VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

MANAŽER HESEL S ARCHITEKTUROU KLIENT-SERVER
CLIENT-SERVER PASSWORD MANAGER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Vojtěch Myška**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Petr Ležák**
SUPERVISOR

BRNO 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Teleinformatika**
Ústav telekomunikací

Student: Vojtěch Myška ***ID:*** 164611
Ročník: 3 ***Akademický rok:*** 2015/16

NÁZEV TÉMATU:

Manažer hesel s architekturou klient-server

POKYNY PRO VYPRACOVÁNÍ:

Implementujte manažer hesel s architekturou typu klient-server. Manažer se bude skládat ze dvou částí - serveru, který bude data ukládat, a klienta, který bude provádět šifrování dat a bude mít vhodné uživatelské rozhraní. Ke stejnému serveru se musí jít přihlásit z více různých klientů. Server musí klienta vhodným způsobem autentizovat a komunikace mezi nimi musí být šifrovaná.

DOPORUČENÁ LITERATURA:

[1] BURDA, Karel. Bezpečnost informačních systémů. Brno: FEKT Vysokého učení technického v Brně. 1.11.2005. 104s

[2] Oracle. Java Platform, Standard Edition 7: API Specification [online]. 2011 Dostupné z: http://docs.oracle.com/javase/7/docs/api/index.html

Termín zadání: 1.2.2016 ***Termín odevzdání:*** 1.6.2016

Vedoucí práce: Ing. Petr Ležák
Konzultant bakalářské práce:

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

ABSTRAKT

Bakalářská práce se zabývá implementací manažeru hesel s architekturou klient-server. Základem práce je návrh uživatelského rozhraní klientské části, používaných klíčů, vzdálených metod, serveru a jeho databáze, výběr použitých technologií a programovacího jazyku. Následuje část pojednávající o implementaci návrhu manažeru hesel.

KLÍČOVÁ SLOVA

Manažer hesel, architektura klient-server, bezpečnost hesel, Java, JavaFX, JavaRMI, Hibernate, ORM, MySQL, AES, šifrování

ABSTRACT

This bachelor thesis deals with implementation password manager with client-server architecture. Basis of thesis is proposal of graphical user interface, used keys, remote methods, server and its database, choise of used technology and programming language. The next part delas with implementation proposal of password manager.

KEYWORDS

Password manager, client-server architecture, password security, Java, JavaFX, JavaRMI, Hibernate, ORM, MySQL, AES, encryption

MYŠKA, Vojtěch *Mananžer hesel s architekturou klient-server*: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2016. 61 s. Vedoucí práce byl Ing. Petr Ležák

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Mananžer hesel s architekturou klient-server“ jsem vypracoval(a) samostatně pod vedením vedoucího bakalářské 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(ka) uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

Vojtěch Myška

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu semestrálního projektu panu Ing. Petru Ležákovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

Vojtěch Myška

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM VÝPISU KÓDU

1 ÚVOD

V současné moderní době začíná přibývat stále více elektronicky dostupných služeb, nejen díky dostupnosti internetového připojení a velkému rozmachu počítačů. Jde například o internetové bankovnictví, elektronickou poštu, sociální sítě, elektronické peněženky, internetové obchody, diskuzní fóra, hesla k domácí bezdrátové síti a podobně. Přístup je k těmto službám obvykle chráněn kombinací uživatelského jména a hesla. Tím nastává problém stále se zvětšujícího množství přihlašovacích údajů v závislosti na počtu využívaných služeb. Uživatelé si proto volí snadná a krátká hesla (viz tab. 1.1 10 nebezpečných hesel, citováno z [18]), která používají pro více služeb, čímž se ale vystavují nebezpečí zneužití svých účtů. Na toto chování uživatele často spoléhají i útočníci a při prolamování hesel některých uživatelů se ani „nezapotí“. Výrazného ztížení nebo dokonce i znemožnění provedení úspěšného útoku se docílí používáním dlouhých a silných hesel, která jsou směsicí písmen a číslic, případně znaků. Útočník tak prakticky ztrácí možnost napadnutí hesla slovníkovým útokem. Tato ochrana má však negativní vliv i na uživatele, protože je prakticky nemožné zapamatovat si desítky zcela odlišných a silných hesel. Problém lze vyřešit zaznamenáním hesel do papírového bločku, případně do poznámkového bloku v počítači. Druhá možnost je přímo nebezpečná, neboť si kdokoliv může přihlašovací údaje přečíst, nebo dokonce zkopírovat. Nejlepší alternativou, při absenci „výborné“ paměti, je použít pro ukládání přihlašovacích údajů programy pracující s hesly, tzv. sejfy hesel. Fungují na principu jednoho silného hlavního hesla sloužícího jako přístupový údaj do šifrované databáze. Jedním z neplacených a dobrých sejfů hesel je například program KeePass. Ten pracuje s databází, která je šifrována pomocí AES algoritmu. Pro přístup do databáze používá porovnání otisku hlavního hesla s právě zadávaným. Otisky jsou vytvářeny SHA-256 hešovací funkcí. Jako přístupové heslo umožňuje KeePass použít řetězec znaků, soubor nebo jejich kombinaci, která zvyšuje odolnost proti útoku. Nevýhody KeePassu vyplynou například při odcizení, či zničení počítače. I přes svoji šifrovanou podobu je databáze volně přístupná útočníkovi a pro uživatele navždy ztracena i s uloženými záznamy. Zmíněné nevýhody má odstranit manažer hesel s architekturou klient-server, jehož návrh a realizace je součástí této bakalářské práce.

Pořadí	Heslo	Pořadí	Heslo
1	123456	6	123456789
2	Password	7	123
3	12345	8	baseball
4	12345678	9	dragon
5	querty	10	football

Tab. 1.1: Nebezpečná hesla

V teoretickém rozboru jsou zmíněny základy symetrických a asymetrických kryptografických systémů. U uvedených kryptosystémů je popsán jejich princip fungování, použití a základní dělení. Dále jsou zmíněny jejich klíčové vlastnosti, včetně využívaných šifrovacích algoritmů. V této části práce se také vyskytuje popis hešovacích funkcí, kde se pojednává o vlastnostech a o jejich základních třech požadavcích.

Součástí následující kapitoly je podrobný popis technologií, jenž jsou využívané realizovaným manažerem hesel s architekturou klient-server. V případě klientské části programu jde především o zvolený šifrovací algoritmus AES, jeho provozní režim, hešovací funkci a autentizační protokol. Další zmínka je o vlastnostech databáze a vybraném databázovém systému, neboť je důležitou součástí serverové části manažeru. Nedílnou součástí kapitoly je popis zvoleného programovacího jazyku Java.

Následuje předposlední kapitola, která byla vypracována v rámci řešení semestrální práce. Je věnována návrhu manažeru hesel skládajícího se ze dvou částí – klientské a serverové. Součástí kapitoly jsou požadované vlastnosti manažeru hesel a navržené RMI rozhraní. Je proveden také návrh grafického rozhraní klienta, záznamu hesla a databáze.

Poslední kapitola popisuje vytvořený projekt manažeru hesel s architekturou klient-server. Je členěna do sekcí pojmenovaných podle popisovaných modulů manažeru hesel. Každá sekce popisuje jednotlivé moduly. Součástí některých sekcí je také zkrácený výpis důležitých částí zdrojového kódu.

2 TEORETICKÝ ROZBOR

Téma bakalářské není přímo zaměřeno na kryptografii. Popsané kryptosystémy a hešovací funkce jsou však její důležitou součástí. Kapitola obsahuje základní dělení a princip činnosti jednotlivých kryptosystémů.

2.1 Kryptografické systémy

Kryptografické systémy dělíme na dva základní typy. Na symetrické a asymetrické kryptosystémy. Prvně zmíněný kryptosystém je charakteristický používáním stejného klíče pro šifrování i dešifrování. V případě asymetrického kryptosystému je princip jiný. Od symetrického kryptosystému se liší tím, že používá dvojici klíčů. Veřejný klíč, jenž je všem dostupný a soukromý klíč, který je držen v tajnosti. Prozatím je díky časové složitosti tajný klíč na základě znalosti klíče veřejného neodvoditelný.

2.1.1 Symetrický kryptosystém

Nyní přejdeme k popisu modelu symetrického kryptosystému zobrazeného na obr. 2.1. Pokud chce odesílatel doručit příjemci zprávu M, která je určena pouze pro něj, musí být zpráva M utajena. Toho se docílí jejím zašifrováním v bloku *Šifrování* pomocí funkce $\mathrm{E}(M, K_E)$, kde K_E je šifrovací klíč odesílatele a M je ona vstupní zpráva. Výstupem popsaného procesu je šifrovaná zpráva C, tzv. kryptogram. Kryptogram je následně přenášen po přenosovém kanálu k příjemci. Přenosový kanál je volně přístupný i útočníkovi, který má možnost získat přenášený kryptogram. Díky procesu šifrování však nemá možnost z kryptogramu C bez znalosti klíče K_D získat původní zprávu M.

Po doručení kryptogramu C příjemci se provede jeho dešifrování pomocí funkce $\mathrm{D}(C, K_D)$, kde K_D je dešifrovací klíč. Výsledkem je původní zpráva M.

Z prostoru klíčů je získán tajný klíč K. Používá se jako šifrovací klíč K_E na straně odesílatele a zároveň jako dešifrovací klíč K_D, který je doručen příjemci přes bezpečný přenosový kanál.

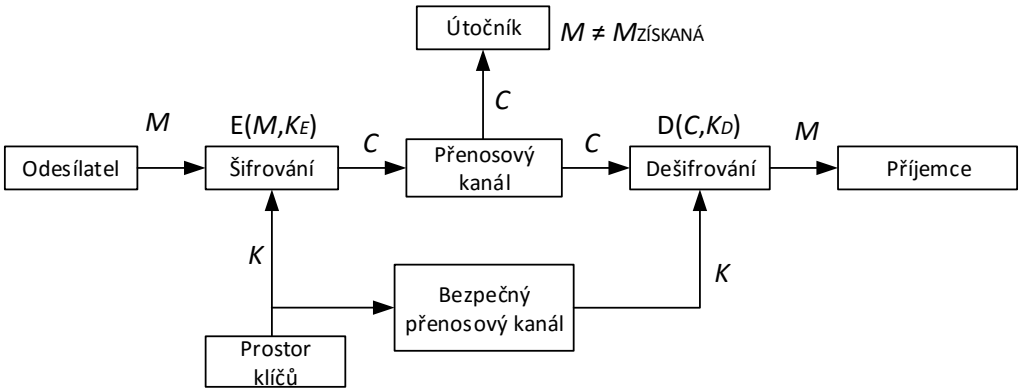
Velkou nevýhodou těchto kryptosystémů je distribuce tajných klíčů a jejich potřebný počet, který lze vypočíst z rovnice 2.1, kde n je počet uživatelů.

$$\frac{n * (n-1)}{2} \tag{2.1}$$

Symetrické kryptosystémy dělíme na dvě skupiny:

- proudové šifry,
- blokové šifry.

Teoretické poznámky byly čerpány z [17],[11].



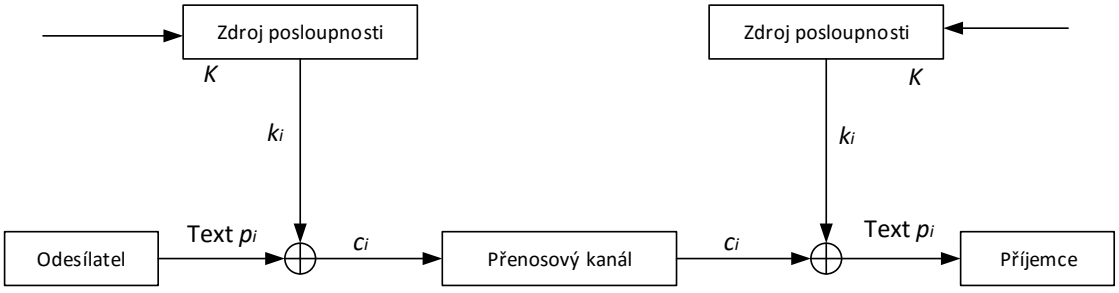
Obr. 2.1: Model symetrického kryptosystému

2.1.2 Proudové šifry

Jsou charakteristické tím, že každý bit zprávy je šifrován zvlášť. Hodnota zašifrovaného bitu je tedy závislá na hodnotě bitu klíče a na hodnotě právě šifrovaného bitu. Proudové šifry se dělí podle použitého zdroje posloupnosti bitů na náhodné a pseudonáhodné.

Princip proudové šifry: V závislosti na klíči K je ze zdroje posloupnosti získána posloupnost bitů. K šifrování dochází při sčítání modulo2 bitu posloupnosti k na pozici i s bitem textu p na pozici i. Výstupem je zašifrovaný bit c na pozici i, který je pomocí přenosového kanálu doručen k příjemci.

Podmínkou pro správný proces dešifrování je, že příjemce musí mít k dispozici stejný klíč K i zdroj posloupnosti jako odesílatel. Proces dešifrování se liší od procesu šifrování tím, že bit c na pozici i slouží jako vstup. Výstupem je dešifrovaný bit textu p na pozici i. Podrobnější popis lze najít například v [2].



Obr. 2.2: Proudová šifra

Výhody: Při vzniku chyby během přenosu se chyba projeví pouze v jediném symbolu zprávy. Mezi další výhody patří vysoká šifrovací rychlost a méně náročná implementace oproti blokovým šifrám.

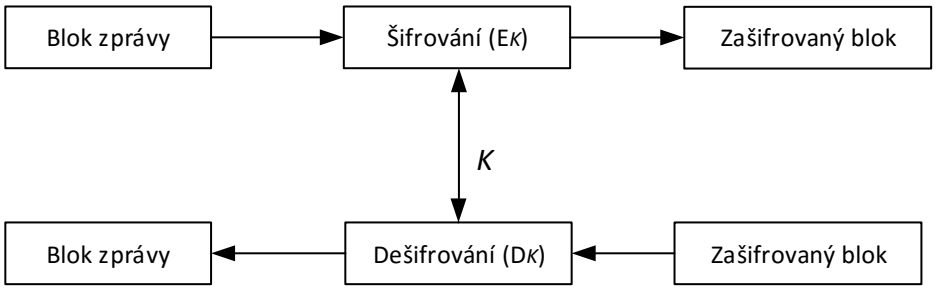
Nevýhody: Hlavní nevýhodou je podstatně menší bezpečnost, než kterou poskytuje bloková šifra.

Příklady proudových šifer: A5, RC4, FISH, Helix, SEAK, WAKE.

Blokové šifry:

Bloková šifra pracuje na zcela odlišném principu, než výše zmiňovaná proudová šifra. Zprávu nejprve rozdělí na bloky o pevné délce (64, 128 nebo 256 bitů). V případě, že poslední blok zprávy nemá potřebnou délku, je provedeno jeho zarovnání, tzv. padding. Bloky jsou následně šifrovány pomocí klíče K. Výsledkem je zašifrovaný blok mající shodnou velikost jako vstupní blok zprávy. Dešifrování probíhá opačným směrem, jako klíč je taktéž použit klíč K, viz 2.3.

Slabinou blokových šifer je opakovaná aplikace jednoho stejného klíče na všechny bloky. Pro odstranění popsané slabiny se používají režimy blokových šifer, například ECV, CBC, CFB, OFB. Popis CBC režimu je součástí sekce 3.3 - *Provozní režim blokové šifry - CBC*. Podrobné popisy ostatních režimů blokových šifer lze najít například v [2].



Obr. 2.3: Bloková šifra

Výhody: Poskytují vyšší bezpečnost, než proudové šifry

Nevýhody: Složitější implementace a pomalejší šifrovací rychlost, chyba se neprojeví pouze v jednom bitu

Příklady blokových šifer: DES, 3DES, AES, Twofish.

2.1.3 Asymetrický kryptosystém

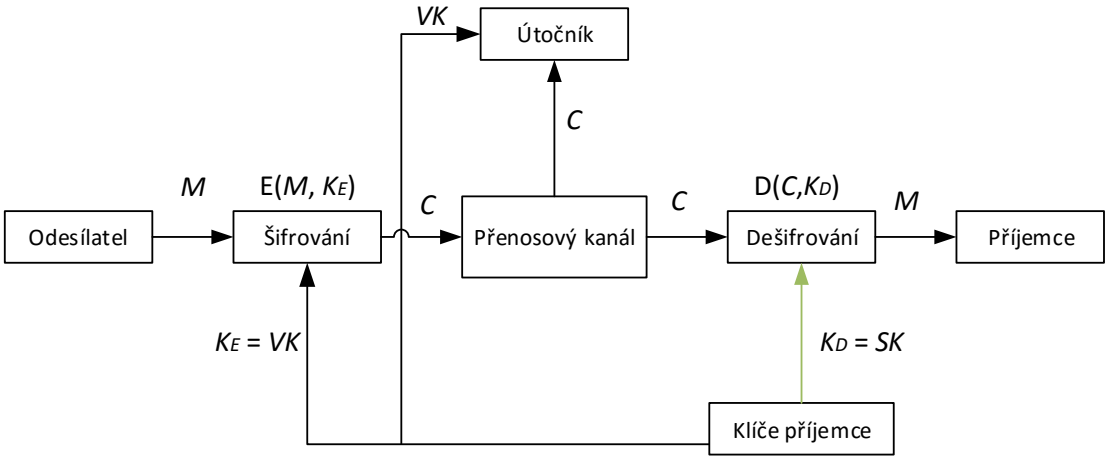
Asymetrická kryptografie na rozdíl od symetrické používá dvojici klíčů – veřejný a soukromý. Soukromý klíč nelze z veřejného klíče díky časové složitosti odvodit. Asymetrickou kryptografii lze využít pro tyto dvě funkce:

- šifrování dat,
- ověření autentičnosti.

Teoretické poznatky využité pro psaní této podkapitoly jsou čerpány z [2],[17],[11].

Princip asymetrického kryptosystém pro šifrování dat

Šifrování dat pomocí asymetrického kryptosystému je zobrazeno na obr. 2.4. Odesílatel posílá příjemci zprávu M, jenž je zašifrována v bloku s názvem *Šifrování* pomocí funkce $\mathrm{E}(M, K_E)$. Šifrovací klíč K_E je veřejný klíč příjemce. Vzniklý kryptogram C je přenesen přes přenosový kanál k příjemci. Může se stát, že útočník získá mimo kryptogramu C i veřejný klíč příjemce. Naštěstí bez znalosti soukromého klíče, který je držen v tajnosti příjemcem, není útočník schopen odchycený kryptogram C dešifrovat. Kryptogram C je po přijetí příjemcem dešifrován v bloku s názvem *Dešifrování* pomocí funkce $\mathrm{D}(C, K_D)$, kde dešifrovací klíč K_D je soukromý klíč příjemce. Výstupem je původní zpráva M.

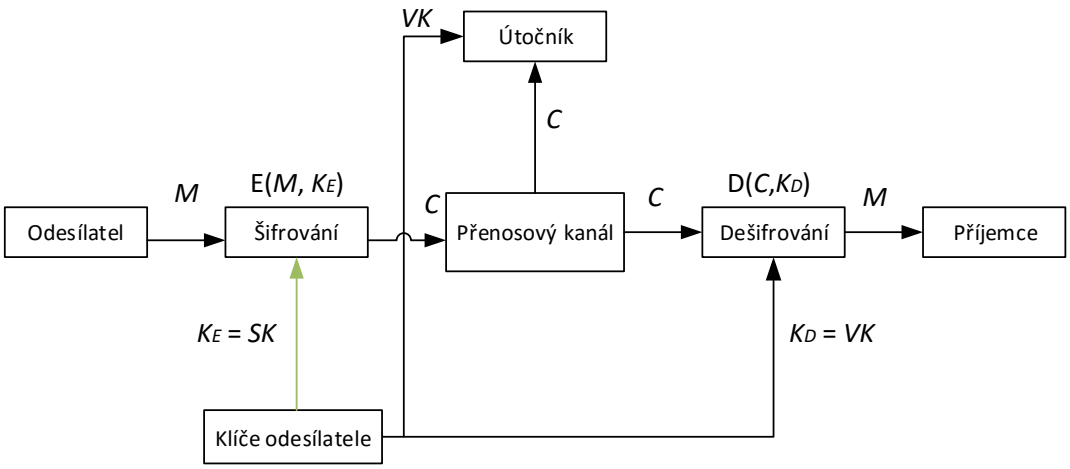


Obr. 2.4: Asymetrický kryptosystém pro šifrování dat

Asymetrický systému umožňující ověření integrity a autentičnosti

Posledním popisovaným kryptografickým modelem je model sloužící pro ověření integrity a autentičnosti, viz obr. 2.5. Princip je obdobný jako u předchozího modelu.

Odesílatel také provede zašifrování zprávy v bloku se jménem *Šifrování* pomocí funkce $\mathrm{E}(M, K_E)$. Rozdíl panuje v použití šifrovacího klíče. Zatímco v předchozím případě byl použit veřejný klíč příjemce, tak zde se využije soukromý klíč odesílatele. Vzniklý kryptogram C je poslán přenosovým kanálem příjemci. K přenosovému kanálu má přístup také útočník, který může získat kromě kryptogramu i veřejný klíč odesílatele. Díky tomu může zprávu dešifrovat, ale není schopen bez znalosti soukromého klíče odesílatele podstrčit příjemci modifikovanou zprávu. Z toho vyplývá takový závěr, že model nezaručuje důvěrnost přenášených dat, nýbrž zajišťuje integritu a autentičnost přenášené zprávy.



Obr. 2.5: Asynchonní kryptosystém pro ověření integrity a autentičnosti

2.2 Hešovací funkce

Využívají se ve více oborech, nicméně v této bakalářské práci budeme popisovat použití pouze v rámci kryptografie. Jejich hlavním úkolem je vytvářet identifikátor, tzv. heš (též nazývaný jako otisk, hešovací kód či miniatura) ze vstupního argumentu, který je tvořen řetězcem bitů libovolné délky n. Výsledný heš má pevnou délku m a je stejně jako vstupní argument řetězcem bitů. Z této vlastnosti vyplývá, že počet vstupních možností je prakticky nekonečný. Výstupních možností je konečný počet a závisí na bitové délce hešovací funkce. Funkce se často využívá k vytváření otisků hesel nebo pro zajištění integrity zprávy, neboť při změně jejích bitů dojde ke změně heše. Aby měla hešovací funkce smysl, musí splňovat tyto tři požadavky:

- jednosměrnost,
- odolnost proti modifikaci vstupu,
- odolnost proti kolizi.

Jednosměrnost

Znamená nevypočitatelnost vstupní hodnoty x ze znalosti výstupní hodnoty y, viz rovnice 2.2.

$$h(x) = y \tag{2.2}$$

Při ukládání hesel do databází se využívá právě této skutečnosti. V případě vypočitatelnosti by byla daná funkce zcela nepoužitelná. Například při použití 256 bitové hešovací funkce je počet výstupních možností přibližně roven $1{,}158 * 10^{77}$, čímž je při současném výpočetním výkonu nezvládnutelné přiřadit výstupní hodnotu ke vstupní.

Odolnost proti modifikaci vstupu

Nyní si popíšeme další požadavek, který je nutný při využívání hešovací funkce pro digitální podpisy. Snahou je, aby bylo pro výstupný heš $y = h(x_1)$ prakticky nemožné najít jiný vstup x_2 vytvářející stejný heš y. Útočník zná vstupní zprávu x_1 a výstupní heš y, jenže není schopen podvrhnout příjemci vlastní nebo upravenou zprávu x_2 mající stejný heš y. Ale, i zde je určitá pravděpodobnost, viz rovnice 2.3.

$$p = \frac{1}{2^n} \tag{2.3}$$

Odolnost proti kolizím

V úvodním popisu hešovací funkce je zmínka o nekonečném počtu vstupních možností a konečném počtu výstupních možností. Toto implikuje existenci kolizí. Odolnost proti kolizím tedy znamená, že má být výpočetně zcela nemožné najít rozdílné vstupy $x_1 \neq x_2$, pro které platí $h(x_1) = h(x_2)$. Pravděpodobnost p nastání této kolize je dána vztahem 2.4.

$$p \approx \frac{1}{2^{\frac{n}{2}}} \tag{2.4}$$

Pravděpodobnost je menší než v případě prvních dvou požadavků, viz rovnice 2.3 a to díky narozeninovému paradoxu, který byl jedním z hlavních důvodů na vylepšení současných hešovacích funkcí. Proto následovalo vypsání soutěže na novou hešovací funkci, která má být odolnější proti zmíněnému jevu. Výsledkem bylo dne 5. 8 .2015 vydání nového standartu SHA-3 NIST[1]. Podrobný popis narozeninového paradoxu lze najít například v [3].

Teoretické poznatky o hešovacích funkcích byly čerpány z [3],[15].

[1]National Institute of Standards and Technology, Národní institut standardů a technologie

3 POUŽITÉ TECHNOLOGIE

Předchozí kapitola obsahovala obecný popis základních kryptografických systémů. V rámci této kapitoly jsou popsány využívané technologie v manažeru hesel s architekturou klient-server. Jde například o šifrovací algoritmus, použitou hešovací funkci, databázový systém a zvolený programovací jazyk.

3.1 Šifrovací algoritmus AES

Jedná se o symetrickou blokovou šifru založenou na algoritmu Rijndael. Pracuje se vstupním blokem dat o pevné velikosti 128 bitů. AES[1] šifra využívá matici o velikosti 4 x 4 bajtů, pro kterou jsou definovány čtyři operace:

- sub bytes,
- shoft rows,
- mix columns,
- add roud key.

Sub bytes

Operace *Sub bytes* má přímý tvar. Přímá substituce využívá S-box, což je matice o velikosti 16 x 16. Blok mezivýsledku *State array* je převeden na novou hodnotu závisející na hodnotě právě transformovaného bajtu, kde 4 bity vyšší priority určují adresu řádku S-boxu a zbývající 4 bity nižší priority udávají adresu sloupce. Jako příklad uvedeme bajt v binární podobě převedený do hexadecimální podoby $11011011_B \rightarrow DB_{HEX}$. Tato hodnota je transformována podle tabulky přímé substituce na hodnotu 9F. Tabulka je k vidění v [3].

Inverse sub bytes: Používá se v procesu dešifrování. Operace funguje na stejném principu, avšak využívá inverzní substituci.

Shift rows

Operace provádí rotaci řádků (posun bajtů) v matici *State array*. První řádek matice je ponechán beze změny. Ve zbylých řádcích se vždy provádí posun vlevo. Ten činí u druhého řádku posun o 1 bajt, u třetího o 2 bajty a u čtvrtého o 3 bajty.

Inverse shift rows: Využití při procesu dešifrování. Princip je stejný, místo posunu vlevo provádí posun vpravo.

[1] Advanced Encryption Standard, standard pokročilého šifrování

Mix columns

Zmiňovaná operace transformuje každý sloupec matice *State array* zvlášť. Původní sloupec se vynásobí maticí. Každý bajt sloupce je přeměněn na novou hodnotou závisející na všech bajtech daného sloupce.

Add round key

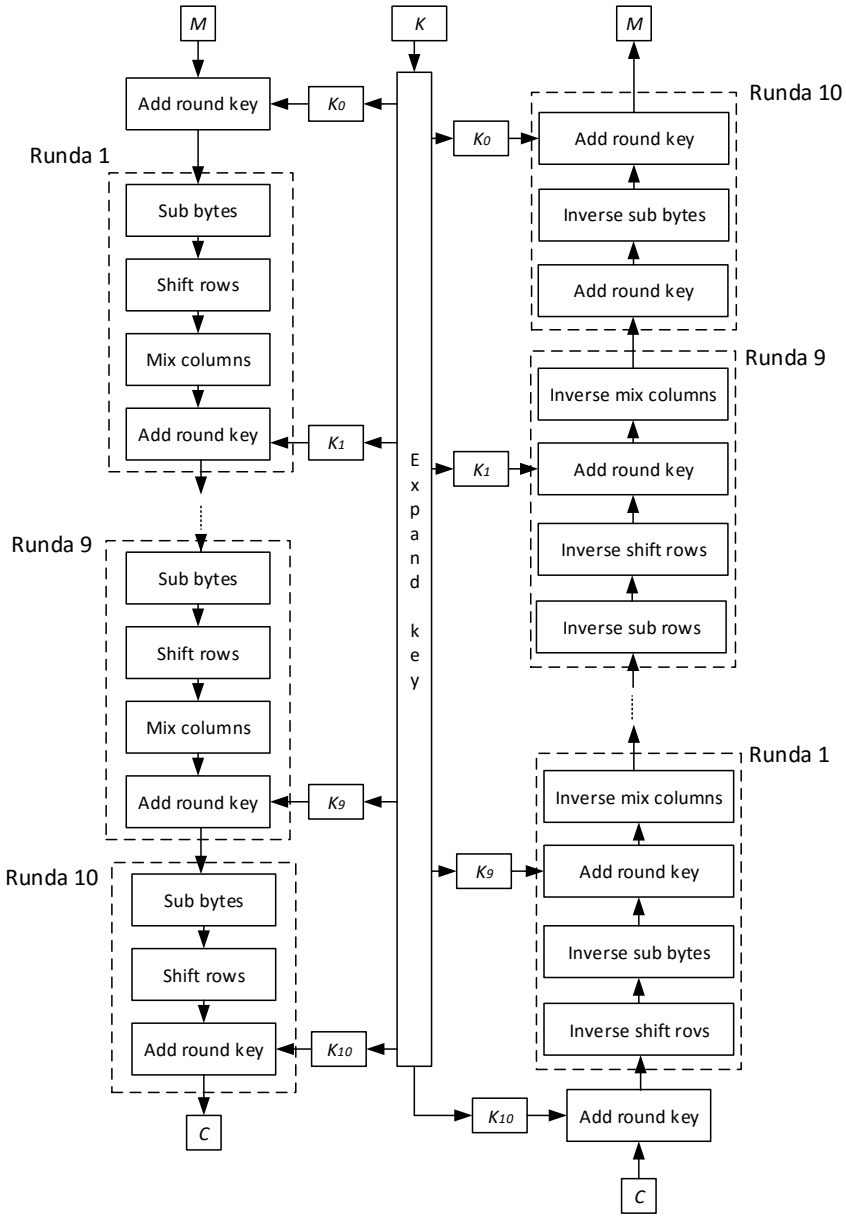
Zde probíhá exklusivní disjunkce matice *State array* s klíčem aktuálně zpracovávané rundy.

Key expand

Ještě zbývá popsat poslední operaci. Jedná se o rozšíření šifrovacího klíče a je popsána varianta se 128 bitovým klíčem. Při procesu šifrování a dešifrování se používá 11 dílčích klíčů formátu 4 x 4 bajtů. Klíč je postupně zapsán po sloupcích do matice o formátu 4 x 4 a použije se jako inicializační klíč K_0. Následující klíče jsou tvořeny po sloupcích v závislosti na předešlých sloupcích. Podrobný princip popisu tvorby klíče lze najít například v [3].

Princip AES

Na začátku šifrování je přičten inicializační klíč K_0 ke zprávě M. Následuje 9 rund složených ze čtyř výše uvedených operací. Do každé operace *Add round key* vstupuje rozšířený klíč K_n, kde n je číslo aktuální rundy. V poslední, desáté rundě se vypustí operace *Mix columns*, aby bylo možné provést dešifrování. Výstupem poslendí rundy je kryptogram C. Dešifrování je prováděno v opačném pořadí a využívá se inverzních operací. Princip lze vidět na obrázku 3.1, šifrování je na levé straně a dešifrování na straně pravé. (překresleno z [11]).



Obr. 3.1: AES šifrování a dešifrování

Informace o AES jsou čerpány z [3],[17].

3.2 Hešovací funkce SHA-256

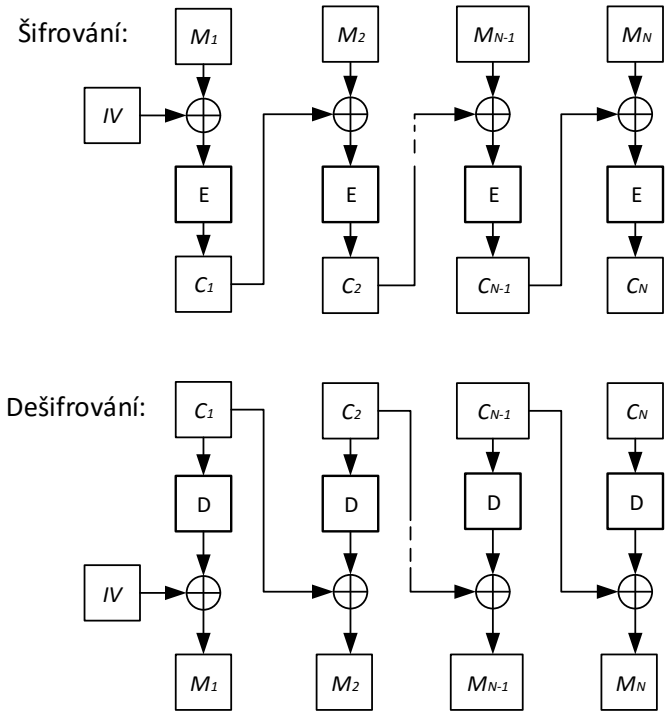
Popisovaná funkce patří do rodiny SHA-2, jenž je navržena NSA[2]. Rodina SHA-2 obsahuje kromě zmiňované funkce další funkce: SHA-224, SHA-384, SHA-512, SHA-512/224 a SHA-512/256. Čísla udávají bitovou délku výstupních hešů. Rodina SHA-

[2]National Security Agency, Národní bezpečnostní agentura

2 byla publikována v roce 2001 a o rok později byla přijata NISTem za standard. Nicméně i přes vyšší bezpečnost, než kterou poskytuje SHA-1, není stále hojně využívaná. Citováno z [15].

3.3 Provozní režim blokové šifry - CBC

Režim CBC[3] je režim pro šifrování zpráv se zpětnou vazbou. Používá k znesnadnění prolomení blokové šifry, CBC je vyobrazen na obrázku A.1.



Obr. 3.2: CBC režim

Obrázek je převzatý z [3].

Zpráva M se rozdělí na bloky o délce p bitů. Před začátkem šifrování je nejprve nutno provést padding, tzn. zarovnat délku posledního bloku zprávy do požadované délky p. Nejčastěji se k tomu přistupuje tak, že se za poslední bit zprávy zařadí logická „1“ a zbytek se vyplní do požadované délky logickou „0“.

Dle obrázku A.1 je zřejmé, že na začátku procesu šifrování je M_1 blok zprávy sečten modulo2 s inicializačním vektorem IV délky l bitů. Ten je unikátní a je generován náhodně, aby se zabránilo shodnosti dvou kryptogramů při šifrování stejné zprávy totožným tajným klíčem. Výstup operace XOR je šifrován a vzniká C_1 blok

[3]Cipher Block Chaining

kryptogramu. Následující blok zprávy je opět vstupem operace XOR. Jenže jako druhý vstup již neslouží inicializační vektor IV, ale předešlý blok kryptogramu C_1. Toto se opakuje, dokud se nedojde k M_N bloku zprávy, tedy k poslednímu bloku. Inicializační vektor je přenášen k příjemci spolu s kryptogramem C, jenž je zřetězen ze všech vzniklých Ci bloků.

Dešifrování probíhá obdobně. Nejprve je přijatý kryptogram C rozdělen na bloky o délce p bitů, poté je provedeno dešifrování prvního bloku kryptogramu. Dešifrovaný blok je sečten modulo2 s přeneseným inicializačním vektorem IV. Výsledkem je první blok původní zprávy. Následující blok původní zprávy je získán prakticky totožně. Jediným rozdílem je, že jako vstup operace XOR slouží předešlý blok kryptogramu C_1. Od posledně získaného bloku zprávy se odstraní všechny logické „0“ až po první logickou „1“ včetně. Zřetězením všech získaných bloků původní zprávy získáme onu původní zprávu M.

Teoretické poznatky byly čerpány z [3].

3.4 Algoritmus HMAC

Algoritmus HMAC[4] se používá v bezpečnosti IP sítí, využívá se například u protokolu SSL. Jde o typ autentizačního kódu zprávy MAC[5] sloužícího pro ověření integrity a autentičnosti zprávy. Výstup je vypočten z hešovací funkce a tajného klíče. Délka výstupu je odvozena od použité hešovací funkce. Tajný klíč musí mít k dispozici odesílatel i příjemce. Síla algoritmu HMAC je úměrná použité hešovací funkci a tajného klíče. I zde platí několikrát uvedená věc, že čím je bitová délka hešovací funkce delší, tím je i výstup HMAC silnější. Vzorec pro výpočet HMAC uvádí následující rovnice 3.1.

$$\mathrm{HMAC}_K(m) = h[(K \oplus \mathrm{opad}) \mid\mid h((K \oplus \mathrm{ipad}) \mid\mid m)] \tag{3.1}$$

Struktura algoritmu HMAC

Popis struktury dle rovnice 3.1.

- h – hešovací funkce,
- K – tajný klíč,
- opad (**o**uter **pad**ding) – vnější zarovnání, jeho hodnota je 0x5c ($5\mathrm{C}_H$, 92_D, 01011100_B),
- ipad (**i**nner **pad**ding) – vnitřní zarovnání, jeho hodnota je 0x36 (36_H, 54_D, 00110110_B),

[4]Keyed-hash Message Authentication Code

[5]Message Authentication Code

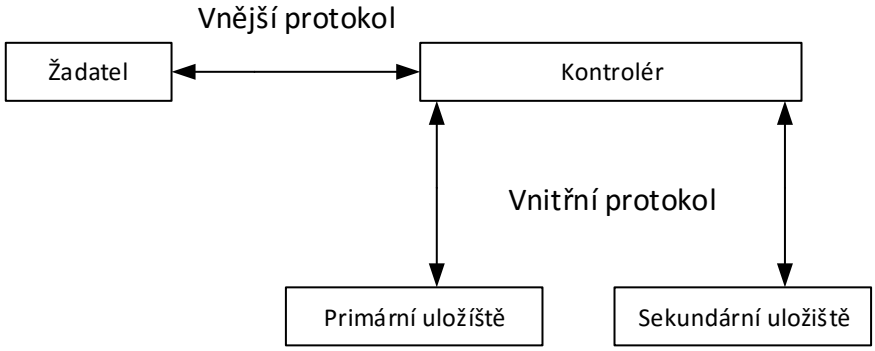
- m – zpráva, ze které se počítá kód.

Konstanty ipad a opad se opakují $b/8$, kde b je bitová délka bloku. Hešovací funkce MD5, SHA-1, SHA-256 apod. pracují s 512 bitovou délkou bloku, tím pádem se tyto konstanty budou v algoritmu vyskytovat 64×.

Teoretické poznatky byly čerpány z [6],[7].

3.5 Autentizační protokol

V projektu manažeru hesel s architekturou klient-server je využíván univerzální protokol anonymní asymetrické autentizace zajišťující nevypočitatelnost a netestovatelnost hodnot. Zmíněný systém je složen ze dvou protokolů. Z vnitřního a vnějšího. Systém je uveden na obr. 3.3.



Obr. 3.3: Autentizační protokol

3.6 Databáze

Serverová databáze je program, umožňující ukládání velkého množství informací v organizovaném formátu, jenž je snadno přístupný prostřednictvím programovacích jazyků (PHP, Java, C++, C#, .NET . . .). V dnešní době jsou databáze všude, kam se podíváme. Nejběžnějším modelem se staly relační databáze. Při tvorbě databáze je nejdůležitější její návrh. Prvním krokem návrhu databázové struktury je nutné si vše pořádně promyslet. Nanečisto si vytvořit několik možných návrhů. Dále je nezbytné si uvědomit k čemu má databáze sloužit, jaké informace bude uchovávat, jaké operace budou v dané databázi prováděny a určit vztah mezi jednotlivými tabulkami. Správný návrh databáze nám může ušetřit spoustu času.

Relace vytváří vztah mezi jednotlivými tabulkami a existuje několik druhů:

- 1:1,
- 1:N,
- M:N.

Popíšeme pouze první dva relační vztahy. O posledním vztahu se pouze krátce zmíníme, jelikož vztah M:N nebude využíván.

Relační vztah 1:1

Je nejjednodušší vztah relace, kde jedna hodnota primárního klíče v hlavní tabulce odpovídá právě jedné hodnotě pole v druhé tabulce. V dřívějším návrhu databáze jsem měl použity dvě tabulky ve vztahu 1:1, ale nakonec byl tento relační vztah tabulek odstraněn a v současném návrhu databáze se nevyužívá. Tabulky vázané tímto vztahem byly sloučeny do jedné.

Relační vztah 1:N

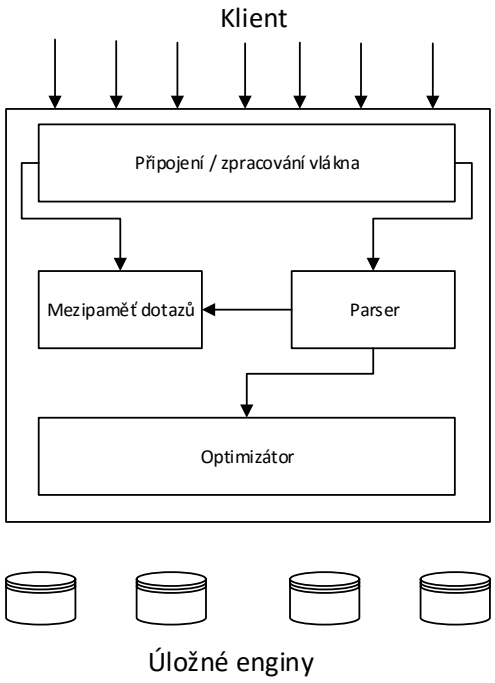
Druhý relační vztah vzniká mezi dvěma tabulkami, kdy jedna hodnota primárního klíče v hlavní tabulce odpovídá několika hodnotám v tabulce druhé. Příkladem použití vztahu 1:N je, že uživatel může mít více záznamů hesel, schválených a čekajících na schválení zařízení, kdežto záznam hesla a zařízení mohou patřit pouze jednomu uživateli. Čerpáno z [19]

3.6.1 Jazyk SQL

SQL je strukturovaný dotazovací jazyk používající se pro práci s daty v relačních databázích. Má za sebou dlouhý vývoj sahající až do roku 1974. Za jeho vývojem stojí firma IBM. V roce 1986 se prosadil jako standart.

3.6.2 Vybraný databázový systém MySQL

MySQL je databázový systém vyvinutý společností MySQL AB, která je v současnosti součástí Oracle Corporation. Není však ve všech ohledech perfektní, ale díky své výkonnosti, spolehlivosti, snadnému užití a hlavně volné šiřitelnosti je lídrem na poli databázových systémů v oblasti webových aplikací. Jde o multiplatformní databázi a z názvu lze odvodit, že využívá dotazovací jazyk SQL. Nejvyšší vrstva obsahuje služby, které nejsou pro MySQL unikátní. Jde o služby jako jsou navázání spojení, autentizace, bezpečnost a podobně. Druhá vrstva je často označována „mozkem“ MySQL. Jsou obsaženy kódy pro parsování dotazů, analýzu, optimalizaci, mezipaměť a ostatní vestavěné funkce (data, čas,...).



Obr. 3.4: GUI správce zařízení

Obsahem poslední vrstvy jsou úložné enginy, které jsou zodpovědné za ukládání a zpřístupnění všech dat. Úkolem enginu není vyřizovat SQL dotazy nebo nekomunikovat s ostatními, nýbrž pouze odpovídat na dotazy serveru. Poznatky jsou čerpány z [16].

3.7 Programovací jazyk

3.7.1 Java

Java je programovací jazyk vyvinutý firmou Sun Microsystems v roce 1995, která byla nedávno koupena firmou Oracle Corporation. Hlavním cílem bylo vytvořit jednodušší a platformě nezávislou alternativu k jazyku C++. Programy napsané v Java jsou spuštěny pomocí nainstalovaného interpreta. Dnes se již využívá JIT[6] překladač, který je součástí JVM a zajišťuje dynamické zkompilování kódu před jeho spuštěním do strojového kódu daného počítače. Zajišťuje zpětnou vazbu na hardware a v reálném čase interpretuje kód. Java patří mezi objektově orientované programovací jazyky, což nám umožňuje jednodušeji reprezentovat objekty reálného světa.

[6]Just In Time

3.7.2 Java RMI

RMI (Remote Method Invocation) je API poskytující mechanizmus pro vytváření distribuovaných aplikací v programovacím jazyce Java. RMI dovoluje objektu běžícím na jednom virtuálním stroji (JVM) volat metody objektu, který běží na jiném virtuálním stroji. Metody, které mohou být volány, jsou deklarovány pomocí Java interface (rozhraní). To deklaruje nějakou množinu metod. Zmíněné metody jsou poté poskytovány vzdáleným objektem. Jednoduše řečeno RMI slouží ke vzdálené komunikaci mezi aplikacemi. Komunikace je řešena pomocí dvou objektů – `stub` (zástupce vzdáleného objektu) a `skeleton` (kostra vzdáleného objektu).

Stub

Neboli zástupce vzdáleného objektu slouží jako brána pro klientskou část aplikace. Instance `stubu` je poskytnuta serverem klientovi a reprezentuje volaný vzdálený objekt. Veškeré odchozí požadavky jsou voláním vzdálených metod z klientské části a jsou směrovány právě přes tuto instanci `stubu`. Pokud je použita nižší verze než Java 2 SDK, v1.2, tak probíhá komunikace mezi `stubem` na straně klienta a skeletonem na straně serveru. V opačném případě odpadá nutnost použití `skeletonu` a komunikují mezi sebou dvě instance `stubu`. Pokud voláme vzdálenou metodu na `stub`, vykoná následující kroky:

- zahájení spojení se vzdáleným virtuálním strojem JVM,
- marshalování[7] parametrů do vzdáleného virtuálního stroje,
- čekání na výsledek volané metody,
- demarshalování a vrácení hodnoty nebo vyjímky,
- vrácení hodnoty volajícímu.

Citováno z [1].

Java JDBC

JDBC je standart pro databázově nezávislé spojení mezi programovacím jazykem Java, širokým rozsahem SQL databází a ostatních tabulkových zdrojů dat. JDBC ovladač umožňuje správně vkládat instance do SQL dotazů.[8]

3.7.3 JavaFX

JavaFX je open-source framework[8] od firmy Oracle Corporation postavený na bázi Javy. Slouží vývoj bohatých aplikací. První verze byla vydána v roce 2008 jako verze

[7]Marshalování je proces přeměny paměťové reprezentace objektu do datového formátu, který je použitelný pro uchovávání nebo pro přenos. V rámci RMI jde o serializaci.

[8]Framework, aplikační rámec

JavaFX1.0. Je považován za srovnatelný s ostatními frameworky jako je například Microsoft Silverlight. JavaFX je často označována jako nástupce Swingu v rámci grafických rozhraní (GUI[9]) na Java platformě. Je dostupná jako veřejné Java API[10]. Obsahuje několik funkcí, díky nimž je upřednostňována pro vývoj bohatých aplikací:

- je napsaná v Javě, což jí umožňuje používat všechny vlastnosti Javy,
- podporuje provázání dat pomocí Java knihoven,
- umožňuje vložit do aplikace webový obsah,
- podporuje přehrávání multimédií,
- nabízí dvě možnosti vytvoření UI a to s pomocí:
  - Java kódu,
  - FXML.

V roce 2014 byla vydána JavaFX 8.0, která oficiálně nahradila zastaralý Swing. Citováno z [14].

FXML

Je vytvořen firmou Oracle Corporation, založený na značkovacím jazyku XML a slouží k popisu uživatelského grafického rozhraní pro JavaFX aplikace. Díky FXML je možné oddělit vytváření uživatelského rozhraní od aplikační logiky kódu. Úkolem je pouze popisovat vzhled aplikace. Citováno z [10], [5].

CSS

Kaskádové styly se používají pro upravení vzhledu komponent uživatelského rozhraní dle specifických požadavků. JavaFX umožňuje pracovat s CSS[11], čehož je využito při úpravě vzhledu grafického rozhraní klienta.

[9] Graphical User Interface, grafické uživatelské rozhraní
[10] Application Programming Interface, rozhraní pro programování aplikace
[11] Cascading Style Sheets

4 NÁVRH ŘEŠENÍ MANAŽERU HESEL

Kapitola pojednává o návrhu funkcí a grafického rozhraní klienta, serveru a databáze.

4.1 Popis vlastností a funkcí klienta

Vzhledem k požadovaným bezpečnostním prvkům je důležité, aby klient implementoval několik ochran bránících odcizení nebo zneužití uložených záznamů hesel. Jak je zmíněno v úvodu žádný systém není neprolomitelný. Hlavním činitelem při prolomení daného systému je selhání lidského faktoru. Proto je nutné, aby uživatel přizpůsobil své chování v digitálním světě základním bezpečnostním pravidlům. Hlavními vlastnostmi navrhovaného manažeru hesel jsou:

- přístup k uloženým heslům přes jedno hlavní heslo,
- umístění databáze hesel mimo klientská zařízení,
- šifrovaná databáze hesel,
- přenos informací o záznamu až na vyžádání uživatele,
- bezpečné spojení klienta se serverem,
- využití protokolu anonymní asymetrické autentizace,
- ochrana při odcizení klientského zařízení.

4.2 Návrh využívaných klíčů

Klíč klienta CLIENT_KEY

Při registraci uživatele nebo spárování zařízení bude vygenerován pomocí kryptograficky silného generátoru v JAVA třídě SecuredRandom [13] 256 bitový unikátní klíč klienta *CLIENT_KEY*. Klíč bude uložen v nepozměněné podobě do souboru `client.key` vytvořeného v adresáři s klientskou části manažera hesel. V případě odcizení nebude samotný *CLIENT_KEY* útočníkovi užitečný, jelikož pro možnost získání *ENC_KEY* je nutné projít procesem autentizace. K té je nutná znalost *CLIENT_KEY*, uživatelského jména a hesla.

Hlavní klíč MAIN_KEY

Při registraci nového uživatele je pro generování *MAIN_KEY* použit také kryptograficky silný generátor. Hlavní 128 bitový klíč se používá jako tajný klíč šifrovacího algoritmu AES. Sloužící k zašifrování záznamů daného uživatele. Záznamy jsou uloženy v databázi serveru. Hlavní klíč je pro všechna klientská zařízení pod daným účtem stejný, pro uživatele je jedinečný a musí si jej zaznamenat. Při ztrátě klíče

způsobeného odcizením, poškozením zařízení nebo smazáním souboru obsahujícího hlavní klíč nebude schopen uložené záznamy v databázi dešifrovat.Zvolený bezpečnostní mechanizmus je pro útočníka prakticky neprolomitelný.

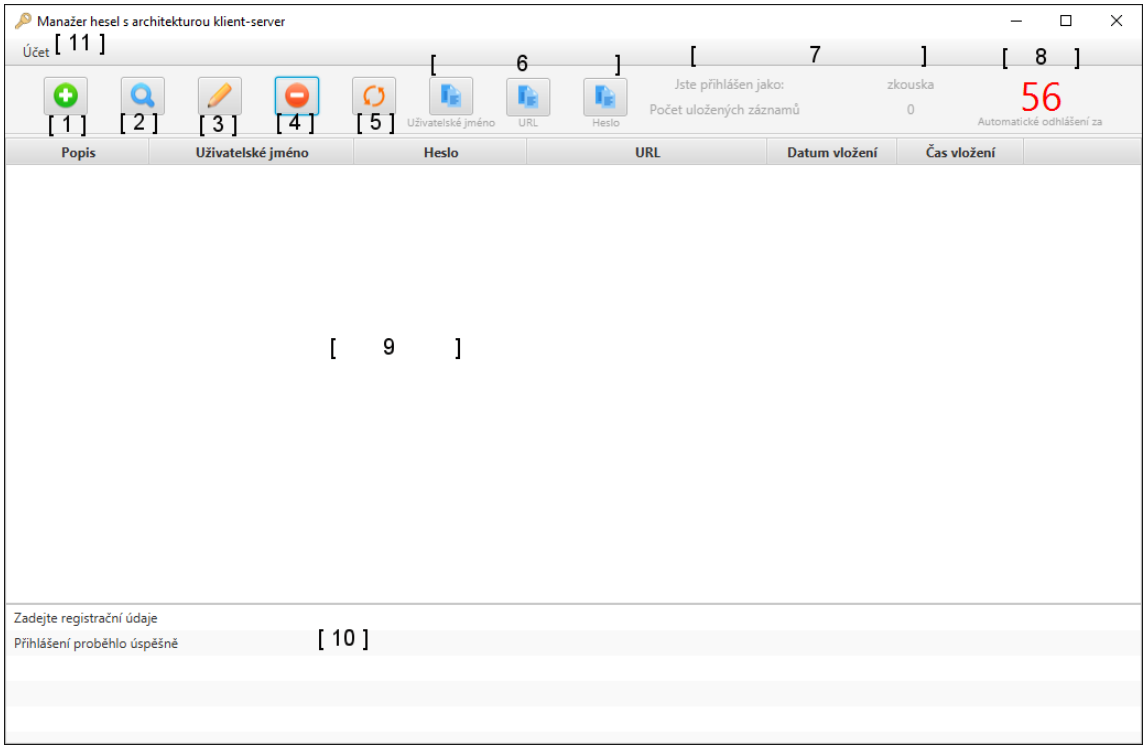
Často se stává, že při registraci účtu uživatel nevěnuje dostatečnou pozornost pokynům. Z tohoto důvodu je navrženo opsání klíče, které neumožní uživateli pokračovat dříve, než hlavní klíč úspěšně opíše. Uživatel také bude mít možnost si hlavní klíč později zobrazit pomocí funkce v menu klientské části.

Šifrovací klíč ENC_KEY

Hlavní klíč je nutné bezpečně uchovávat. Bylo zvoleno následující řešení. Protože klient potřebuje při každém spuštění načíst *MAIN_KEY*, tak nelze pro uložení využít například hešovací funkci (pro její jednosměrnost). Z tohoto důvodu musíme využít symetrický nebo asymetrický algoritmus. Byl vybrán symetrický algoritmus AES, protože je bezpečný, efektivní a málo náročný na paměť. Tajný klíč *ENC_KEY* je také vytvořen pomocí kryptograficky silného generátoru při registraci uživatele nebo spárování zařízení. Je uložen jako atribut záznamu zařízení v databázi serveru. Pro každé zařízení je unikátní. V případě přihlášení uživatele je získán z databáze. Slouží pro šifrování i dešifrování hlavního klíče *MAIN_KEY*, který je uložen v adresáři klientské části manažeru v souboru `main.key`.

4.3 Navrhované grafické rozhraní

Vzhled klientské části programu lze vidět na obr. 4.1. Každá důležitá část programu je označena pomocí hranatých závorek a popsána níže.



Obr. 4.1: GUI klienta

[1 – 6]Tlačítka:

V horní části GUI klientské části manažeru je k dispozici 8 tlačítek typu Button. Tlačítko [1] slouží pro vložení nového záznamu hesla. Po kliknutí se zobrazí okno obsahující textová pole, viz obr. 4.3. Do nich uživatel vyplní informace o heslu. Po kliknutí na tlačítko (2) zobrazené na obrázku 4.3 se uloží šifrovaný záznam do serverové databáze. Tlačítko [2] je určeno pro získání šifrovaných dat ze serveru, která jsou dešifrována a zobrazena v tabulce [9]. Tlačítko [3] zajišťuje editaci dříve uložených záznamů. Po kliknutí bude zobrazeno stejné okno jako na obrázku 4.3. V textových polích jsou vyplněny údaje editovaného záznamu. Následující tlačítko [4] zajišťuje smazání záznamu hesla. Jedno z posledně očíslovaných tlačítek je označeno [5]. Jeho funkcí je aktualizování seznamu hesel. Může být použito i pro prodloužení platnosti relace se serverem. Skupina tlačítek označena [6] slouží pro zkopírování informace označeného záznamu hesla. Pod každým tlačítkem je popis udávající jaká informace bude zkopírovaná do systémové schránky.

[7 - 8] Informace pro uživatele

Skupina popisků označená [7] je pro uživatele informativní. Je zde vypsáno uživatelské jméno přihlášeného uživatele a počet jeho záznamů hesel uložených v databázi. Pod označením [8] se skrývá popisek typu Label. Informuje o zbývajícím čase platnosti relace mezi klientem a serverem. V případě vypršení časového limitu dojde v klientské části k zablokování funkce a zobrazení přihlašovacího formuláře.

[9] Tabulka:

Jako tabulka slouží komponenta TableView skládající se ze sloupců TableColumn. Dominuje grafickému rozhraní. Po úspěšném přihlášení, v případě existence záznamů hesel pod přihlášeným účtem, se v tabulce objeví seznam všech záznamů. Je vyplněn pouze sloupec popis, datum a čas vložení záznamu. Ostatní podrobnosti, jako je uživatelské jméno, heslo a URL, se přenášejí ze serverové databáze až na vyžádání uživatele. Inicializuje se kliknutím na tlačítko [2].

[10] LOG:

Na obrázku 4.1 je k vidění komponenta ListView vypisující záznamy akcí, které uživatel v rámci dané relace provedl. Záznam akcí je důležitý nejen z informativního hlediska, ale také z důvodu bezpečnosti. Může se stát, že se uživatel na chvíli vzdálí od zařízení. Zapomene vypnout klientskou část manažeru hesel. Jestliže nevyprší čas pro automatické odhlášení, může dojít k neoprávněnému použití klientské části. Uživatel díky výpisu akcí následně zjistí, zda byl klient bez jeho vědomí použít.

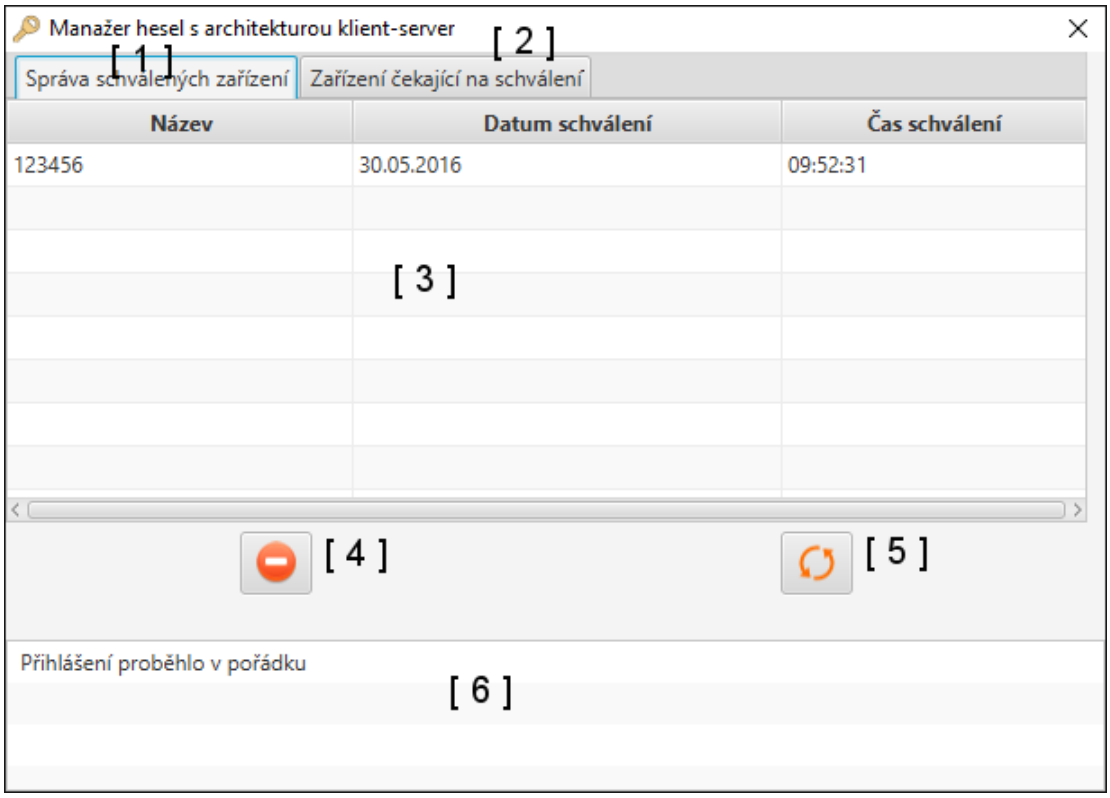
[11] Menu:

V rámci menu vidíme položku účet, ve které se po rozbalení objeví následující možnosti:

- přidat / odebrat zařízení,
- spárovat tuto instalaci s již existujícím účtem,
- zobrazit hlavní klíč,
- zadat hlavní klíč.

Přídat / odebrat zařízení obr. 4.2 slouží ke spuštění správce zařízení. Pro vstup do správce zařízení je vyžadováno opětovné přihlášení. Přihlašovací údaje se vyplňují do komponent TextField. Při úspěšném přihlášení se zobrazí komponenta TabPane obsahující dvě záložky - komponenty Tab [1, 2]. Každá komponenta Tab obsahuje tlačítka a tabulku (TableView) složenou ze sloupců (TableColumn). První záložka [1] je aktivní idned po úspěšném přihlášení a je zobrazena na obrázku 4.2. Slouží ke

správě zařízení, která mají autorizovaný přístup do databáze. Zařízení jsou vypsána v tabulce [3]. Přístup schváleným zařízením lze odebrat kromě právě aktivního zařízení. Druhá záložka [2] slouží k potvrzení nebo odmítnutí zařízení čekajících na autorizování přístupu do databáze hesel. Jejich seznam je zobrazen v tabulce [3]. Aby mohl uživatel používat nové zařízení, musí jej nejdříve schválit v této tabulce. V případě odmítnutí zařízení čekajícího na schválení nezíská přístup do databáze. Tlačítko [3] zobrazené na obrázku zajišťuje odebrání autentizace schválenému zařízení. Pomocí tlačítka [5] lze aktualizovat seznam schválených zařízení. Provedené akce jsou zaznamenány a zobrazeny v komponentě ListView [5]. Pro uživatele mají pouze informativní charakter.

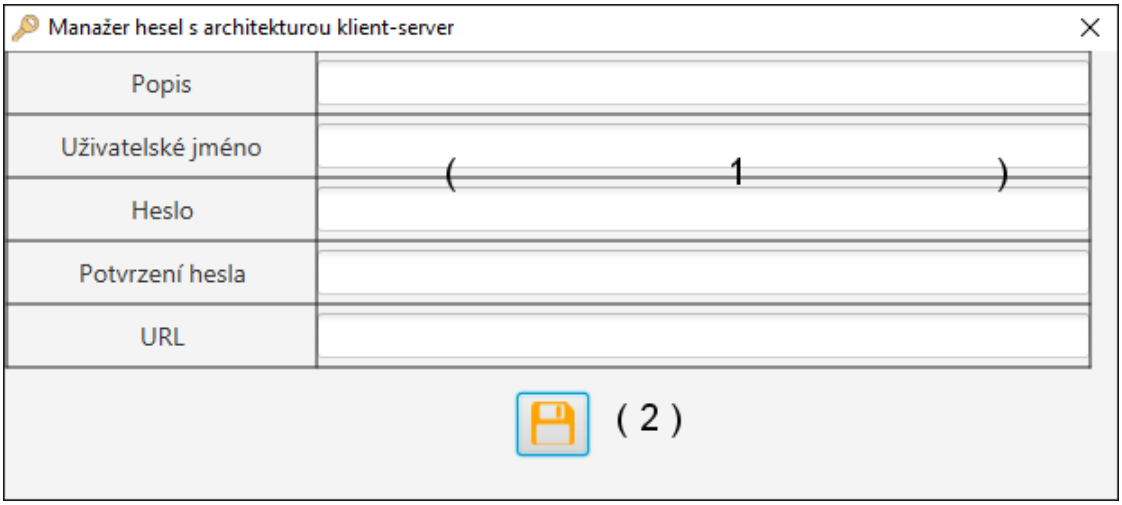


Obr. 4.2: GUI správce zařízení

Spárovat tuto instalaci s již existujícím účtem: Pokud chce uživatel používat manažer hesel i na jiných zařízeních, tak musí využít právě popisovanou funkci. Vyplní uživatelské jméno, emailovou adresu, PIN, heslo a zadá libovolně zvolený párovací kód. Údaje si musí uživatel zkontrolovat. Z bezpečnostních důvodů manažer hesel neupozorní na špatně zadané údaje. Párovací kód si uživatel uchová do té doby, než potvrdí autentizaci ve správci zařízení z některého již potvrzeného zařízení.

Zobrazit hlavní klíč: Po kliknutí se v textovém poli zobrazí hlavní klíč.

Zadat hlavní klíč: Sem uživatel vkládá hlavní klíč *MAIN_KEY* zaznamenaný při registraci. Funkce se využije pouze v případě, že se uživatel přihlásil na novém zařízení. Pro úspěsné načtení hlavního klíče je vyžadováno opětovné spuštění klientské části.



Obr. 4.3: GUI okna pro přidání nového záznamu

Podklady k návrhu GUI byly čerpány z [4].

4.4 Návrh záznamu hesla

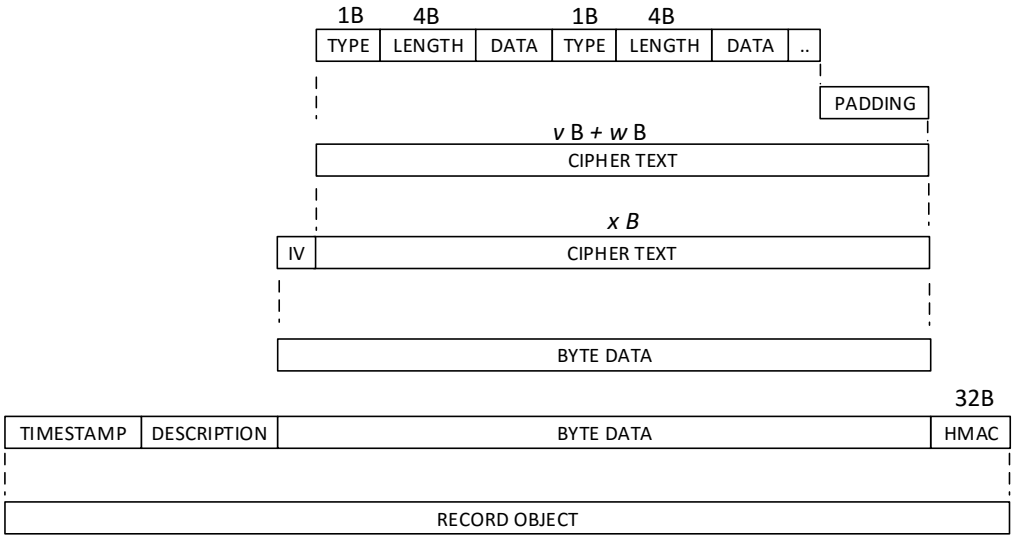
V návrhu nutné stanovit, jak má vypadat záznam hesla, jenž bude výstupem klienta. Jak již bylo zmíněno, bude využívat blokové šifry AES s CBC režimem a paddingem souvisejícím s blokovou šifrou. Na následujícím obrázku 4.4 vidíme navržený objekt *Record* včetně vytvoření šifrových dat.

Typ – TYPE

Má pevnou délku 1 bajt, teoreticky může nabývat až 256 hodnot. Každá hodnota informuje o jaký typ informace se jedná, viz tab. 4.1. Položka TYPE umožní do budoucna rozšíření počtu ukládaných informací o heslech.

VALUE	TYPE
0	USERNAME
1	PASSWORD
2	URL

Tab. 4.1: Hodnoty pole TYPE



Obr. 4.4: Návrh záznamu hesla

Délka – LENGTH

Nese informaci o bitové délce (má velikost 2 bajty , čili maximální délka uložených dat je $2^{16} - 1 = 65535$ bitů) uložených dat, čímž umožňuje zpětné rozložení řetězce o délce v bajtů.

Data – DATA

Data obsahují pole bajtů reprezentující data typu informace.

Šifrový text – CIPHER TEXT

Šifrový text je složen ze všech dříve popsaných údajů. Jsou zarovnány paddingem o délce w bajtů a zašifrovány. Výsledná délka šifrového textu je $v + w$ bajtů (je vždy k násobek bajtové velikosti šifrovaného bloku – x bajtů).

Inicializační vektor – IV

Je využíváno AES šifrování s CBC režimem. S tím souvisí nutnost vygenerovat náhodný inicializační vektor pomocí kryptograficky silného generátoru v Java třídě SecureRandom. Je následně vložen před *CIPHER TEXT*. Inicializační vektor má velikost 16 bajtů.

Veškerá data – ALL DATA

Představuje zašifrované pole bajtů obsahující inicializační vektor a všechny informace o hesle.

Popis – DESCRIPTION

Popis je nezašifrovaný text, který si uživatel volí pro identifikaci daného záznamu.

HMAC

HMAC je vypočten z *ALL DATA*, potřebný tajný klíč bude vypočítán ze vztahu $HMAC\_KEY$ = SHA256 ($MAIN\_KEY$). Tento klíč bude pro všechna zařízení stejný a slouží k ověření integrity dat záznamu hesla.

TIMESTAMP

Časová známka nesoucí informaci o datu a času vytvoření záznamu hesla.

Záznam hesla – RECORD OBJECT

Jde o objekt reprezentující záznam hesla, který obsahuje popsané atributy. V serverové části je doplněn o identifikátor záznamu a uživatele.

4.5 Návrh serverové části

Návrh databáze

Navržený databázový model je uveden na obrázku 4.5. Obsahuje 6 tabulek. Začneme popisem tabulky *users*. Jedná se o hlavní tabulku obsahující pět atributů, ve kterých jsou uloženy základní informace o uživateli. Každý uživatel je identifikován celočíselným primárním klíčem *user_id*. Je automaticky inkrementován podle MySQL atributu *AUTO_INCREMENT*. Zvolení primárního klíče vyplynulo z nutnosti jeho jedinečnosti. Té se nejjednodušeji docílí právě číselnou inkrementací. Navíc bude tento primární klíč využíván jako cizí klíč pro ostatní záznamy ve zbylých tabulkách. Další čtyři atributy obsahují registrační informace uživatele, jde například o uživatelské jméno, e-mailovou adresu, heslo a PIN. Atributy heslo a PIN budou uchovány ve formě 256 bitového heše. Z tohoto důvodu byl zvolen datový typ BLOB. Uživatelské jméno a e-mailová adresa musí být také jedinečné. Toho se docílí ošetřením proti duplicitě v kódu na straně klienta a serveru. Před registrací bude uživatel informován, zda jedinečnost těchto údajů splnil. Nyní bude popsána tabulka *records*. Z názvu

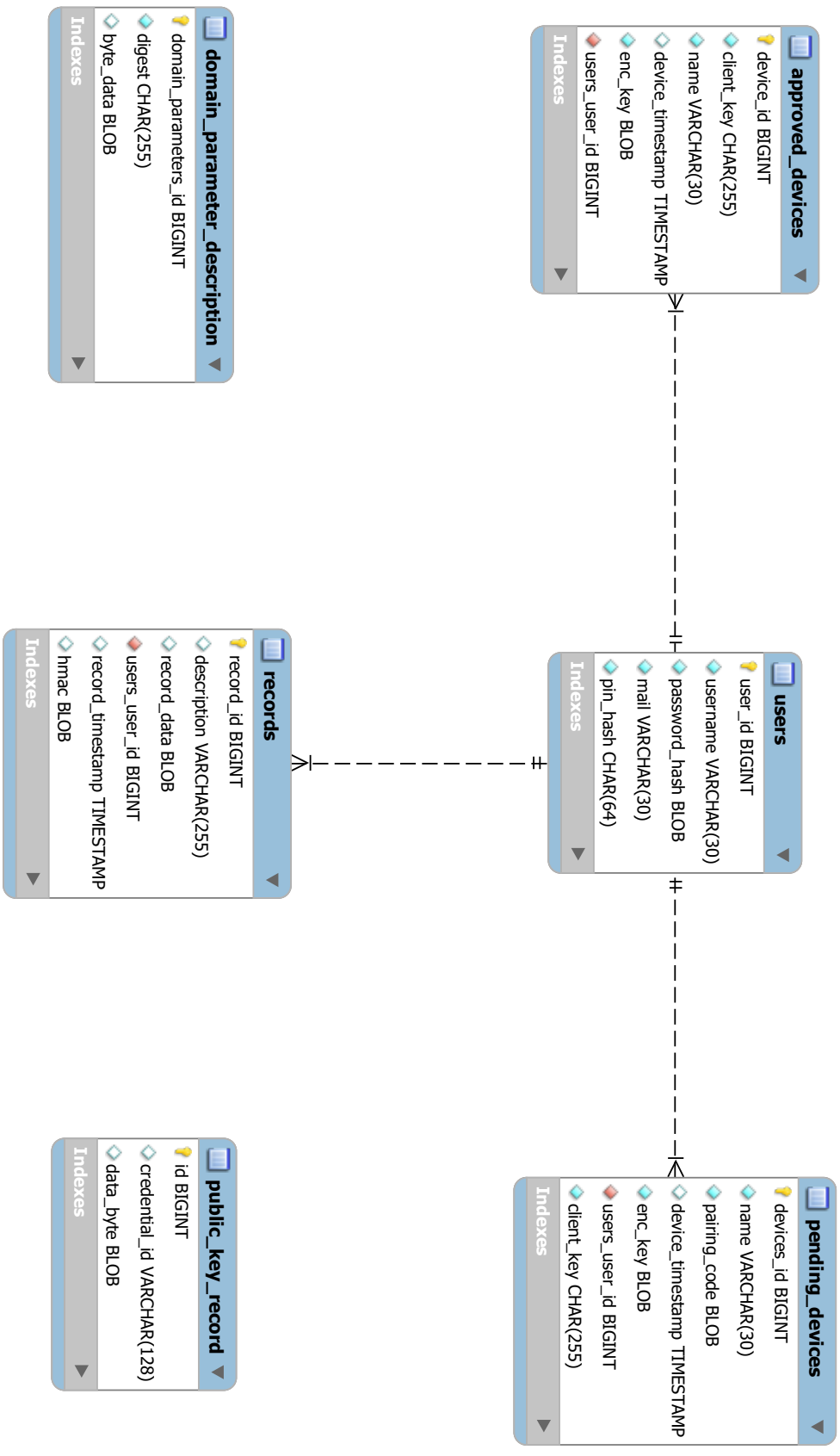
vyplývá, že tabulka bude obsahovat jednotlivé záznamy hesel uživatelů. Každý záznam bude opět jednoznačně identifikován celočíselným primárním klíčem *record_id*, který je opět automaticky inkrementován. Tabulka obsahuje navíc ještě dalších pět položek, kde *description* obsahuje nešifrovaný popis daného záznamu. Důvodem je informovanost uživatele, co se v záznamu hesla nachází. Ke stejnému účelu slouží atribut *record_timestamp* obsahující časovou známku vytvoření záznamu. Entita *record_data* obsahuje údaje o záznamu hesla v šifrované podobě, proto je zvolen datový typ `blob`. Položka *users_user_id* je cizím klíčem, je ve vztahu N:1 k tabulce *users* (jeden uživatel může mít N záznamů hesel, ale každý záznam hesla může patřit pouze k jednomu uživateli). Poslední položka *hmac* slouží pro ověření integrity a autentičnosti šifrovaných dat v položce *record_data*.

V následujících dvou tabulkách *pending_devices* a *pending_devices* se uchovávají informace o zařízeních čekajících na schválení a schválených. Tabulky obsahují položky s údaji o zařízeních, například jméno zařízení, časovou známku, šifrovací klíč apod. První tabulka obsahuje navíc pouze položku *pairing_code* obsahující heš párovacího kódu. Způsob volby primárního klíče u obou tabulek není třeba popisovat – je úplně stejný jako u tabulky *users* a *records*. Obě navíc obsahují *users_user_id* sloužící jako cizí klíč a jsou taktéž ve vztahu N:1 k tabulce *users*.

Tabulky *public_key_record* a *domain_parameter_description* obsahují data důležitá pro funkčnost protokolu anonymní asymetrické autentizace. Model navrhované databáze lze vidět na obrázku 4.5.

Návrh serverových funkcí

Serverová část bude komunikovat s klientovou části pomocí Java RMI. Množina vzdálených metod, které bude vzdálený objekt poskytovat, je nadefinována v Java Interface. Toto rozhraní musí implementovat jak serverová, tak i klientská část manažeru hesel. Klient obdrží po úspěšné autentizaci klíč. Bude platný pouze po určitý čas. Klíčem je pole bajtů vrácené autentizačním protokolem. Časová platnost klíče se obnovuje komunikací se serverem. Po jejím vypršení dojde k přerušení spojení. Klient bude informován o nutnosti opětovné provedení autentizace. Uživatel se musí znovu přihlásit, pokud bude chtít pokračovat v práci s klientem. Implementované metody budou obstarávat komunikaci s databází. Jedná se například o ukládání, editování, zobrazení a odstranění záznamů hesel, ověřování autentizačních údajů, přidání nové nebo odebrání stávající autentizace zařízení, autentizování či odmítnutí nově přidaného zařízení apod.



Obr. 4.5: EER model navrhované databáze

5 REALIZACE MANAŽERU HESEL S ARCHITEKTUROU KLIENT-SERVER

Poslední kapitola se zabývá hlavní částí bakalářské práce, tedy realizací projektu manažeru hesel s architekturou klient-server. Realizace manažeru hesel probíhala v integrovaném vývojovém prostředí (IDE) NetBeans IDE, které je vlastněno firmou Oracle Corporation. Projekt s názvem KeyManager je založen jako Maven projekt obsahující několik Maven modulů. Každý modul je zodpovědný za určitou oblast funkcí. Celkem se KeyManager skládá z šesti modulů, jmenovitě z:

- `KeyManagerServer` - modul se serverovou částí manažeru hesel,
- `KeyManagerObjects` - modul s využívanými objekty v manažeru hesel,
- `KeyManagerIfaceRMI` - modul s RMI rozhraním,
- `ExpiringMap` - modul s časovou mapou,
- `aalg` - modul s protokolem anonymní asymetrické autentizace,
- `KeyManagerClient` - modul s klientskou částí manažeru hesel.

Moduly jsou navzájem na sobě závislé a v následujících částech blíže specifikuji a vysvětlím jejich funkčnost s krátkou ukázkou některých důležitých částí kódu.

5.1 Modul `KeyManagerIfaceRMI`

V tomto modulu se nachází balíček `cz.vutbr.feec.bbct.keymanager.ifacermi` obsahující pouze jediný soubor a tím je rozhraní `RmiInterface`. V rozhraní je deklarovaná množina metod, kterou poskytuje vzdálený objekt. V tomto případě Java třída `Server` v modulu `KeyManagerServer`. Při vytváření rozhraní bylo nutné dodržet dvě podmínky. První z nich je, že RMI rozhraní musí dědit z rozhraní `java.rmi.Remote` a druhou podmínkou je, že každá metoda z množiny metod musí být deklarována s propagací výjimky `RemoteException`. Tato výjimka je nadtřídou všech výjimek souvisejících s komunikací se vzdáleným objektem.

V rozhraní je deklarováno 16 metod, které obecně zajišťují komunikaci mezi serverovou a klientskou částí manažeru hesel. Jde například o registraci uživatele, správu zařízení, přenos seznamu záznamů uživatele, identifikaci zařízení, zjištění dostupnosti uživatelského jména apod. Kompletní výčet metod, včetně popisu jejich účelu, vstupních a výstupních parametrů lze nalézt v dokumentaci JavaDOC umístěné v příloze bakalářské práce.

Zmíněné parametry metod často pracují s objekty, jejichž třídy jsou součástí modulu `KeyManagerObjects`. Proto modul `KeyManagerIfaceRMI` obsahuje závislost na modulu, který obsahuje třídy objektů.

5.2 Modul `KeyManagerClient`

Součástí modulu je klientská část manažeru hesel s architekturou klient-server. Ta zprostředkovává komunikaci mezi uživatelem a serverem. Jedním z hlavních pilířů klienta je grafické rozhraní GUI. To je realizováno pomocí softwarové platformy JavaFX postavené na Javě, viz 3.7.3. Modul s klientskou částí obsahuje celkem 4 balíčky:

- `cz.vutbr.feec.bbct.keymanager.client.functions`,
- `cz.vutbr.feec.bbct.keymanager.client.functionsImpl`,
- `cz.vutbr.feec.bbct.keymanager.client.gui`,
- `cz.vutbr.feec.bbct.keymanager.client.images`.

5.2.1 Balíček `functions`

Obsahem je několik Java rozhraní. Každé je implementováno v jedné Java třídě v balíčku `functionsImpl`. Snahou bylo pojmenovat rozhraní podle typu metod, které deklarují, respektive podle grafických scén využívající deklarované metody. Podobně jako v případě `RmiInterface` jsou metody jednotlivých rozhraní popsané i se vstupními a výstupními parametry v dokumentaci JavaDOC, dostupné v příloze práce.

5.2.2 Balíček `functionsImpl`

Balíček obsahuje Java třídy, kde každá třída implementuje jedno rozhraní. Podle těchto rozhraní jsou třídy pojmenovány, navíc na konci jména každé třídy je `Impl`. Některé třídy mají více implementovaných metod, než je počet deklarovaných v rozhraní.

Za zmínku stojí například metoda `createNewRecord` ve třídě `NewRecordImpl` sloužící pro přidání nového záznamu. Zkrácený výpis je ochuzený o inicializaci proměnných, získání časové známky, ošetření výjimky metody `hmac`, vytvoření objektu `Record` a jeho vrácení, viz výpis 5.1. Na této metodě bude vysvětlen způsob vytvoření zašifrovaného pole dat.

Formálním parametrem metody je komponenta *gridPane* typu `GridPane`. Ta obsahuje další komponenty `Label`, `TextField` a `PasswordField`. Pro vytvoření nového záznamu hesla nás zajímají pouze poslední dva typy komponent obsahující uživatelem vyplněné informace o záznamu. Dle zadání bylo požadavkem vytvořit strukturu metody a objektu pro možnou rozšiřitelnost počtu informací o záznamu, aniž by bylo nutné zasahovat do aplikační logiky. Rozšiřitelnosti je dosaženo pomocí řádků kódu s číslem 4 až 22. Proměnná *number* je získána z počtu řádků v komponentě *gridPane*. Každý řádek odpovídá jednomu typu informace o záznamu. Proměnná

childrens obsahuje seznam všech uzlů, které jsou součástí vstupního parametru, tedy komponenty typu `GridPane`.

Cyklus na řádcích 4 až 22 proběhne $(n-1)\times$, kde n je počet řádků *gridPane*. Řádek 6 a 7 přetypovává uzel na pozici $(i+number)$ na typ `TextField` a ukládá do proměnné *dataTextField*. Proměnná *contentType* obsahuje přetypovanou proměnnou i a slouží jako identifikace typu informace záznamu. Následně je *contentType* vložen do pole bajtů *type* o velikosti 1B, z toho vyplývá omezení maximálního počtu typů informací o záznamu na 256. Na řádku 11 a 12 probíhá získání obsahu informace zadané uživatelem z proměnné *dataTextField* a následné převedení do pole bajtů *contentByteArray*. Dále musíme získat informaci o délce dat získaných v předešlém kroku. K tomu slouží řádek 13 a 14. Následuje vytvoření pomocné proměnné *temp*, do které je uložen obsah z proměnné *recordByte*. Pole bajtů *contentInfo* je získáno spojením dvou polí –*type* a *contentArrayLenght*. Pole o velikosti 5B obsahuje typ a délku zpracovávané informace o záznamu. Další pole bajtů *content* vzniká taktéž spojením dvou polí. Tentokrát dříve spojeného pole *contentInfo* s polem *contentByteArray* obsahujícím data zpracovávané informace záznamu. Toto pole obsahuje typ, délku a data informace o záznamu. Následuje poslední spojení polí – pole z pomocné proměnné *temp* s dříve získaným polem.

V prvním cyklu je proměnná *temp* nulová. Během dalších cyklů k sobě připojuje další pole bajtů zpracovávaných informací záznamu. Tím je docíleno možnosti rozšíření o informativní položky záznamu hesla, jelikož se pouze zvýší počet vykonávaných cyklů a velikost výsledneho pole bajtů. Každá hodnota informace záznamu je zatížena 5B režií zajišťující možnost správného rozlišení jednotlivých záznamů v jednom poli bytů.

Cyklus vynechává uzel na pozici 0 a 3. Na pozici 0 se nachází popis záznamu, který je nešifrovaný a má informativní charakter. Pozice 3 obsahuje komponentu `PasswordField` s ověřením ukládaného hesla. Hodnotu není nutné ukládat. Rozlišovat komponenty `TextField` a `PasswordField` má smysl pouze z hlediska GUI, kde `PasswordField` zobrazuje místo znaku znak „*“. Po skončení cyklu je výsledné pole zašifrováno a ze zašifrovaných dat proveden výpočet HMAC pomocí volání metody *hmac* na řádku 23.

Následuje vytvoření instance objektu `Record` s atributy *description*, *timestamp*, *hmac* a *data*. Při případném rozšíření počtu informací o záznam není nutné zasahovat do datové struktury objektu `Record`, protože dojde pouze k nárůstu velikosti atributu *data*. Ostatní atributy zůstanou nedotčené.

Listing 5.1: Metoda pro vytvoření nového záznamu

```
1  public Record createNewRecord(GridPane gridPane) {
2  int number = (gridPane.getRowConstraints().size());
3  ObservableList<Node> childrens = gridPane.getChildren();
4     for(int i = 1; i <= number - 1; i++) {
5        if (i != 3) {
6           TextField dataTextField = (TextField)
7              (childrens.get(i + number));
8           byte contentType = (byte) i;
9           byte[] type = ByteBuffer.allocate(1).
10               put(contentType).array();
11          byte[] contentByteArray = dataTextField.
12               getText().getBytes();
13          byte[] contentArrayLenght = ByteBuffer.allocate(4).
14               putInt(contentByteArray.length).array();
15          byte[] temp = recordByte;
16          byte[] contentInfo = ArrayUtils.
17               addAll(type, contentArrayLenght);
18          byte[] content = ArrayUtils.
19                 addAll(contentInfo, contentByteArray);
20          recordByte = ArrayUtils.
21                 addAll(temp, content);
22             }
23     hmac = HMAC(recordByte);
24         }
25     }
```

Další ukázkou je část kódu zobrazená ve výpisu 5.2. Jedná se o metodu `encrypt` ze třídy `AESImpl`. Metoda je deklarována s variabilním počtem vstupních argumentů, kde je variabilní parametr inicializován jako pole. Metoda vždy vrací pole zašifrovaných dat. Důvodem pro takový návrh metody byla skutečnost, že se pro šifrování záznamů hesel používá klíč *MAIN_KEY*. Musí být taktéž zašifrován pomocí *ENC_KEY*. Bylo by tedy zbytečné mít dvě prakticky totožné metody. Na řádku číslo 2 je podmínka, která je splněna, pokud je délka pole vstupních argumentů větší než 1. V případě splnění podmínky je druhý prvek tohoto vstupního argumentu nahrán do obsahu proměnné *encyptKey*. Takto je metoda volána v případě registrace uživatele nebo uložení hlavního klíče na novém zařízení. Slouží k zašifrování hlavního klíče *MAIN_KEY*, který je v tomto případě v „roli“ dat připravených k šifrování. V ostatních případech je použit jako šifrovací klíč již existující hlavní klíč.

Ve výpisu není uvedena inicializace šifrovací funkce. Na řádku 8 lze vidět, že data, která mají být šifrována, jsou vždy reprezentována na nulté pozici pole argumentů. Pro možnost dešifrování dat je nutné uložit pole bajtů obsahující inicializační vektor. Způsob spojení je na řádku 9. Je totožný se způsobem spojování polí v metodě `createNewRecord`. Posledním řádkem je vrácení pole vytvořeného na předchozím řádku. Obdobně je navržena metoda `decrypt`, kde se pro dešifrování také používají dva rozdílné klíče. Proto je metoda jako v předchozím případě deklarována s variabilní počtem vstupních argumentů.

Listing 5.2: Metoda šifrování dat

```
1  public byte[] encrypt(byte[]... data) {
2    if (data.length > 1) {
3      encryptKey = data[1];
4          } else {
5              encryptKey = mainKey;
6          }
7    //INICIALIZACE ŠIFROVACÍ FUNKCE
8    encrypted = cipher.doFinal(data[0]);
9    encryptedData = ArrayUtils.addAll(initialVector, encrypted);
10   return encrypted;
11     }
```

Poslední metodu popsanou metodou bude metoda `register` ve třídě `KeyManagerImpl` vyobrazená ve výpisu 5.3. Z názvu vyplývá, že metoda je volána v případě registrace uživatele. Vstupním parametrem je komponenta *loginGridPane* typu `GridPane` obsahující uživatelem vyplněná data k registraci. Výpis je opět zkrácený. Data z komponenty `loginGridPane` jsou získána stejným způsobem jako u popsané metody `createNewRecord`. Se získaným heslem a PINem uživatele se však zachází odlišně. Je získán otisk pomocí SHA-256 hešovací funkce zavolané metodou `digestMessage`, viz řádek číslo 3. Následuje vytvoření objektu `User` reprezentující registrační údaje uživatele. Vytvoření objektu je na čtvrtém řádku výpisu. Proměnné *username* a *mail* jsou textové řetězce. Kód na řádku 7 zajišťuje vygenerování, pomocí kryptograficky silného generátoru v Java třídě SecureRandom, 256 bitového klíče *clinetKey* sloužícího jako *CLIENT_KEY*. Ostatní klíče *MAIN_KEY* a *ENC_KEY* jsou generovány stejně, s tím rozdílem, že jsou 128 bitové. Následně je volána metoda `encrypt` ze třídy `AESImpl`. Výsledkem je pole bajtů obsahující zašifrovaný *mainKey*. Pole je následně uloženo do souboru *main.key* pomocí metody `writeToFile` ze třídy `fileRwImpl`. Následuje vytvoření objektu `ApprovedDevice` a provedení registrace autentizačním protokolem. Do něho vstupuje objekt `Credentials` vytvořený metodou `createCredentials` z komponenty *loginGridPane* a klientského klíče *clientKey*. Součástí registrace je uložení objektů `User` a `ApprovedDevice` do serverové data-

báze. Metoda vrací hlavní klíč *mainKey* v hexadecimální podobě, aby si jej mohl uživatel zaznamenat.

Listing 5.3: Registrační metoda třídy KeyManagerImpl

```
1  public String register(GridPane loginGridPane) {
2    password =(((TextField) childrens.get(7)).getText())
3              .getBytes();
4    passwordDigest = digestMessage(password);
5    User user = new User(username, passwordDigest,
6                         mail, pinDigest);
7    byte[] clientKey = SecureRandom.getSeed(32);
8    byte[] encKey = SecureRandom.getSeed(16);
9    byte[] encryptedMainKey = aes.encrypt(mainKey, encKey);
10   fileRwImpl.writeToFile(encryptedMainKey, "main.key");
11   ApprovedDevice approvedDevice =
12     new ApprovedDevice(encKey, deviceName, timestamp);
13   authenticationImpl.register
14     (createCredentials(loginGridPane));
15
16 return Hex.encodeHexString(mainKey);
17 }
```

5.2.3 Balíček images

Zmíněný balíček obsahuje obrázkové soubory ve formátu `png`, které jsou využívány v grafickém rozhraní GUI, většinou jako pozadí tlačítek. Všechny obrázkové soubory, kromě `saveButtonImage`, `infoButtonImage` a ikony manažeru hesel, jsou šířeny pod licencí Creative Commons Attribution 3.0 License. Povinností uvádět autorství. U všech souborů, kromě `saveButtonImage`, `infoButtonImage` a ikony manažeru hesel je autorem Icojam.com. Zbylé soubory jsou šířeny pod licencí Creative Commons Attribution-NoDerivs 3.0 Unported. V tomto případě je autorem Icons.com.

5.2.4 Balíček gui

Obsahuje spustitelnou hlavní třídu `Main.java`, která vytvoří scénu z FXML souboru `KeyManager.fxml`. Nastaví scéně ikonu a vlastnosti, jako minimální výšku a šířku. Scéna je objekt a je základní kontejner pro obsah v grafu scény. Dále se v balíčku nachází pět FXML souborů, ve kterých je nadefinován grafický vzhled jednotlivých scén. Pro návrh grafického vzhledu byl použit program `SceneBuilder`. V balíčku je stejný počet kontrolérů. Každý kontrolér je přiřazen k jednomu FXML souboru. Jde vlastně o obyčejnou Java třídu, ve které jsou nadefinovány kontejnery pomocí

anotace `@FXML`. Obsahuje také obslužné metody stisku tlačítek volající metody z balíčku `functionsImpl` skrze rozhraní z balíčku `functions`, případně vytvářejí nové scény, které jsou nadefinované v ostatních FXML souborech. V kontrolérech se nenachází aplikační kód, ale pouze kód upravující vzhled grafického rozhraní klienta. Jedinou výjimkou je formátování řetězců, které mají být zobrazeny v komponentách `TableColumn`.

5.3 Modul KeymanagerObjects

Úkolem modulu je nadefinovat datovou strukturu objektů. Jsou využívány napříč celým programem a vzniknou vytvořením instance jedné z šesti tříd, které se v modulu nacházejí. Jako příklad je vybrána třída `Record` reprezentující šablonu pro vytváření objektů se záznamem hesla. Pro zkrácení výpis 5.5 neobsahuje konstruktory, přetížené metody, gettery a settery, ale pouze proměnné včetně jejich JPA anotací. Anotace na prvním řádku `@Entity` je součástí každé perzistentní třídy. Za touto anotací následuje další anotace `@Table` s atributem `name = "records"`. Říká nám, že objekt `Record` bude případně vložen do tabulky se jménem `records`. Řádek 4 až 7 nám říká, že proměnná *recordId* typu `long` je identifikátorem objektu (anotace `@Id`). Hodnota je automaticky generována (anotace `@GeneratedValue`) a identifikátor je namapován do sloupce (anotace `@Column`), který je identifikován svým jménem `record_id` (část 6. řádku – name=`"record_id"`). Obdobným způsobem se pracuje s dalšími proměnnými, čímž dojde k namapování celého objektu.

Listing 5.4: Proměnné třídy Record

```
1  @Entity
2  @Table(name = "records")
3  public class Record implements Serializable {
4      @Id
5      @GeneratedValue(strategy = GenerationType.IDENTITY)
6      @Column(name = "record_id")
7      private long recordId;
8      @Column(name = "users_user_id")
9      private long userId;
10     private String description;
11     @Column(name ="record_timestamp")
12     private Timestamp recordTimestamp;
13     private byte[] recordData;
14     private byte[] hmac;
15   }
```

5.4 Modul KeyManagerServer

Server je druhou částí manažeru, starající se především o ukládání dat. To probíhá prostřednictvím objektově relační mapování (ORM), zajišťující konverzi dat mezi relační databází a objektově orientovaným programováním. Mapování je součástí šablon, podle kterých dochází k vytváření objektů, viz sekce 5.3. Modul serveru je složen ze 4 balíčků:

- `cz.vutbr.feec.bbct.keymanager.server.serverImpl`.
- `cz.vutbr.feec.bbct.keymanager.server.aalg`,
- `cz.vutbr.feec.bbct.keymanager.server.dao`,
- `cz.vutbr.feec.bbct.keymanager.server.daoimpl`,

5.4.1 Balíček serverImpl

V balíčku jsou celkem tři třídy. Důležitou třídou je třída `Server`. V jejím konstruktoru dochází k vytvoření `EntityManagerFactory`, z perzistentní jednotky `KeyManagerObjects` nacházející se v XML souboru `persistence.xml` v modulu `KeyManagerObjects`. Další důležitou a obsáhlou částí kódu v konstruktoru je inicializace protokolu anonymní asymetrické autentizace. Jedná se o načtení primárního a sekundárního úložiště ze souboru, vytvoření instance registrů `PublicKeyRegister`, `DomainParameterRegister` a mapy s časově omezenou dobou životnosti vložených záznamů, viz 5.6. Do této mapy vkládá autentizační protokol vypočtené klíče na základě autentizace, společně s objektem `DeviceInfo`. Časová životnost těchto záznamů je obnovována aktivitou uživatele. Pokud bude uživatel neaktivní, dojde ke smazání záznamu a bude se muset znovu autentizovat.

Třída `ServerMain` obsahuje hlavní spustitelnou metodu `main`, ve které se vytváří instance popsané třídy `Server`. Podstatným krokem je volání metody `run` třídy `Server`. Metoda slouží pro zprovoznění RMI. Je zde vytvoření bezpečných soketů, registrů na portu `1099`. Provázání objektu autentizačního protokolu a serveru se jmennou službou. Server implementuje metody z rozhraní `RmiInterface` z modulu `KeyManagerIfaceRMI`, pomocí kterých probíhá komunikace s klientskou částí. Dále je zde cyklus pro vypnutí serverové části. Navíc díky cyklu je držena reference na vytvořený objekt třídy `Server`. Pro vypnutí serveru lze zadat do konzole: `STOP`.

5.4.2 Balíček aalg

Autentizační protokol pracuje s primárním a sekundárním úložištěm klíčů a také s registry `DomainParametersRegister` a `PublicKeyRegister`. Primární a sekundární úložiště jsou načteny ze souborů při spuštění serveru. Záznamy z registrů se ukládají do databázové tabulky. Proto bylo nutné vytvořit třídy, které umožňují

konverzi dat registrů mezi relační databází a strukturou dat používaných autentizačním protokolem. Například ve třídě `DomainParametersRegisterImpl` metoda `addDomainParameters` vrací objekt `DomainParametersId` prezentující proměnnou *id* uloženého objektu `DomainParametrsDescription` (dále `DPD`). Na řádku číslo 3 a 4 je `DPD` převeden to bajtové podoby, která by měla sloužit jako primární klíč v databázi. To by byl problém, proto je na řádku 5 ze získaného pole vytvořen otisk hešovací funkcí SHA-256. Výstupem hešovací metody je taktéž pole bajtů. Z toho důvodu je před vrácením převedeno do řetězcové podoby. Šestý a sedmý řádek ukazuje proměnnou *result* typu `List` s výsledky Hibernate dotazu, kdy se porovnával získaný otisk současného `DPD` s dříve uloženými záznamy. V případě, že je list prázdný, následuje uložení současného `DPD` vytvořením instance třídy `DomainParametersDescriptionDAO` v modulu `KeyManagerObjects`. Výsledkem je namapovaný objekt a obsahující otisk DPD, pole bajtů `DPD` a identifikátor objektu představující `DomainParametersId`. Identifikátor uloženého objektu je zpět konvertován do podoby, jakou požaduje autentizační protokol.

V případě, že byl nalezen záznam objektu, jenž má shodný otisk s právě vypočteným, tak je z něho pomocí getteru získán identifikátor, který je taktéž konvertován do očekávaného návratového typu. Podobně se postupuje v případě `PublicKeyRegister`. Třídy v tomto balíčku jsou základem stejné s původními třídami v protokolu, avšak jsou doplněné o atributy sloužící pro ORM.

Listing 5.5: Proměnné třídy Record

```
1  public DomainParametersId addDomainParameters
2    (DomainParametersDescription description) {
3        byte[] descriptionByte =
4          description.writeToByteArray();
5        String dataDigest = GetSha256(descriptionByte);
6        List<DomainParametersDescriptionDAO> result =
7          em.createQuery(criteria).getResultList();
8        if (result.size() == 0) {
9          //Uložení do databáze
10          long id = domainParametersDescriptionDAO.
11             getDomainParametersId();
12          domainParametersId = new DomainParametersId(id);
13       } else {
14          domainParametersId =
15           new DomainParametersId(result.get(0).
16             getDomainParametersId());
17       }
18     return domainParametersId;
19   }
```

5.4.3 Balíček dao

Součástí balíčku jsou DAO[1] rozhraní deklarující množiny metod, které mohou být volány na instance tříd nacházejících se v balíčku `daoimpl`. Deklarované metody poskytují vyhledávání, ukládání, mazání, převod objektové reprezentace objektu na řádkovou strukturu a opačně. Bližší popis metod lze nalézt v dokumentaci JavaDoc.

5.4.4 Balíček daoimpl

Jedná se o balíček s Java třídami, které implementují DAO rozhraní. Třída implementuje veškeré metody poskytované DAO rozhraním. Je tedy přímo zodpovědná za práci s daty a jejich převodem mezi relační a objektově orientovanou reprezentací. Jako ukázka kódu zobrazená ve výpisu 5.6 je zvolen konstruktor třídy `RecordDAOImpl` a implementaci metody `addNewRecord`. Konstruktor je uveden na řádku 1 až 4. Řádek číslo 3 ukazuje vytváření objektu *em* typu `EntityManageru` z objektu *emf* typu `EntityManagerFactory` předaného parametrem konstruktoru. Stejné konstruktory obsahují i ostatní třídy. Popíšeme metodu obsaženu na řádcích 6 až 12. Vstupními argumenty metody jsou objekt záznamu hesla *record* a proměnná

[1] Data Access Objects

userId slouží jako identifikátor uživatele požadujícího uložení jeho záznamu. Sedmý řádek kódu získá instanci transakce, která je zahájena na následujícím řádku. Objektu je nastaven identifikátor uživatele a následuje vložení objektu do databáze. Řádek 10 slouží k ukončení transakce. Ve výpisu není provedeno ošetření transakce, aby nezůstala otevřena a nezpůsobila nefunkčnost serveru. Ošetření je provedeno pomocí bloku `try-catch-finally`.

Listing 5.6: Metoda obsluhující uložení záznamu do databáze

```
1    public RecordDAOImpl(EntityManagerFactory emf) {
2         this.emf = emf;
3         em = emf.createEntityManager();
4     }
5
6  public void addNewRecord(long userId, Record recod) {
7         EntityTransaction transaction = em.getTransaction();
8         transaction.begin();
9         recod.setUserId(userId);
10        em.persist(recod);
11        transaction.commit();
12    }
```

5.5 Modul aalg

Jedná se o protokol anonymní asymetrické autentizace. Jeho úkolem je autentizace uživatele, výpočet autentizačního klíče z uživatelského jména, hesla a unikátního klientského klíče. Více informací lze nalézt v [12].

5.6 Modul ExpiringMap

Modul obsahuje stejnojmennou mapu umožňující časovou životnost vložených záznamů. Autorem ExpiringMap je Jonathan Halterman a projekt je napsán pod svobodnou licencí Apache Licence 2.0. Časová mapa je využívána pro omezení životnosti vypočtených autorizačních klíčů *commonKey* vložených autentizačním protokolem. Projekt lze nalézt na uvedené adrese v citaci [9].

5.7 MySQL databáze

Požadavkem řešení manažeru hesel bylo vytvoření stálé databáze. Pro návrh MySQL databáze je použit program MySQL Workbench firmy Oracle Corporation. Program

umožňuje exportovat návrh databáze jako SQL skript, který následně slouží pro vytvoření databázové struktury. MySQL Workbench je také využitelný přímo pro běh a správu MySQL databáze. Databázová struktura byla vytvořena pomocí vyexportovaného SQL skript – je součástí přílohy. Pro představu níže uvedena část SQL skriptu, která slouží pro vytvoření tabulky `users`.

Listing 5.7: SQL skript pro vytvoření databáze.

```
1  -- Schema KeyManagerDB
2  -- -----------------------------------------------------
3  -- Table `KeyManagerDB`.`users`
4  -- -----------------------------------------------------
5  CREATE TABLE IF NOT EXISTS `KeyManagerDB`.`users` (
6    `user_id` INT NOT NULL AUTO_INCREMENT,
7    `username` VARCHAR(20) NOT NULL,
8    `password` VARCHAR(32) NOT NULL,
9    `mail` VARCHAR(30) NOT NULL,
10   `pin` VARCHAR(32) NOT NULL,
11   PRIMARY KEY (`user_id`),
12   UNIQUE INDEX `user_id_UNIQUE` (`user_id` ASC),
13   UNIQUE INDEX `mail_UNIQUE` (`mail` ASC))
14 ENGINE = InnoDB;
```

6 ZÁVĚR

Cílem bakalářské práce byla implementace manažeru hesel s architekturou typu klient server. Manažer se skládá ze dvou částí – klienta a serveru. Klientská část manažeru hesel provádí šifrování dat a je doplněna vhodným uživatelským rozhraním. To je implementováno pomocí softwarové platformy JavaFX postavené na bázi platformy Java. Šifrování dat a ostatní aplikační logika byla implementovaná pomocí programovacího jazyku Java. Spojení se serverovou částí je realizováno technologií Java RMI umožňující vzdálenou komunikaci mezi virtuálními stroji JVM. Vzdálené metody jsou deklarovány v RMI rozhraní. Komunikace mezi klientem a serverem je zabezpečena použitím protokolu TLS. Díky implementaci správce zařízení existuje možnost odebrání autentizace zařízení a přihlášení se z více zařízení pod stejný uživatelský účet.

Serverová část se stará o práci s daty umístěnými v MySQL databázi. Autentizaci zajišťuje univerzální protokol anonymní asymetrické autentizace, který je součástí serverové části. Byla implementovaná pomocí jazyku Java. S databází komunikuje prostřednictvím frameworku Hibernate umožňujícím objektově-relační mapování (ORM). Cíle bakalářské práce byly splněny.

V budoucnosti je možné upravit klientskou část přidáním funkce umožňující dynamické přidávání typů informací o záznamech hesel. Toho se docílí úpravou uživatelského rozhraní. Jinou část nebude nutné upravovat. Jedním z dalších možných vylepšení je přidání správy uživatelského účtu. Jako aktuálně lepší možnost k vylepšení manažeru hesel je v úpravě klientské části, aby byla schopna pracovat na tabletech a chytrých telefonech využívajících operační systém Android a iOS.

LITERATURA

[1] *An Overview of RMI Applications*. Oracle. [online]. 2.12.2015 [cit. 2015-12-02]. Dostupné z: https://docs.oracle.com/javase/tutorial/rmi/overview.html

[2] BURDA, Karel. *Bezpečnost informačních systémů*. Brno: FEKT Vysokého učení technického v Brně. 1. 11. 2005. 104s

[3] BURDA, Karel. *Aplikovaná kryptografie*. 1. vyd. Brno: VUTIUM, 2013, 255 s. ISBN 978-80-214-4612-0.

[4] *Client Technologies: Java Platform, Standard Edition (Java SE) 8*. Oracle. [online]. 2.12.2015 [cit. 2015-12-02]. Dostupné z: http://docs.oracle.com/javase/8/javase-clienttechnologies.htm

[5] *FXML*. Wikipedia: the free encyclopedia. [online]. 2001- [cit. 2016-04-10]. Dostupné z: https://cs.wikipedia.org/wiki/HMAC

[6] *HMAC*. Wikipedia: the free encyclopedia. [online]. 2001- [cit. 2015-12-10]. Dostupné z: https://en.wikipedia.org/wiki/FXML

[7] *Základy kryptografie pro manažery: HMAC*. Clever and Smart. [online]. 2001- [cit. 2015-12-10]. Dostupné z: http://www.cleverandsmart.cz/zaklady-kryptografie-pro-manazery-hmac/

[8] *Java SE Technologies - Database*. Oracle. [online]. 2.12.2015 [cit. 2015-12-06]. Dostupné z: http://www.oracle.com/technetwork/java/javase/jdbc/index.html

[9] *GitHub – jhalterman/expiringMap*. GitHub. [online]. 2.12.2015 [cit. 2015-12-06]. Dostupné z: https://github.com/jhalterman/expiringmap

[10] *JavaFX: Mastering FXML*. Oracle. [online]. 2.12.2015 [cit. 2015-12-02]. Dostupné z https://docs.oracle.com/javase/8/javafx/fxml-tutorial/why_use_fxml.htm

[11] LEVICKÝ, Dušan. *Kryptografia v informačnej bezpečnosti*. Vyd. 1. Košice: Elfa, 2005, [7], 266 s. ISBN 80-8086-022-x.

[12] LEŽÁK, Petr. *Univerzální protokol anonymní asymetrické autentizace*. Elektrorevue - Internetový časopis (http://www.elektrorevue.cz), 2015, roč. 2015, č. 4, s. 142-149. ISSN: 1213- 1539.

[13] Oracle. *Java Platform, Standard Edition 7: API Specification* [online]. 2011 Dostupné z: http://docs.oracle.com/javase/7/docs/api/index.html

[14] KISHORI, Sharan. *Learn JavaFX 8.* Apress: c2015, 1210 s. ISBN 978-1-4842-1142-7.

[15] SCHNEIER, Bruce a N FERGUSON. *Practical cryptography.* Indianapolis: Wiley, c2003, xx, 410 s. ISBN 0471223573.

[16] SCHWARTZ, Baron, Peter ZAITSEV a Vadim TKACHENKO. *High performance MySQL. 3rd ed. Cambridge* [Mass.]: O'Reilly, c2012, xxviii, 793 s. ISBN 1449314287.

[17] Šilhavý, Pavel. *Datová komunikace. Brno: Vysoké učení technické v Brně*, 2011. s. 1-211. ISBN: 978-80-214-4455-3.

[18] *Worst passwords of 2014.* TemsID [online]. [cit. 2016-05-26]. Dostupné z: https://www.teamsid.com/worst-passwords-of-2014/

[19] *Začínáme s MySQL 4. – návrh databáze.* Živě. [online]. 2.12.2015 [cit. 2015-12-02]. Dostupné z: http://www.zive.cz/clanky/zaciname-s-mysql-4–navrh-databaze/sc-3-a-102907/

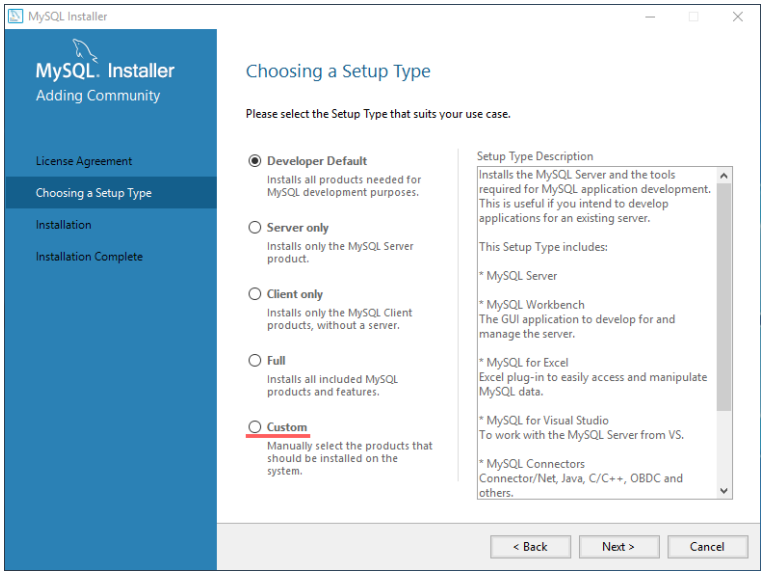
SEZNAM PŘÍLOH

A NÁVOD NA ZPROVOZNĚNÍ MANAŽERU HESEL

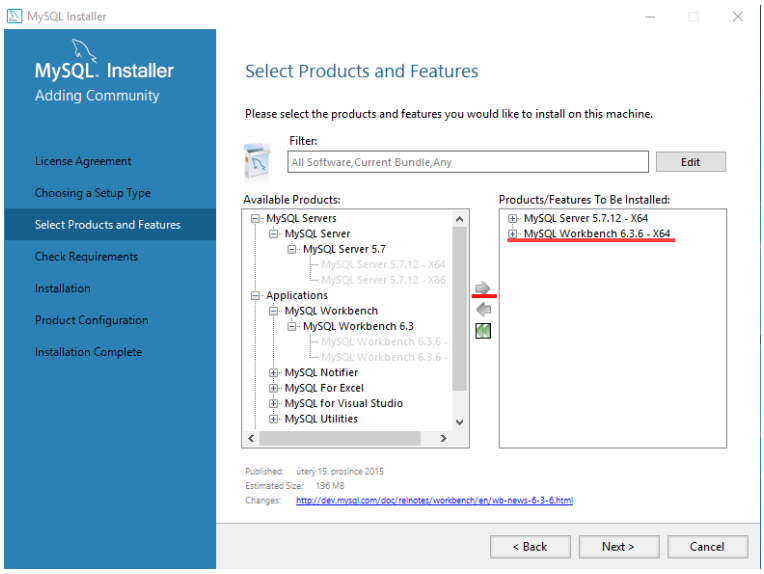
A.1 Instalace MySQL serveru

Instalační MySQL balíček pro operační systém Windows lze stáhnout zde. V případě operačního systému Linux jsou ke stažení repositáře APT, Yum a SUSE zde. Stažení instalačního balíčku je podmíněno vlastnictvím Oracle účtu. Návod je psán pro instalaci na operačním systému Widnows.

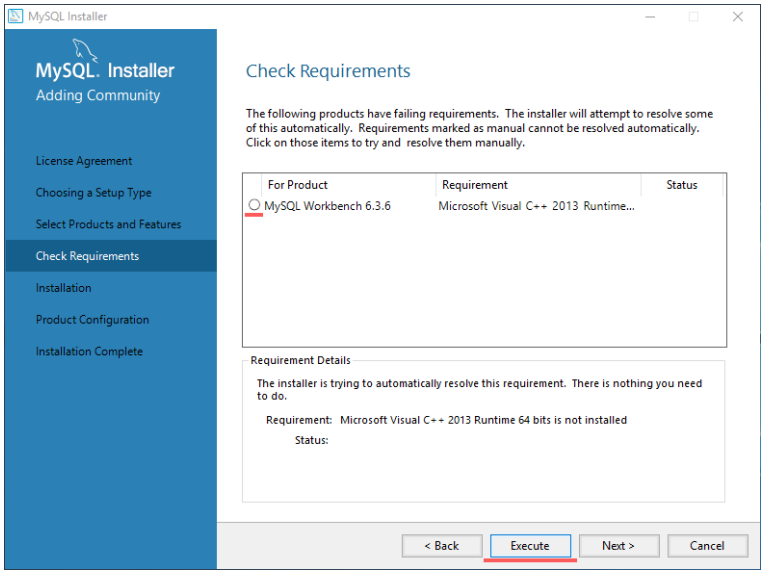
Spusťte stažený instalační balíček. Po akceptování licenčních podmínek vyberte typ instalace `Custom` a klikněte na tlačítko `Next`, viz obrázek níže.



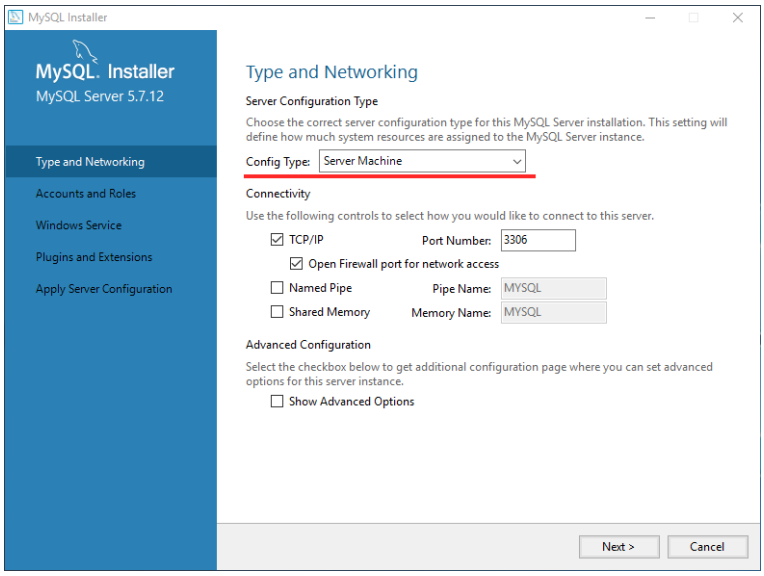
V následujícím kroku vyberte položku `MySQL Server 5.7.12` dle bitové verze operačního systému nacházející se v části `Avaliable Products`. Následně klikněte na podtrženou šipku na obrázku. Instalovat produkt `MySQL Workbench` není nutné. V případě jeho instalace je pak možné spustit skript pro vytvoření databáze `KeyManagerDB` nejen pomocí příkazového řádku, ale také pomocí GUI. Klikněte na `Next`.



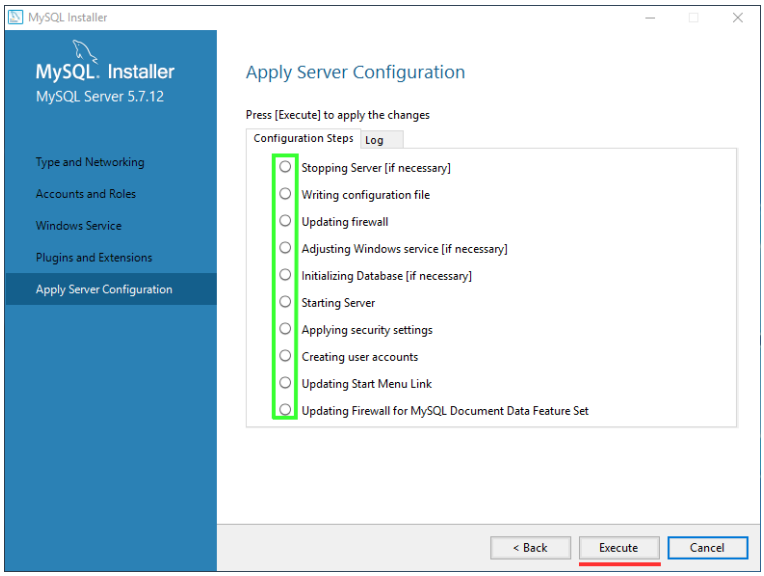
Zde se kontrolují závislosti, které jsou nutné k běhu v předešlém kroku zvolených produktů. Pokud je vyžadovaná instalace nějaké závislosti, tak klikněte na tlačítko `Execute`. Před pokračováním v instalaci by měly být všechny závislosti označené zelenou fajfkou. Po dokončení všech předešlých kroků se přistoupí k samotné instalaci vybraných produktů. V dalším kroku změňte atribut `Config Type` na `Server`



`Machine`, ostatní položky nechte beze změny. Poté zvolte heslo uživatele `root` v kolonce `MySQL Root Password`, které nastavte na `root`. Klikněte na `Next`. Dále nemusíte nic měnit, pouze případně odškrtněte automatické spuštění MySQL serveru při startu počítače. V tomto případě budete muset před spuštěním serverové části manažeru hesel nejdříve spustit MySQL server, pokud nepoběží.



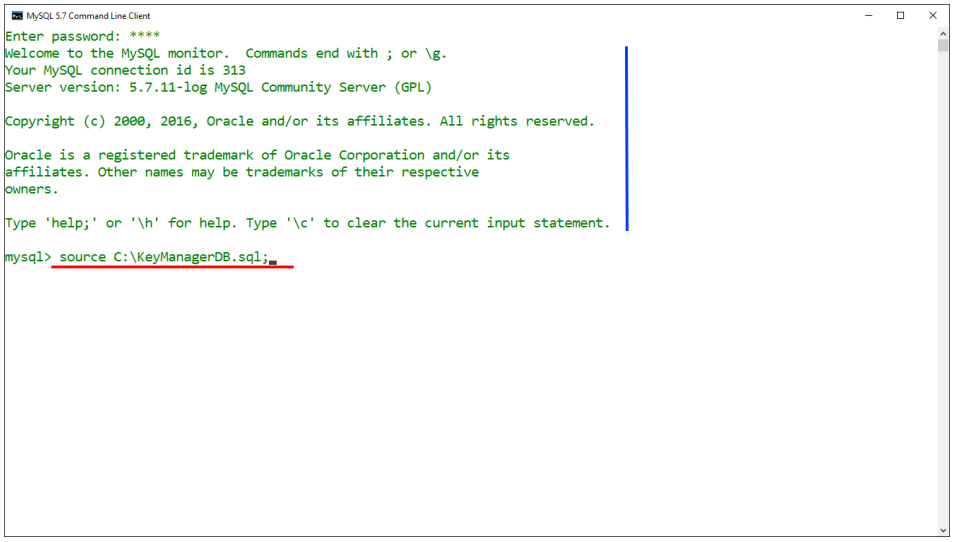
Následuje poslední krok, kterým je konfigurace MySQL serveru. Klikněte na `Execute`. Po skončení konfigurace bude celá zeleně označená oblast vyplněna zelenými fajfkami. Tímto je instalace MySQL dokončena. V případě jiných přihlašovacích údajů k MySQL serveru nebo pokud server běží na jiném portu, tak postupujte dle pokynů A.4.2.



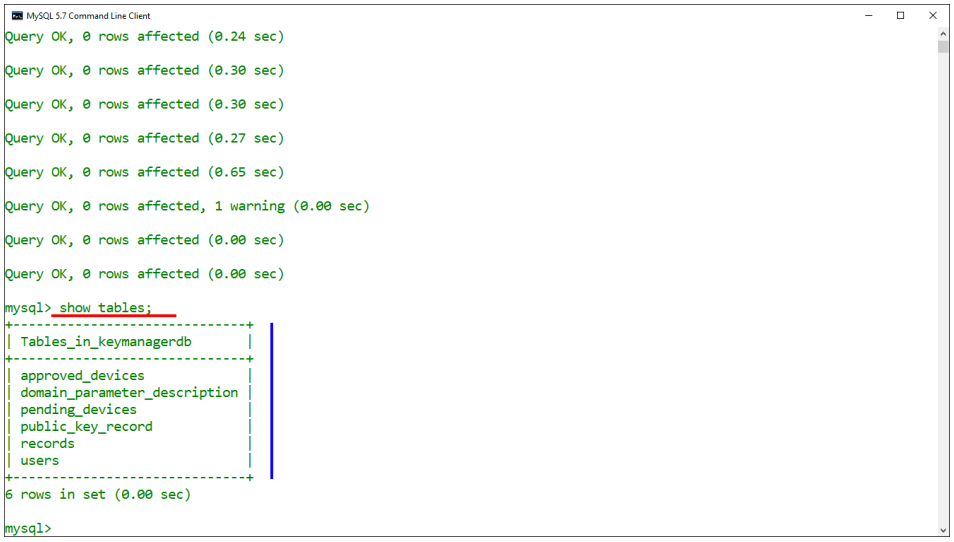
A.2 Vytvoření MySQL databáze pomocí příkazového řádku

V nabídce start najděte a spusťte program `MySQL Command Line Client`. Po jeho spuštění budete vyzváni k zadání hesla, kterým je `root`. V případě úspěšného při-

hlášení se objeví podobný výpis jako je modře označený na obrázku níže. Zadejte následující příkaz:
`source "cesta_k_SQL_skriptu" KeyManagerDB.sql;` !!pozor na konci každého příkazu je středník „;“



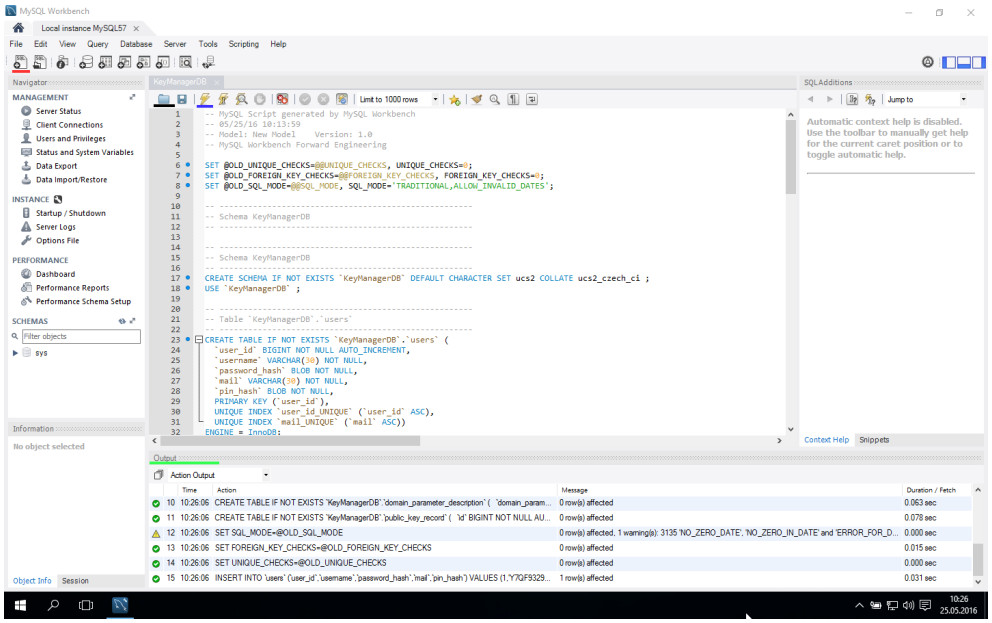
Úspěšné vytvoření databáze lze ověřit příkazem:
`show tables;`
Měli byste vidět stejné údaje jako v označeném modrém poli na obrázku.



A.3 Vytvoření MySQL databáze pomocí MySQL Workbench

Spusťte zmíněný program a klikněte na `Local instance MySQL57`. Budete vyzvání k zadání hesla uživatele root, to je také `root`. Při úspěšném přihlášení uvidíte po-

dobný vzhled jako je na obrázku, s tím rozdílem, že dominující prostřední okno a okno `Output` budou prázdné. Nejdříve klikněte na ikonu obsahující „SQL a plus“



podtrženou červeně, tím se v dominujícím prostředním okně objeví prázdný list. Pokračujte kliknutím na černě podtrženou ikonku, kde vyberte skript `KeyManagerDB.sql`. Po úspěšném načtení se v prostředním okně objeví příkazy skriptu. Pro jeho spuštění klikněte na ikonu s „bleskem“ podtrženou modře. Výsledkem bude podobné pole `Output` podtržené zeleně.

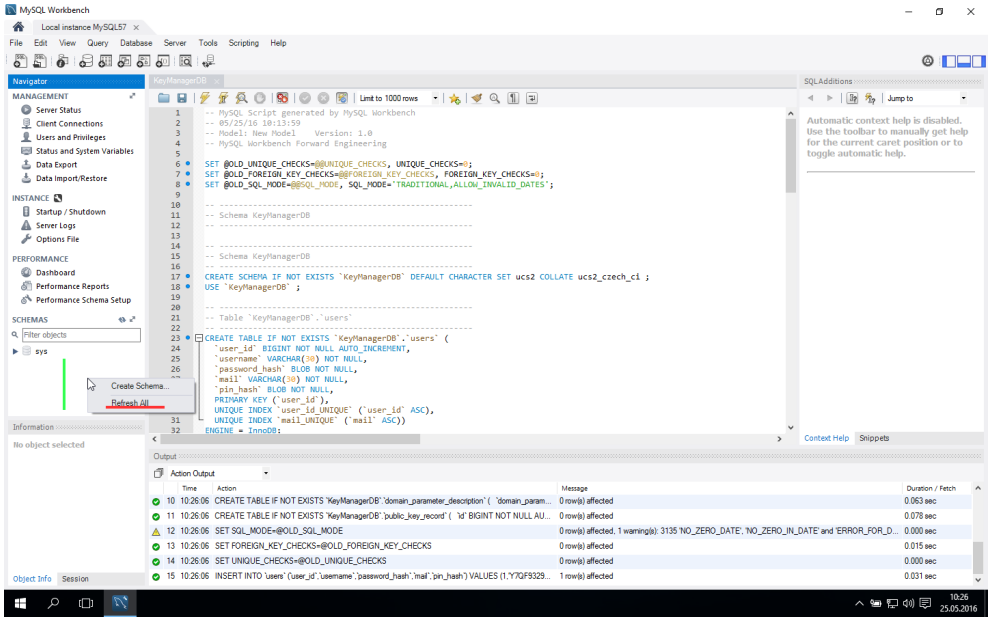
Vytvoření databáze můžete ověřit kliknutím pravého tlačítka myši do zeleně označené oblasti a následným vybráním `Refresh all`. Výsledkem bude zobrazení databáze se jménem `keymanagerdb` obsahující šest tabulek.

A.4 Spuštění manažeru hesel

Pro bezproblémový běh aplikace je důležité mít v počítači nainstalovanou Javu, lze stáhnout zde.

A.4.1 Spuštění serverové části

Spolu se serverovou částí manažeru hesel musí být umístěny soubory:
`identity.jks`, `primary.storage` a `secondary.storage`. Spuštění serverové části je vhodné provést v příkazovém řádku, neboť se serverová část nespustí pouze na pozadí. Toho lze docílit příkazem :
`java -jar KeyManagerServer-0.0.1-jar-with-dependencies.jar`
Musíme vyplnit cestu k souboru. Nebo lze efektivněji otevřít příkazovou řádku



v místě umístění souboru a to podržením klávesy `Shift` a kliknutím pravého tlačítka myši na volný prostor v adresáři. Otevře se nabídka, kde vyberte: `Zde otevřít příkazové okno`.

A.4.2 Postup při jiné konfiguraci MySQL serveru

Máte-li již nainstalovaný MySQL server, který běží na jiném portu než `3306` nebo nastavené jiné heslo uživatele `root` než `root`, tak otevřete JAR se serverovou částí v archivačním programu. Otevřete složku `META-INF` a nastavte hodnoty následujících vlastností v souboru `persistence.xml` dle vaší konfigurace MySQL serveru.
`javax.persistence.jdbc.url`
`javax.persistence.jdbc.user`
`javax.persistence.jdbc.password`

A.5 Spuštění klientské části

Klientská část jde spustit klasickým způsobem, v případě výskytu chyby při spuštění je vhodné použít příkaz:
`java -jar KeyManagerClient-0.0.1-jar-with-dependencies.jar`. Pro simulaci více zařízení je vhodné vytvořit více adresářů, do kterých nakopírujte spustitelný JAR soubor spolu se souborem `trust.jks` a `server.txt`. Soubor `server.txt` slouží pro zadání adresy serveru pro testovací účely.

A.5.1 Postup pro přidání nového zařízení

Vytvořte kopii adresáře s klientskou částí. Spusťte klienta, v menu `Účet` vyberte jedinou aktivní položku `Spárovat tuto instalaci s existujícím účtem`. V zobrazeném okně vyplňte uživatelské jméno, pod které chcete přidat nové zařízení. Heslo a PIN zadaný při registraci, jméno zařízení a zvolte si párovací kód, který si zapamatujte. **!Pozor, program Vás neupozorní na špatně zadané údaje, proto je zkontrolujte!**. Spusťte klienta, ze kterého jste provedli registraci a přihlaste se. Po úspěšném přihlášení vyberte menu `Účet` a zvolte položku `Přidat / odebrat zařízení`. Po opětovném zadání přihlašovacích údajů vyberte tabulku `Zařízení čekající na schválení`. Pokud jste zadali správné údaje při procesu přidání nového zařízení, tak tabulka bude obsahovat jednu položku. Následně ji označte a klikněte na zelenou ikonku s fajfkou. Budete vyzváni k zadání párovacího kódu. Pokud jej zadáte správně, tak budete programem informování a položka z tabulky `Zařízení čekající na schválení` zmizí. Schválení zařízení zkontrolujte v tabulce `Správa schválených zařízení`. Na schváleném zařízení nezapomeňte přidat hlavní klíč pomocí položky `Zadat hlavní klíč` v menu.

B OBSAH PŘILOŽENÉHO CD