměnit za svůj vlastní, který potom odešle aplikaci, za účelem dešifrování tajné komunikace ze strany aplikace. Dešifrovaná data potom útočník zašifruje skutečným veřejným klíčem serveru a odešle je serveru, který se tak o přítomnosti útočníka ani nedoví.

Zabránit takovému útoku lze přidáním kontrolní hodnoty ke klíči. Tato hodnota je vypočítána pomocí hashovací funkce, což je funkce, která pořídí kontrolní hodnotu, tzv. „otisk“ dat. Ten musí být pro stejná data vždy stejný. V aplikaci je pro výpočet hashovací funkce použit algoritmus SHA-1 (Secure Hash Algorithm), jeho hlavní vlastností je, že malá změna na vstupu způsobí velkou změnu na výstupu, tedy vytvoření úplně odlišného otisku.

V aplikaci je uložen otisk veřejného klíče serveru, který byl předán jinou cestou než nebezpečným kanálem. Po přijetí klíče aplikace pomocí SHA-1 algoritmu vypočte otisk klíče a porovná ho s uloženým. Pokud je jsou oba otisky shodné je tímto ověřeno, že se opravdu jedná o klíč serveru. Pokud ne, došlo k podvržení klíče nebo chybě při přenosu dat.

11.6 Organizace tříd v aplikaci

Jádrem celé aplikace je třída `MHPapplication`, která je napsána jako Xlet (viz kap. 6.3). Kromě standardních metod, které musí obsahovat každý Xlet (initXlet(), startXlet(), …) obsahuje metody pro vykreslení uživatelského rozhraní. Další funkce, které jsou nezbytné pro správný chod aplikace, a které spolu souvisí, jsou umístěny v samostatných třídách. Následuje seznam tříd a jejich stručný popis.

`MHPapplication` – jádro aplikace implementující rozhraní Xlet, zobrazení GUI

`KeyRequest` – stažení veřejného klíče (RSA) serveru

`KeyReadWrite` – čtení / ukládání RSA klíče z / do souboru

`AESKeyGenerator` – generování pseudonáhodného AES klíče pro komunikaci

`AESkeyChange` – odeslání AES klíče na server

`AEStables` – sestavuje 256B tabulky nezbytné pro šifrování a dešifrování AES

`Copy` – kopírování polí bytů (využito v AES implementaci)

`AESencrypt` – šifrování řetězců AES šifrou

`AESdecrypt` – dešifrování řetězců

`AESkeyRelease` – smazání AES klíče z databáze

`RSA` – implementace RSA algoritmu, šifrování, dešifrování

`Loging` – přihlášení uživatele k databázi

`Bet` – odeslání informací o uživatelově sázce do databáze

`Results` – stažení výsledku zápasu a sázky z databáze

`Hex` – převod hexadecimálního řetězce na pole bytů a naopak

11.6.1 Třída `MHPapplication`

Jak již bylo zmíněno výše, jedná se o hlavní třídu, implementující rozhraní `Xlet`. Touto třídou je tedy spouštěna celá aplikace. Kromě rozhraní `Xlet` implementuje třída několik dalších rozhraní. Rozhraní `Runnable`, umožňuje v aplikaci využít vláken, `HActionListener` umožňuje zachytávání událostí vyvolaných jednotlivými komponentami GUI (např. stisknutí tlačítka), `HKeyListener`umožňuje reagovat na stisknutí tlačítka ovladače.

Běh MHP aplikace začíná voláním metody `initXlet()`, ve které probíhá načtení obrázků a vytvoření objektu `HScene`. Následuje volání metody `startXlet()`. V této metodě je vytvořeno a spuštěno vlákno ve kterém aplikace běží.

Prvním krokem ve vlákně probíhající aplikace je získání veřejného klíče serveru, se kterým bude šifrovaně komunikovat. Spojení se serverem je zajištěno pomocí třídy `URLConnection`, která umožňuje vždy jen jeden dotaz na server a následně jednu odpověď. Získání klíče je provedeno voláním metody `keyRequest()` ze stejnojmenné třídy. Dalším krokem je porovnání hashe spočítaného ze staženého klíče a jeho porovnání s hashem uloženým v aplikaci využitím metod třídy `KeyReadWrite`. Pokud si hashe odpovídají, aplikace nabídne uložení klíče do souboru a při příštím spuštění už se na uložení klíče neptá, pokud se liší, aplikace se ukončí. V případě shody hashů, bez ohledu na uložení nebo neuložení klíče, následuje vygenerování AES klíče pro komunikace mezi aplikací a serverem, pomocí třídy `AESKeyGenerator` a jeho zašifrování pomocí RSA a odeslání na server, třída `AESkeyChange`.

Po provedení výměny klíčů je zobrazeno hlavní okno aplikace. Při zadávání uživatelského jména a hesla do komponent typu `HSingleLineEntry` jsou odchytávány jednotlivé znaky zapsané do komponenty a ty jsou ukládány do odpovídajících řetězců pro další zpracování. Řetězec s heslem se při psaní nezobrazuje a je nahrazen řetězcem hvězdiček. Stiskem tlačítka „Přihlášení“ se odešle login a heslo na server, tam je uživatel nalezen v databázi a v aplikaci se zobrazí informace o výši kreditu uživatele. To je zajištěno pomocí třídy `Loging`. Před odesláním je ještě zkontrolováno, zda uživatelské jméno a heslo splňuje definované požadavky na počet znaků a jestli obsahuje pouze povolené znaky (viz manuál k aplikaci).

Zadávání výše sázky a tipu na zápas má na starost třída `Bet`, princip je analogický jako u přihlašování uživatele. Třída, která se serverem komunikuje jako poslední je třída `Result`, která zajišťuje stažení výsledků zápasu z databáze a jejich zobrazení společně s vyhranou částkou.

11.6.2 Třída KeyRequest

Tato třída komunikuje se serverem pomocí `URLConnection` a posílá žádost o veřejný klíč serveru. Skript na serveru vygeneruje pár RSA klíčů a veřejný klíč pošle aplikaci.

11.6.3 Třída KeyReadWrite

Instanci této třídy jsou při vytváření předány dva parametry, veřejný klíč serveru a xlet kontext. Xlet kontext je nezbytný k nalezení cesty k paměti STB, do které je možné ukládat data. Třída obsahuje tři metody. Metoda `writeKey()` provádí zápis veřejného klíče do souboru, metoda `readKey()` ho ze souboru načítá a metoda `compareKey()` porovná klíč uložený v souboru s klíčem předaným objektu této třídy při inicializaci.

11.6.4 Třída AESKeyGenerator

Třída využívá generátor pseudonáhodných čísel `SecureRandom`, který je založen na algoritmu SHA1. Tato třída poskytuje kryptograficky silná čísla. Samotné generování AES klíče provádí metoda `generateKey()`. Generuje náhodná čísla v rozsahu 0 – 127. Aby nedocházelo k nesprávné interpretaci klíče v různých znakových sadách, je klíč převeden do šestnáctkové soustavy. Délka klíče je 128 bitů, vyjádřena parametrem *keylength*, kterým je inicializován objekt této třídy.

11.6.5 Třída AESKeyChange

Stejně jako všechny ostatní třídy komunikující se severem využívá i tato třída pro komunikaci `URLConnection`. Jako parametry do třídy vstupuje vygenerovaný AES klíč a veřejný RSA klíč serveru. Metoda `sendKey()` zašifruje AES klíč veřejným klíčem serveru a odešle jej. Server přijatý AES klíč dekóduje svým soukromým klíčem a uloží jej do databáze. Aby server odlišil klíče od různých aplikací, přidělí každé aplikaci číslo, aplikace si jej uloží do parametru *appID*.

11.6.6 Třídy AEStables, AESencrypt, AESdecrypt, Copy

Tyto čtyři třídy slouži k šifrování a dešifrování řetězců symetrickou kryptografickou metodou AES. Jelikož platforma MHP 1.1.2 neobsahuje knihovnu pro kryptografii, bylo nutné nalézt implementaci AES šifry, která bude odpovídat standardu a bude kompatibilní s kryptografickými metodami v php. Vytvoření implementace AES šifry je velmi komplexní úkol a vyžaduje hlubší studii, přesahující rámec této práce. V aplikaci je tedy s laskavým svolením autora použita implementace AES šifry od Neala R. Wagnera. profesora Univerzity Texasu a San Antonia. AES šifrování využívá 128 bitový klíč a pracuje v módu EBC. Z pohledu vytvářené aplikace, ale na módu ve kterém AES šifrování probíhá příliš nezáleží (viz kapitola 10.2.1.2).

11.6.7 Třída `RSA`

Do této třídy je jako parametr předán veřejný klíč serveru. Tím jsou zašifrována data v podobě řetězce, která jsou předávány jako parametr funkci `encrypt()`, která provádí vlastní šifrování. Šifrování probíhá podle vzorce (1).

11.6.8 Třídy Loging, Bet, Results

Všechny tyto tři třídy fungují analogicky. Využívají třídy `URLConnection` pro odeslání požadavku na serveru a stažení odpovědi. Třída `Loging`, slouží k přihlášení uživatele, přenáší se login a heslo na server, pokud je uživatel ověřen, vrací se výše kreditu

uživatele. Třída `Loging` umožňuje tři pokusy na přihlášení uživatele, pokud ani jeden není úspěšný, aplikace se ukončí. Třída `Bet` slouží k uskutečnění sázky, na server se odesílá výše sázky, která samozřejmě nesmí přesáhnout aktuální výši kreditu a tip na zápas. Třída `Results` odesílá žádost o výsledky zápasu. Server jako odpověď posílá výsledek zápasu a výsledek uživatelovy sázky.

Veškerá komunikace v rámci těchto tří tříd je šifrována AES šifrou. Kromě uživatelských dat je na server současně posílán identifikátor aplikace *appID*, ve kterém je uloženo číslo aplikace přidělené serverem. Díky tomuto číslu je server schopen načíst z databáze odpovídající AES klíč pro dekódování přijatých dat.

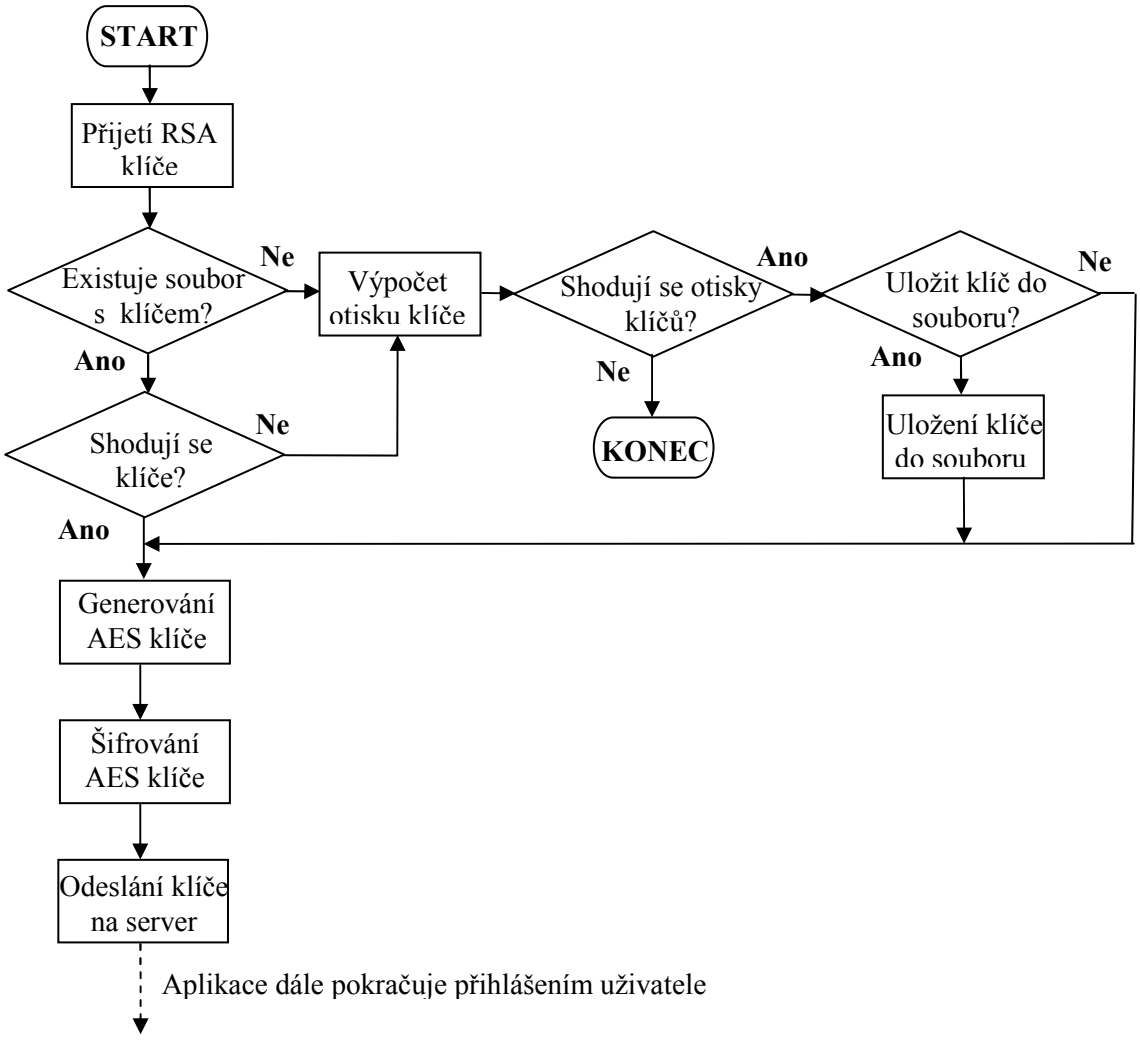
11.6.9 Třída AESkeyRelease

Stejně jako předešlé třídy komunikující se serverem, využívá i tato třída ke komunikaci třídu URLConnection. Třída slouží pro smazání AES klíče příslušejícího dané aplikaci z databáze při jejím ukončení.

11.6.10 Třída HEX

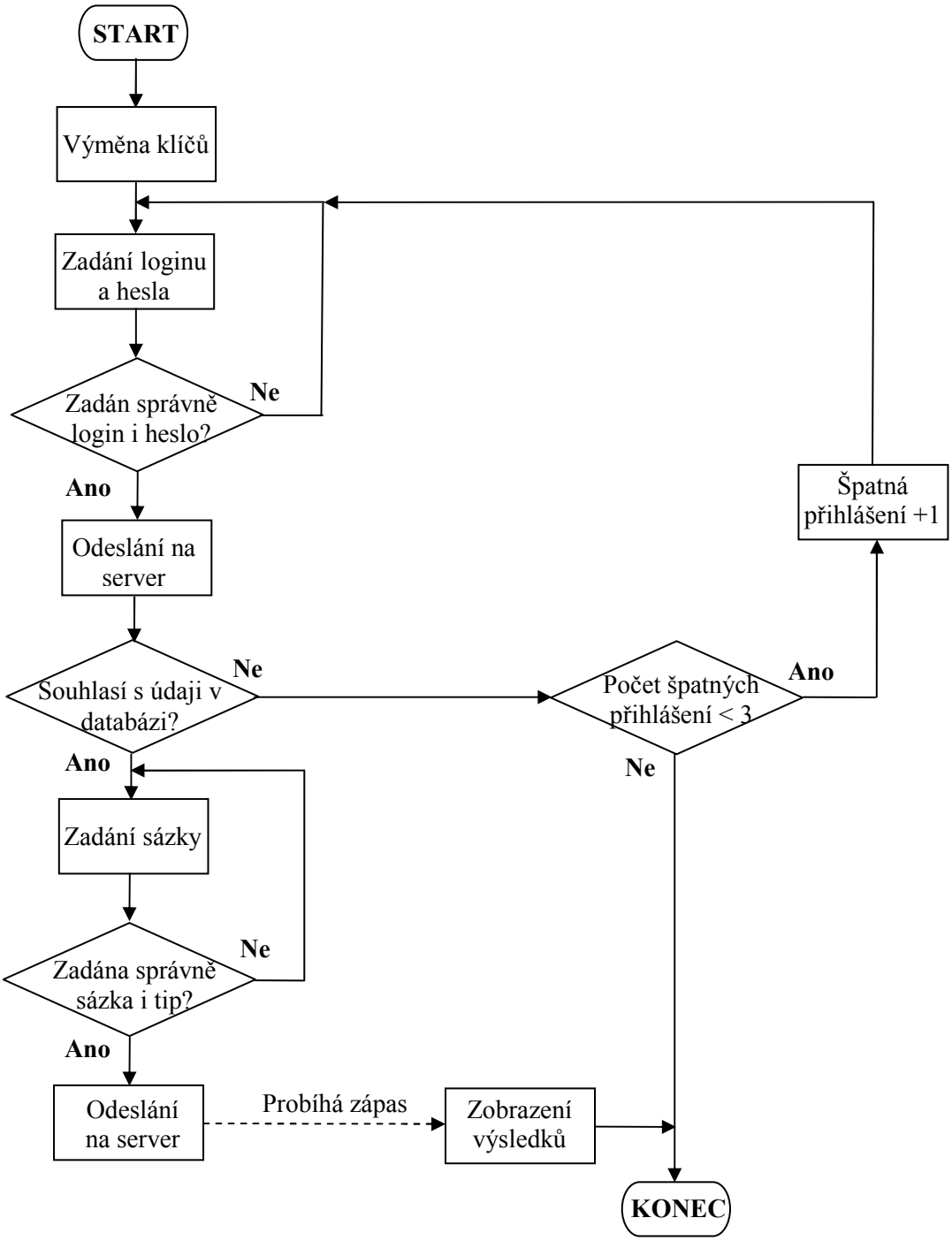
Tato třída slouží k převodu pole bytů na do hexa znaků a naopak.

11.7 Vývojový diagram popisující funkci aplikace



Obr. 11.6: Vývojový diagram výměny klíčů

Funkce aplikace je již popsána v předcházejících kapitolách, ale pro názornost je ještě uveden vývojový diagram výměny klíčů mezi aplikací a serverem (obr. 11.6) a vývojový diagram znázorňující zjednodušeně celou funkci aplikace pro online sázení (obr. 11.7).



Obr. 11.7: Zjednodušený vývojový diagram znázorňující funkci aplikace

11.8 Uživatelské rozhraní aplikace

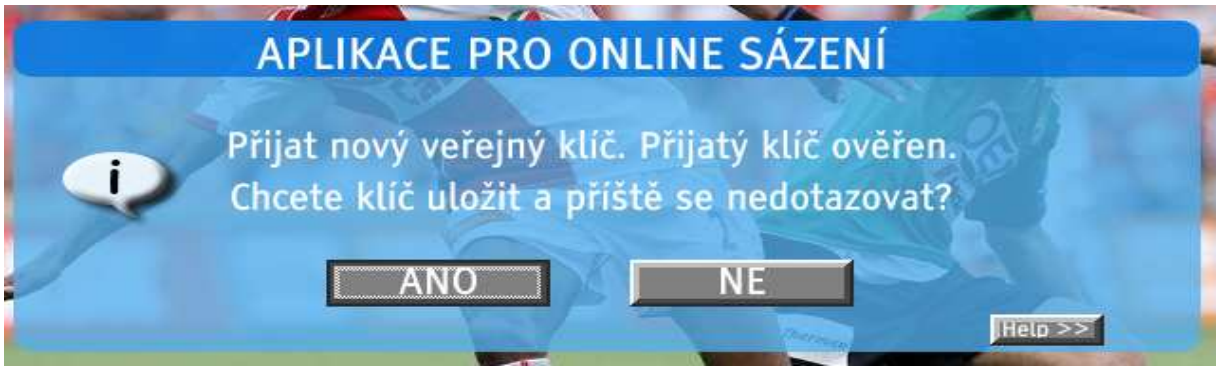
Uživatelské rozhraní aplikace je vytvořeno s ohledem na to, že aplikace bude fungovat na STB a bude ovládána pouze dálkovým ovladačem. Takové ovládání není příliš rychlé a pohodlné při navigaci komplexním uživatelským rozhraním, proto je uživatelské rozhraní tomuto faktu přizpůsobeno a je vytvořeno s důrazem na snadné a rychlé ovládání aplikace.

V souvislosti s uživatelským rozhraním je v textu používán pojem okno aplikace, nejedná se ale o klasické okno ve smyslu Windows nebo klasických javovských aplikací, STB totiž nemá žádný manažer oken.

11.8.1 Okno s dotazem na uložení klíče

Při prvním spuštění aplikace se zobrazí dotaz na uložení klíče. Okno s dotazem je na obr. 11.8. Toto okno se zobrazí pouze v případě, že byla ověřena pravost přijatého klíče. Stisknutím tlačítka ANO dojde k uložení klíče do paměti STB a při příštím spuštění je už hlavní okno aplikace spuštěno přímo bez zobrazení dotazu na uložení klíče. Kontrola pravosti klíče je pak prováděna pouze porovnáním uloženého a přijatého klíče. Pokud je přijat nový klíč, je zobrazeno toto okno znovu. Stisknutím tlačítka NE se klíč neuloží a při příštím spuštění je opět kontrolován hash klíče a je zobrazen znovu dotaz na uložení klíče. Tlačítko Help slouží k zobrazení stručného popisu činnosti po stisku jednotlivých tlačítek.

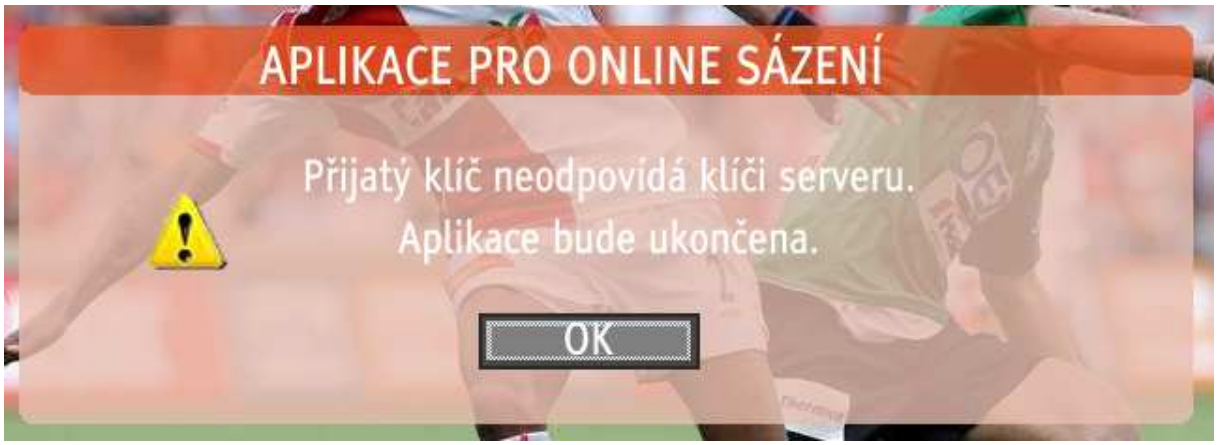
Bez ohledu na uložení nebo neuložení klíče do paměti pokračuje dále program zobrazením hlavního okna aplikace.



Obr. 11.8: Dotaz na uložení klíče

Obr. 11.9: Dotaz na uložení klíče s nápovědou

Pokud hash spočítaný z přijatého klíče nesouhlasí s hashem uloženým v aplikaci, znamená to, že klíč byl podvržen nebo poškozen při přenosu. Aplikace zobrazí chybové okno, které je na obr. 11.10. Po stisku tlačítka OK je aplikace ukončena.



Obr. 11.10: Chybové okno

11.8.2 Přihlášení do aplikace

Přihlašovací okno aplikace je na obr. 11.11. Uživatelské jméno a heslo, které se zadává do polí Login a Heslo, musí být dlouhé minimálně 8 znaků, maximálně však 16 znaků a smí obsahovat pouze malá a velká písmena a čísla. Stiskem tlačítka Přihlášení dojde k přihlášení uživatele k databázi a zobrazení aktuální výše kreditu v poli Kredit.

Při zadání neplatného uživatelského jména nebo hesla umožní aplikace další dva pokusy o přihlášení, potom se ukončí. Na zadání neplatného uživatelského jména a hesla reaguje aplikace oknem odpovídajícím obr. 11.12. Stiskem tlačítka OK je uživateli umožněno zadat znovu své přihlašovací údaje.

Obr. 11.11: Přihlašovací okno

V případě zadání uživatelského jména nebo hesla nesplňujícího některý z požadavků (kratší než 8 znaků, neplatné znaky) k přihlášení k databázi uživatelů vůbec nedojde. Aplikace zobrazí stejné varovné okno jako při neplatných přihlašovacích údajích, s tím rozdílem, že varovný text informuje o zadání neplatného znaku. Po stisku tlačítka OK může uživatel opakovaně vyplnit uživatelské informace. Teprve po zadání platných údajů umožní přihlášení k databázi.

Pokud jsou vyčerpány všechny tři pokusy o přihlášení, aplikace zobrazí stejné varovné okno jako u předešlých případů, opět je rozdíl pouze v obsahu varovného sdělení. Po stisku tlačítka OK je aplikace ukončena.

Obr. 11.12: Reakce aplikace na zadání neplatných uživatelských údajů

11.8.3 Hlavní okno aplikace

Po úspěšném přihlášení lze zadat výši sázky do stejnojmenného pole a tip na výsledek zápasu. Částka musí být menší nebo rovna výši kreditu uživatele a musí být celočíselná. Minimální výše sázky je 20 Kč. Tip na výsledek zápasu může nabývat pouze hodnot 1, 0, 2, 10 a 02, představujících po řadě výhru domácích, remízu, výhru hostů, „neprohru“ domácích a „neprohru“ hostů. „Neprohra“ znamená, že domácí tým, respektive tým hostů vyhraje nebo remizuje.

Obr. 11.13: Hlavní okno aplikace pro online sázení

Po zadání sázky se aplikace skryje, a po skončení zápasu se znovu zobrazí s výsledkem sázky a aktuální výší kreditu. Stiskem tlačítka *Konec* se aplikace ukončí.

11.9 Testovací STB

Aplikace byla testována na STB STRONG SRT 5510 MHP. Jedná se o MHP přijímač s ethernet portem pro multimediální aplikace.

Technické parametry

- Demodulace COFDM/16 QAM, 64 QAM
- Přenosové pásmo: 2K, 8K
- Kmitočtový rozsah: 177,5-466,0 Mhz; 474,0-858,0 Mhz
- Šířka pásma: 8/7 Mhz
- Videodekodér MPEG-2MP@ML
- Rychlost přenosu dat: až 15Mbits/s
- Videoformát: 4:3, 16:9
- 32 Bit RISC, MHP 1.1.2
- Flash memory: 48 MB
- SDRAM: 64MB

12 Závěr

Projekt aplikace pro online sázení na platformě MHP v prostředí DVB-T tvoří dvě části. První z nich je samotná aplikace napsaná v jazyce Java, která běží na Set-top boxu, částí druhou je serverová strana zahrnující databáze s informacemi o uživatelích a o probíhajícím zápase a skript, který k databázi přistupuje, vyhledává v ní potřebné informace a komunikuje s aplikací.

Pro vytvoření aplikace bylo využito vývojové prostředí NetBeans verze 5.5.1 s příslušnými knihovnami pro JavaTV, funkčnost aplikace byla testována softwarem IRT MHP RI 1.0.2 – 3final a na reálném STB STRONG SRT 5510.

Skript je napsán v jazyce PHP a pracuje s databází mySQL. PHP samozřejmě umí pracovat i s jinými databázemi, např. Oracle, Postgree nebo MS Access. Výhodou mySQL je, že tento typ databáze podporuje mnoho freewebů, čehož bylo využito při vývoji aplikace. Alternativou k PHP je například skriptovací technologie ASP nebo technologie k tvorbě webových aplikací ASP.NET. Každá z těchto technologií má svá pro a proti, PHP ale vítězí svou jednoduchostí a dostupností.

Komunikace mezi aplikací a serverem je uskutečněna pomocí knihovny `URLConnection` z balíku síťových služeb `java.net`. Jedná se službu zpřístupňující obsah síťového zdroje na základě jeho identifikace pomocí URL.

Na straně aplikace je vytvořeno uživatelské rozhraní, které umožňuje zadání uživatelského jména a hesla, jejich následné odeslání na server a zobrazení výše kreditu, který je možno vsadit. Na straně serveru je vytvořena mySQL databáze obsahující informace potřebné k identifikaci příslušného uživatele a informace o výši jeho kreditu a PHP skript pracující s databází. Po přijetí uživatelského jména a hesla, odeslaného aplikací, vyhledá skript příslušného uživatele v databázi a odešle aplikaci informaci o výši jeho kreditu.

Po úspěšném přihlášení je uživateli umožněno vsadit na nabízený zápas. Po uskutečnění sázky uživatel skryje aplikaci a může sledovat průběh utkání na které si vsadil. Po jeho skončení si spustí opět aplikaci, která se připojí k serveru a stáhne výsledky zápasu i sázky a ty zobrazí. Pro komerční použití by bylo možné aplikaci upravit tak, aby se před začátkem přenosu a po jeho skončení spouštěla automaticky speciálním signálem z televizního vysílání.

Celá komunikace mezi aplikací a serverem je šifrována symetrickou šifrou AES. Šifrování komunikace symetrickou šifrou je oproti použití asymetrické šifry řádově rychlejší a tudíž i vhodnější pro zpracování set-top boxem, který má omezené výpočetní možnosti. Klíč kterým budou data symetricky šifrována je generován aplikací a na server je posílán zašifrovaný asymetrickou šifrou RSA. Veřejný klíč potřebný pro toto asymetrické šifrování je zaslán aplikaci serverem a jeho autentičnost je ověřena spočtením „otisku“ klíče pomocí SHA-1 algoritmu a porovnáním s uloženou hodnotou.

Seznam použité literatury

[1] SCHWALB, E. M. ITV Handbook: Technologies and Standarts, London: Prentice Hall,2004. ISBN 978-0131003125

[2] STEVEN, M. Interactive TV standards: A Guide to MHP, OCAP, and JavaTV. Elsevier: Focal Press. 2005, ISBN 978-0240806662

[3] O´LEARY, Seamus. Understanding Digital Terrestrial Broadcasting . 1st edition. Norwood : Artech House, INC., 2000. 259 s. ISBN 1-58053-462-7.

[4] HEROUT, Pavel. Učebnice jazyka Java. 2. vyd. České Budějovice : KOPP, 2006. 344 s. ISBN 80-7232-115-3.

[5] MORRIS, Steven. Interactive TV Web [online]. 2002 , Last updated: 2005-09-14 [cit. 2007-12-08]. Dostupný z WWW: <http://www.interactivetvweb.org>.

[6] The PHP Group. PHP: Hypertext Preprocessor [online]. 2001 , Last updated: Dec 17 2007 [cit. 2007-12-10]. Dostupný z WWW: <http://www.php.net>.

[7] GRIMMICH, Šimon, et al. Tvorba webu [online]. 2003 [cit. 2007-12-08]. Dostupný z WWW: <http://www.tvorba-webu.cz>.

[8] Sun Microsystems, INC. Java Technology [online]. 1994 [cit. 2007-12-12]. Dostupný z WWW: <http://java.sun.com>.

[9] WAGNER, Neal R.. The Laws of Cryptography [online]. 2001 [cit. 2008-04-21]. Dostupný z WWW: <http://www.cs.utsa.edu/~wagner/laws/>.

[10] Wikipedia [online]. last modified on 25 May 2008 [cit. 2008-04-15]. Dostupný z WWW: <www.wikipedia.org>.