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

# UČEBNÍ POMŮCKA PRO VIZUALIZACI ASYMETRICKÉ KRYPTOGRAFIE

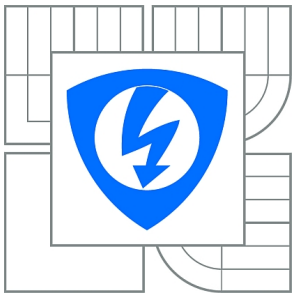
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE JAN KUŽELA
AUTHOR

BRNO 2014

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

# UČEBNÍ POMŮCKA PRO VIZUALIZACI ASYMETRICKÉ KRYPTOGRAFIE

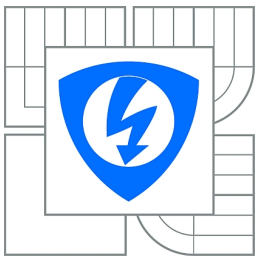
VISUALISATION OF ASYMETRIC CRYPTOGRAPHY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JAN KUŽELA
AUTHOR

VEDOUCÍ PRÁCE — Ing. PETR LEŽÁK
SUPERVISOR

BRNO 2014



**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**

**Fakulta elektrotechniky a komunikačních technologií**

**Ústav telekomunikací**

# Bakalářská práce

bakalářský studijní obor
**Teleinformatika**

***Student:*** Jan Kužela ***ID:*** 147531
***Ročník:*** 3 ***Akademický rok:*** 2013/2014

**NÁZEV TÉMATU:**

## Učební pomůcka pro vizualizaci asymetrické kryptografie

**POKYNY PRO VYPRACOVÁNÍ:**

Vytvořte učební pomůcku do předmětu BIS umožňující vizualizovat průběh Diffie-Hellmanova protokolu, kryptosystému RSA a případně volitelně jiných asymetrických kryptosystémů.

**DOPORUČENÁ LITERATURA:**

[1] BURDA, Karel. Aplikovaná kryptografie. První vydání. Brno: VUTIUM, 2013. ISBN 978-80-214-4612-0.
[2] SMART, Nigel. Cryptography: An Introduction. [online]. 3rd Edition. [cit. 2012-11-27]. Dostupné z: http://www.cs.bris.ac.uk/~nigel/Crypto_Book/

***Termín zadání:*** 10.2.2014 ***Termín odevzdání:*** 4.6.2014

***Vedoucí práce:*** Ing. Petr Ležák
***Konzultanti bakalářské práce:***

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

**UPOZORNĚNÍ:**

## ABSTRAKT

Bakalářská práce se zabývá problematikou kryptografických protokolů, zejména asymetrických. Cílem této práce bylo vytvořit aplikaci pro vizualizaci průběhu matematických operací v asymetrických kryptosystémech. Nejvíce porostoru je věnováno asymetrickým kryptosystémům typu Diffie-Hellman, DSA a RSA. V teoretické části práce je popsána historie a vývoj kryptografie, dále jsou popsány moderní kryptosystémy používané pro ochranu digitálních dat. Praktická část představuje návrh a tvorbu aplikace. Je zde popsána problematika tvorby aplikace a implementace aplikace v programovacím jazyku Java s využitím JavaFX. Výstupem práce je aplikace, která vizualizuje protokoly DH, DSA a RSA. Tyto systémy jsou vizualizovány s hodnotami, které zadá uživatel do vstupních polí, nebo je nechá vygenerovat. Aplikace krok za krokem zobrazuje výpočty, které jsou použité v daném protokolu.

## KLÍČOVÁ SLOVA

Kryptografie, RSA, DSA, Java, Diffie Hellman, digitální podpis

## ABSTRACT

Bachelor thesis is considered in cryptographic protocols, especially asymetrical. The purpose of the thesis is to create an aplication for visualisation of mathematical operations in asymetrical cryptosystems. The emphasis is put on the asymetrical cryptosystems Diffie-Hellman, DSA and RSA. There is a summary of history of cryptography in the theoretical part as well as modern cryphographical systems used for protection of digital data. Practical part shows the design and creation of the application and implementation in the programming language Java and JavaFX. The output is the application, which visualizes protocols DH, DSA and RSA. Systems are visualized with the input data from the user or with randomly generated data. Application shows the calculations made by protocols step by step.

## KEYWORDS

Cryptography, RSA, DSA, Java, Diffie Hellman, digital signature

KUŽELA, Jan *Učební pomůcka pro vizualizaci asymetrické kryptografie*: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2013. 56 s. Vedoucí práce byl prof. Ing. Petr Ležák,

# PROHLÁŠENÍ

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

Brno ..... .....

(podpis autora)

# PODĚKOVÁNÍ

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


Brno ..... .....

(podpis autora)

# OBSAH

# SEZNAM OBRÁZKŮ

# ÚVOD

Komunikace hraje v lidském životě velmi důležitou roli a již od nepaměti je třeba některé informace skrýt před neoprávněnými osobami. Tato skutečnost vedla ke vzniku šifer, pomocí kterých jsou zprávy převedeny do podoby, čitelné jen pro osobu, která má určitou speciální informaci navíc. Tato informace je nazývána šifrovací/dešifrovací klíč. Současně se ale oběvily skupiny lidí, které se snaží získat informaci bez znalosti dešifrovacího klíče. Oběma těmto problémům se věnuje kryptologie, což je soubor kryptografie a kryptoanalýzy. Kryptografie se zabývá utajením informace a kryptologie se zabývá převedením informace zpět do čitelné formy bez dešifrovacího klíče. Kryptografie se dlouhodobě vyvíjela a je hojně využívána dodnes. Z důvodu, že v dnešní době, mají informace vysokou hodnotu a jejich odcizení by mohlo způsbit velké škody, se snažíme zabránit odcizení těchto informací použitím kryptografie. Obecně lze rozdělit kryptografické systémy do dvou velkých skupin. První skupinou je symetrická kryptografie. Do této skupiny patří všechny systémy využívající jedinného tajného klíče k šifrování i dešifrování. Takové systémy mají však jednu slabinu, kterou je kanál pro distribuci klíčů. Postupem času kvůli nedostatkům symetrické kryptografie vznikla asymtrická kryptografie, která je založena na principu dvou klíčů, z nichž jeden slouží k šifrování a druhý k dešifrování. Není již zapotřebí bezpečný kanál pro přenos klíče, protože veřejný klíč je známý všem a slouží jen k šifrování. Každý účastník má tedy svůj pár klíčů a pokud chce komunikovat s druhou stranou, použije její veřejný klíč. Asymetrická šifra nám ale nezaručí, že sa za odesílatele zprávy nevydává někdo jiný. Za účelem zajištění důvěryhodnosti informací vznikly algoritmy digitálního podpisu a další mechanismy jako například infrastruktura veřejných klíčů.

Cílem projektu je navrhnout aplikaci, která bude názorně vizualizivat proces ustanovení klíčů u Diffie-Hellmanova protokolu, šifrování v protokolu RSA a podepisování systémem DSA.

První část je věnována teorii a popisu funkce jednotlivých kryptosystémů. Je popsán matematický aparát, kterého využívají jednotlivé kryptosystémy ke své funkci.

Druhá část se věnuje samotnému návrhu aplikace tedy jakým způsobem bude rozloženo grafické rozhraní aplikace. Je zde popsána použitá technologie a funkcionalita jednotlivých prvků.

# 1 HISTORIE A VÝVOJ KRYPTOGRAFIE

Vývoj kryptografie sahá až do starověku, kdy egypští písaři používali modifikované hieroglyfy. Tyto uchované zápisy z Egypta jsou považovány za první doložené použití psané kryptografie. Hebrejští písaři používali šifru zvanou ATBASH, tato šifra fungovala na principu převrácení abecedy (tedy první znak abecedy byl nahrazen posledním, atd.), nebo se znaky prohazovaly dle určitých schémat. Řekové používali nástroj zvaný skytale, tento nástroj fungoval jako šifrovací i dešifrovací klíč. Byla to dřevěná tyč určitého průměru, na tuto tyč se namotal papyrový proužek. Proužek musel být namotán šroubovicově tak, aby pokryl povrch tyče. Posléze byl na takto vzniklý povrch napsán vzkaz podle osy tyče. Při přenosu vzkazu byl poslán jen samostatný proužek, příjemce si tuto zprávu mohl přečíst jen pokud měl tyč stejného průměru. Tzn. pokud někdo vzkaz odcizil, tak pro něj nebyl čitelný, protože pravděpodobně ani nevěděl co má s takovým proužkem dělat a neměl k dispozici tyč daného průměru. Samozdřejmě by bylo možné tento typ šifry prolomit zkoušením různých průměrů tyčí, ale vzhledem k tomu, že v tomto období neuměl číst každý, byla tato šifra bezpečná. V dobách Viktoriánské Anglie patřilo mezi oblíbené kryptografické metody nahrazování znaků abecedy znaky jinými. A to buď znaky ze stejné abecedy, nebo abecedy odlišné, ale o odpovídajícím počtu znaků. Tato metoda byla velice úspěšná až do doby kdy byl objeven takzvaný statistický útok. Při útoku tohoto typu se spočte průměrný počet znaků v libovolném textu, který je psán ve stejném jazyce jako šifrovaný text. Následně podle četnosti výskytu určitých znaků, jsou tyto znaky přiřazeny k znakům v šifrovaném textu. Postupem vývoje techniky vznikaly různé šifrovací stroje a mechanismy, které prováděly proces šifrování a dešifrování v reálném čase. Příkladem těchto šifrovacích strojů je přístroj Enigma, který byl vynalezen v Německu během 2. světové války. V tomto období došlo k velkému pokroku v oblasti kryptografie. K dalšímu velkému rozvoji kryptografie došlo během studené války. V této době byla již známa spousta technik, jelikož doposud existující kryptosystémy byly založeny na symetrické kryptografii, měli jednu velkou slabinu kterou byla bezpečná distribuce klíče. Tato slabina byla natolik významná že vedla ke vzniku asymetrické kryptografie, která tento problém vyřešila. V dnešní době je pro většinu přenosů informací využíváno digitální techniky, i při těchto přenosech, pokud má být informace utajena se šifruje za pomoci symetrické či asymetrické kryptografie. Používají se oba způsoby šifrování, protože každý typ šifrování má jisté výhody a nevýhody. Kryptosystém je volen na základě v jakém odvětví je použit. Podle toho jak velká je potřeba míra zabezpečení a zároveň jaké máme k dispozici prostředky. Více zde: [4] a [13], [14].

# 2 ZÁKLADNÍ POJMY

V této kapitole se seznámíme se základními pojmy používanými v kryptografii, protože bez znalosti jejich významu by mohlo dojít k neúplnému pochopení dalšího textu.

Informace je abstraktní zobrazení skutečnosti, nehmotná informace je obrazem skutečného světa, zobrazuje stav jak hmotných tak i nehmotných věcí. Informace je přenášena pomocí nosičů informací.

Nosič informace je médium, pomocí kterého probíhá přenos informací. Nosiče informace dělíme na nosiče paměťové a přenosové. Paměťové nosiče (např. magnetická páska, pevný disk, cd, dvd, polovodičové paměti), přenášejí informaci časem. Přenosové nosiče (Počítačové či telefonní sítě) přenášejí informaci v prostoru. Ideální přenosový kanál má minimální chybovost a útlum. U reálných kanálů se toho snažíme docílit, ale přesto má každý reálný přenosový kanál určitou chybovost a útlum. Nosič informace nabývá několika stavů dané veličiny. Nosič, který nabývá nekonečného počtu stavů dané veličiny, nazýváme analogovým nosičem. Analogový nosič přenáší analogový signál. Naopak nosič, který nabývá konečného počtu (minimálně dvou) stavů dané veličiny je takzvaný digitální nosič. Takovýto nosič má počet stavů daný určitou abecedou. Digitální nosič přenáší digitální signál. Z fyzického hlediska může být stejný jako analogový nosič. V dnešní době se majoritně využívají digitální nosiče. Z toho důvodu, že digitální signál nepodléhá degradaci, protože díky tomu, že signál nabývá jen konečného počtu hodnot, je možno využít protichybových kódů k detekci a opravě chyb. Při přenosu na velké vzdálenosti je možné rekonstruovat signál degradovaný útlumem vedení, zařazením vhodného opakovače do přenosové cesty. Zatímco rekonstrukce řušeného analogového signálu je z důvodu nekonečného počtu hodnot téměř nemožná. A v případě, že bychom potřebovali zesilovat analogový signál, který degradoval útlumem na vedení zesílíme původní šum a zaneseme do signálu další šum, který vznikne v zesilovači. Analogový signál také zabírá zbytečně velkou šířku pásma, takže pro nás má přenosový kanál mnohem nižší kapacitu. Tzn. na fyzicky stejném přenosovém médiu, za stejný čas, přeneseme méně informací při použití analogového signálu, protože pro analogový signál nelze použít kódy pro redukci redundance[1] a irelevance. [2] Na přenosovém médiu je veličinou zastupující informaci nejčastěji elektrický proud, napětí, případně intenzita světla proměnná v čase. U paměťových nosičů jsou to změny magnetického pole, elektrického náboje či vodivosti proměnné v prostoru. Skutečnost že je

[1] redundance - nadbytečnost (stejnou informaci lze vyjádřit i menším počtem bitů)

[2] irelevance - zbytečnost

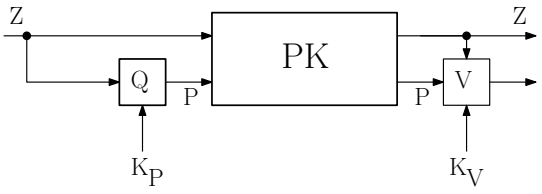
veličina proměnná v čase nazveme časová posloupnost (signál), veličinu proměnnou v prostoru nazýváme prostorová posloupnost (záznam). V kryptografii ale není podstatné, jaká je forma zprávy z fyzického hlediska, proto budeme zprávu brát jako posloupnost celých čísel, ve které ja zakódována určitá informace. K přenosovým kanálům má ve většině případů přístup více účastníků, mezi těmito účastníky může být i útočník, který může být pasivní nebo aktivní. Pasivní útočník odposlouchává přenos, aby využil získané informace ve svůj prospěch. Aktivní útočník se snaží modofokovat přenášenou zprávu tak, aby poškodil komunikující entity.[1]

Šifra má za úkol přeměnit zprávu tak, aby byla nečitelná pro neoprávněné příjemce, ale aby zůstala čitelná pro příjemce, pro kterého je určena. Tento příjemce má speciální informaci navíc (Klíč), kterým zprávu převede zpět do čitelné podoby. Šifra řeší utajení zprávy, ale nezaručí nám autentičnost přenášené zprávy. Například u asymetrických kryptosystémů, může kdokoliv použít veřejný klíč k zašifrování zprávy a příjemce již nemůže zjistit, jestli se za odesílatele nevydává někdo jiný.

Digitální podpis je systém sloužící k autentizaci zpráv a jednotlivých komunikujících entit. Jinými slovy ověřuje pravost elektronických dat. Je zde využito algoritmů asymetrické kryptografie, s tou výjimkou, že narozdíl od šifrování se při podepisování u systému RSA podepisuje soukromým klíčem. Veřejným klíčem se ověřuje autentičnost. Zprávu zašifrovanou soukromým klíčem může dešifrovat kdokoliv, kdo zná veřejný klíč odesílatele, ale nikdo ji nemůže zfalšovat, protože nezná soukromý klíč odesílatele. Odesílatel, zašifruje zprávu $Z$, takto zašifrovanou zprávu pak připojí k odesílanému dokumentu (podepíše se). Následně tento dokument odešle. Příjemce připojenou zprávu dešifruje veřejným klíčem odesílatele. Pokud je tato zpráva neporušená je dokument považován za pravý. Dále lze při podepisování využít trochu jinou techniku s využitím hashovací funkce. A to tak, že odesílatel z odesílané zprávy nejprve vytvoří pomocí hashovací funkce hash (otisk), který zašifruje soukromým klíčem a přiloží jej k odesílanému dokumentu. Příjemce pak dešifruje podpis, a vytvoří ze zprávy hash. Pokud je takto vzniklý hash stejný jako podpis po dešifrování je tato zpráva autentická. Využití hashovací hunkce má tu výhodu, že hash je mnohonásobně menší než odesílaná zpráva. Zašifrování hashe je rychlejší než zašifrování celé zprávy. Více informací lze nalézt [4]

Kde Z je zpráva, $K_p$ je soukromý klíč, $K_v$ je veřejný klíč, P je podpis. V modulu Q probíhá zašifrování zprávy soukromým klíčem, v modulu V na straně příjemce je pak ověřena pravost porovnáním podpisu zprávy s přijatou zprávou.

Šifrování je vědní obor zabývající se technikou utajení informací, zajištění autentičnosti zpráv a komunikuících entit (např. digitální podpis), tím způsobem že




Obr. 2.1: Znázornění kryptografického systému pro zajištění autentičnosti zpráv.

přenášenou informaci převede do takové podoby, aby byla čitelná jen se speciální znalostí, kterou má příjemce. Výstupem šifrování je šifra.

Kryptografie se zabývá návrhem matematických ochran. Výstupem kryptografie je kryptografický systém. Rozlišujeme symetrickou a asymetrickou kryptografii.

Kryptografickým systémem budeme rozumět systém k zajištění důvěrnosti a autentičnosti zpráv. Výstupem kryptografického systému je tzv. kryptogram, jedná se o zašifrovanou zprávu, která má určitou dobu rezistence. Doba rezistence udává čas, který bude potřebovat útočník k prolomení šifry a přečtení zprávy bez dešifrovacího klíče, je volena individuálně v závislosti na aplikaci (Některé informace stačí udržet v tajnosti řádově několik hodin, jiné informace musí být v tajnosti několik desetiletí).

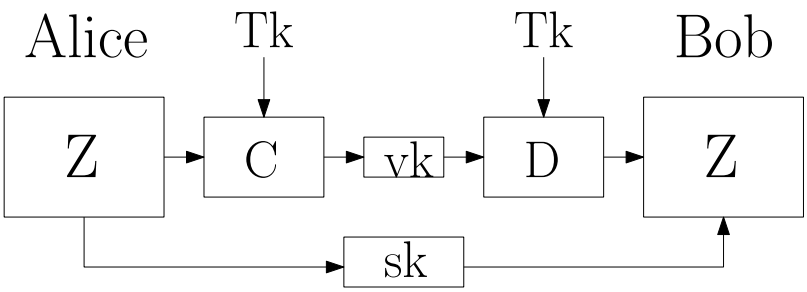
Kryptografická ochrana je založená na nemožnosti řešení určitého matematického problému v reálném čase.

Kryptoanalýza je věda zabývající se rozluštěním zašifrované zprávy bez znalosti klíče. Soubor kryptografie a kryptoanalýzy se nazývá kryptologie. Tyto informace jsem nabyl zde:[1] a zde:[7].

# 3 ROZBOR PROBLEMATIKY

## 3.1 Symetrická kryptografie

Symetrická kryptografie je soubor kryptografických systémů, který využíva jedinný klíč - Tajný klíč *Tk*. Tento klíč slouží jak pro šifrování tak i pro dešifrování $K_s = K_d$[1], což je důvod k tomu, že musí každá strana klíč znát a musí jej držet v tajnosti. V jistých případech nemusí být klíče pro šifrování a dešifrování úplně stejné, stačí pouze když lze jeden klíč od toho druhého snadno odvodit. Tyto systémy mají však jednu velkou slabinu, a to takovou, že při odcizení klíče je útočník schopen jednoduše dešifrovat zprávu, dále (pokud je využíváno dvou klíčů), může útočník snadno vypočítat dešifrovací klíč z šifrovacího. Další nevýhodou je, že u tohoto typu kryptosystémů musí existovat samostatný zabezpečený kanál pro přenos klíčů. Symetrická kryptografie není založena na žádném matematickém problému. Proto je nutné dodržet podmínky při volbě parametrů potřebných pro šifrování. Je třeba používat velké klíče řádově 80 bitů a větší. Symetrická kryptografie je díky své rychlosti při výpočtu klíče a šifrování nejrozšířenější metodou pro přenos tajných zpráv na nedůvěrném kanále.



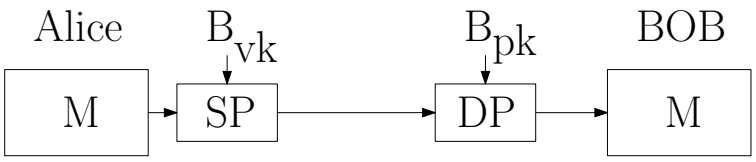

Obr. 3.1: Znázornění symetrického kryptosystému.

Kde Z je zpráva, vk je veřejný kanál, sk soukromý kanál. V modulu C probíhá šifrování a v modulu D probíhá dešifrování. Soukromý kanál musí být bezpečný veřejně nepřístupný komunikační kanál. Prostřednictvím tohoto kanálu probíhá výměna klíčů, ve většině případů bývá tento kanál vytvořen některým z asymetrických kryptosystémů, protože vytvářet samostatný fyzický kanál jen pro přenos klíčů by nebylo praktické. Fyzické řešení tohoto kanálu bývá zpravidla pomocí takzvaných tokenů (speciální USB klíč). Tyto tokeny mají jen oprávnění uživatelé, samotné tokeny mohou být ještě chráněny heslem.

[1] $K_s$ - šifrovací klíč $K_d$ - dešifrovací klíč

## 3.2 Asymetrická kryptografie

Asymetrická kryptografie je soubor kryptosystémů založený na principu tzv. jednocestné funkce se zadními vrátky (Tapdoor function) $y = f(x)$. Klíčovou funkcí tohoto matematického problému je, že je řešitelná pouze v jednom směru, pokud nemáme k dizpozici jistou informaci takzvané zadní vrátka (trapdoor). Výpočet hodnoty proměnné $y$ ze vstupní proměnné $x$ je poměrně jednoduchý, avšak výpočet $x$ z výstupní proměnné $y$ je bez zadních vrátek velmi časově i výpočtově náročný proces. Z toho plyne, že je tento problém neřešitelný v reálném čase. Asymetrická kryptografie využívá sadu dvou klíčů. Veřejný klíč $K_{pub}$, jehož parametry jsou veřejně známé a soukromý klíč $K_{priv}$, jehož hodnotu zná jen generující entita nebo její majitel. Veřejný klíč se používá jako šifrovací klíč při utajování informace, nebo jako dešifrovací klíč pro ověření digitálního podpisu. Využívá se zde jednocestná funkce z důvodu aby byla zajištěna nemožnost vygenerování privátního klíče z veřejného. Komunikace v asymetrické kryprografii funguje následovně: Alice a Bob si vygenerují sadu klíčů, veřejné klíče pak uveřejní. Pokud chce alice poslat Bobovi zprávu, tak pro zašifrování použije jeho veřejný klíč. Pokud chce bob poslat zprávu Alici rovněž ji zašifruje jejím veřejným klíčem. V žádném případě nesmí žádná z komunikujících entit pro zašifrování použít svůj soukromý klíč. V takovém případě by přenášená zpráva nebyla chráněna, protože by ji dokázal dešifrovat každý, kdo zná veřejný klíč a celý předchozí proces by byl naprosto zbytečný. Asymetrická kryptografie byla vyvinuta z důvodů slabin kryptografických systémů založených na symetrické kryptografii. Mezi hlvní nevýhody systémů symetrické kryptografie patří skutečnost, že je vyžadován samostatný bezpečný kanál pro přenos klíčů. Jelikož u asymetrické kryptografie je jedením klíčem šifrováno a druhý slouží k dešifrování, je zde tento problém jen částečně eliminován. Je zde potřeba ještě řešit autentičnost klíčů. Tedy zda veřejný klíč skutečně patří entitě, se kterou si myslíme že komunikujeme. Tuto problematiku řeší infrastruktura veřejných klíčů. Viz. podkapitola 3.3. Asymetrická kryptografie je využívána k šifrování zpráv, digitálnímu podpisu, autentizaci jednotlivých entit, k přenosu klíčů pro symetrickou kryptografii, kdy je vytvořen bezpečný kanál pro přenos klíče. Dále se zbytek komunikace šifruje pomocí symetrické kryptografie. Sestavení klíče pro symetrickou kryptografii po nezabezpečeném kanálu řeší např. Diffie-Hellmanův protokol.




Obr. 3.2: Znázornění asymetrického šifrování.

Kde M je zpráva, SP je šifrovací proces, DP je dešifrovací proces $B_{vk}$ je Bobův veřejný klíč a $B_{pk}$ je Bobův privátní klíč.

## 3.3 Infrastruktura veřejných klíčů

Infrastruktura veřejných klíčů(PKI Public Key Infrastructure) je systém digitálních certifikátů, certifikačních autorit(CA) a registračních autorit. Tento systém slouží k ověřování platnosti všech stran zúčastněných v jisté elektronické transakci s využitím asymetrické kryptografie. Infrastruktura veřejných klíčů zajišťuje integritu, důvěrnost, autenticitu a zodpovědnost.
Digitální certifikát je dokument vydaný důvěryhodnou třetí stranou(autoritou). Tento dokument obsahuje tvrzení, že k jisté entitě patřící veřejný klíč má určitou číselnou hodnotu. Certifikáty zajišťují důvěru, že veřejný klíč patří skutečně dané entitě.
Digitální certifikáty obsahují následující informace:

- jméno certifikační autority vydávajíci certifikát
- veřejný klíč
- dobu vypršení platnosti
- pořadové číslo
- informaci o tom jakým způsobem má být klíč používán
- digitální podpis vydavatele certifikátu

Certifikáty nesmí být padělatelné, musí být získávány bezpečnou cestou. Vytváření certifikátu musí probíhat bezpečným způsobem tak, aby byla zaručena odolnost vůči potenciálním útokům. Přesnější popis infrastruktury veřejných klíčů je nad rámec tohoto textu. Problematika je podrobně popsána v [2] a [1].

# 4 METEMATICKÉ PROBLÉMY V ASYMETRICKÝCH KRYPTOSYSTÉMECH

## 4.1 Problém výpočtu diskrétního logaritmu

Problém výpočtu diskrétního logaritmu (discrete logarithm - DL) je další z problémů, se kterými počítají různé kryptosystémy asymetrické kryptografie.

$$y = f(x) = g^x \bmod p \tag{4.1}$$

Problém diskrétního logaritmu je popsán v rovnici 4.1 kde $p$ je velké prvočíslo a $g$ je známé přirozené číslo. V tomto problému je relativně snadné vypočítat hodnotu $y$ pro dané $x$. Ale pro dostatečně velké $p$ je prakticky nemožné s dnešní technikou z daného čísla $y$ určit hodnotu $x$ v rozumném čase. Parametry $y$, $g$, $p$ jsou zveřejněny jen parametr $x$ je soukromý. Kryptosystémy založené na diskrétním logaritmu se nepoužívají k šifrování z toho důvodu, že kryptogram má dvojnásobnou délku oproti původní zprávě. To znamená že po zašifrování zprávy je potřeba přenést dvojnásobný objem oproti zprávě samotné. Kryptosystémy založené na problému diskrétního logaritmu mají své využití v systémech pro digitální podpis. Dále se používají k ustanovení klíče (tzv. Diffie Hellmanův protokol). Problém diskrétního logaritmu je považován za neřešitelný v reálném čase pro dostatečně velká celá čísla. Proto jsou kryptosystémy založene na diskrétním logaritmu považovány za velmi bezpečné při správné délce klíče. Problém výpočtu diskrétního logaritmu je další z problémů, se kterými počítají různé kryptosystémy asymetrické kryptografie. Funguje na principu rozložení složeného čísla na mocninu jiného čísla $a = g^x$. Nalezená mocnina $x$, kterou lze zapsat jako $x = \log_g a$, se pak nazývá diskrétní logaritmus. Na tomto problému pracují systémy typu ElGamel, Diffie-Hellmanův protokol a další. Obecně lze napsat, že pro $g$ $z$ $G$ je třeba najít číslo $x$ takové, že $g^x \equiv a \bmod n$, kde $n$ je řád skupiny $G$ a $a$ je známý parametr. Při počítání diskrétního logaritmu je třeba použít vhodnou metodu, protože každá metoda počítá hodnotu $x$ jinak. Při náročném problému se může nevhodně zvolená metoda vypočítat nesprávný výsledek nebo může výpočet trvat delší čas než je nutné. Problém diskrétního logaritmu je považován za neřešitelný v reálném čase pro dostatečně velká celá čísla. Proto jsou kryptosystémy založene na diskrétním logaritmu považovány za velmi bezpečné při správné délce klíče.

### 4.1.1 Zkušební dělení

Jakmile je určeno, že faktorizované číslo je složeno z prvočísel, je doporučeno použít algoritmus zkušebního dělení. Tato metoda je efektivní pro celá čísla, pokud se číslo

$n$ skládá z malých prvočísel $p$.

## 4.2 Fermatův test prvočíselnosti

Fermatův test prvočíselnosti dokáže určit, zda je dané číslo prvočíslem, nebo číslem složeným. Tento test využívá malé fermatovy věty a funguje s určitou pravděpodobností. Jelikož je tento test pravděpodobnostní, je třeba použít parametr $i$, díky kterému se chráníme před a Fermatovými lháři. Ale stále zde zůstává problém Carmichaelových čísel, před kterými se nedá tak jednoduše chránit, protože opakování testu nám nepomůže.

Fermatův test prvočíselnosti se v praxi příliš nevyužívá právě z důvodu existence Carmichaelových čísel. Proto je v praxi využito Rabin-Millerova nebo Lehmannova testu, které tímto nedostatkem netrpí. Více informací naleznete v[5].

### 4.2.1 Malá Fermatova věta

Malá Fermatova věta říká, že pro každé prvočíslo $p$, které je nesoudělné s číslem $a$ platí následující tvrzení:

$$a^{p-1} \equiv 1(\mathrm{mod}p) \tag{4.2}$$

$$a^{p} \equiv a(\mathrm{mod}p) \tag{4.3}$$

Informace jsem čerpal z [5].

#### Složená čísla a prvočísla

Prvočíslo je beze zbytku dělitelné, jen číslem 1 a nebo sebou samým. Tzn. číslo 1 není prvočíslo. Čísla která jsou různá od jedné a nejsou prvočísly jsou nazývány čísly složenými.

Všechna složená čísla lze vyjádřit jako součin prvočísel. Rozklad složeného čísla na prvočinitele se nazývá faktorizace složených čísel.

#### Fermatův lhář

Pokud fermatova věta pro číslo a bude platit, přestože p je složené číslo. Pak je číslo a Fermatův lhář vzhledem k číslu p.

**Carmichaelovo číslo**

Složená čísla, pro která platí $GCD(a, p) = 1$ [1] při libovolné bázi a se nazývyjí Carmichaelova čísla. Těchto čísel je nekonečně mnoho například čísla (1105, 2465, 2821 atd.).

## 4.3 Eratosthenovo síto

Eratosthenovo síto je algoritmus, který slouží k nalezení všech prvočísel, která jsou menší než zadaná mez.

1. V prvním kroku se vytvoří seznam čísel v daném rozsahu (2 - Zadané maximum)
2. Poté se vyjme první číslo ze seznamu. (2 je první prvočíslo)
3. Ze seznamu jsou odebrány všechny násobky tohoto čísla, protože z definice již nemohou být prvočísly.
4. Tento postup je opakován až do okamžiku, než je ze seznamu odebráno poslední prvočíslo, nebo pokud je číslo vyšší než odmocnina nejvyššího čísla prohlášeno za prvočíslo - v tomto případě už musí být všechna čísla prvočísly. [10]

[1] *GCD(Greatest common divisor) - Největší společný dělitel, někdy také česky označovaný NSD.

# 5 ASYMETRICKÉ KRYPTOSYSTÉMY

## 5.1 Diffie-Hellmanův protokol

Diffie-Hellmanův protokol zkráceně DH, patří mezi protokoly asymetrické kryptografie. DH protokol řeší problematiku ustanovení tajného klíče $T_k$ po nazabezpečeném kanálu. Komunikující strany si veřejně vymění dvě čísla, pomocí kterých sestrojí Tajný klíč, který je totožný pro obě strany. Tento klíč slouží jak k šifrování tak k dešifrování zprávy. Následující komunikace probíhá pomocí některého ze symetrických kryptosystémů.
Účastníky komunikace si označíme jmény Alice a Bob. Alice si zvolí a zveřejní dostatečně velké prvočíslo $p$ a generátor $g$, zde platí $1 \leq g \leq n-2$. Číslo $p$ musí být takové, aby hodnota $p-1$ měla alespoň jeden velký prvočíselný dělitel nebo aby platilo $p-1 = 2\text{n}$, kde $n$ je prvočíslo. Hodnoty $p$ a $g$ se ustanovují před započetím komunikace a ustanovováním klíčů, mohou mít i dlouhodobě stejnou hodnotu. Tato čísla jsou veřejně dostupná. Alice si vygeneruje svoje tajné číslo $a$ pro které platí $1 \leq a \leq n-2$ vypočte:

$$A = g^a \bmod p. \tag{5.1}$$

Bob si zvolí číslo b, platí $1 \leq b \leq n-2$:

$$B = g^b \bmod p. \tag{5.2}$$

Nyní může Alice spočítat:

$$S = B^a \bmod p. \tag{5.3}$$

Bob spočítá:

$$S' = A^b \bmod p. \tag{5.4}$$

V této chvíli mají Alice a Bob spočtena čísla S a S', tyto čísla tzv. sdílený klíč, jsou totožné a slouží k šifrování a dešifrování zprávy.

Pro lepší pochopení přidávám následující příklad s konkrétními hodnotami.

1. Máme dáno prvočíslo $p = 11$ a generátor $g = 6$
2. Alice si zvolí číslo $a = 8$ a vypočte $A = 6^8 \bmod 11 = 4$. Vypočtenou hodnotu (4) pošle Bobovi.

3. Bob si zvolí číslo $b = 3$ a vypočte $B = 6^3 \bmod 11 = 7$. Vypočtenou hodnotu (7) pošle Alici.

4. Alice si spočte $S = 7^8 \bmod 11 = 9$

5. Bob si spočte $S' = 4^3 \bmod 11 = 9$

6. Nyní mají oba účastníci stajnou hodnotu klíče.

Jak můžeme vidět Diffie-Helmanův protokol je založen na problému Diskrétního logartimu který je popsán v kapitole 4.1
Záškodník pasivním odposloucháváním komunikačního kanálu nemůže zjistit sdílený klíč, protože není schopen zjistit čísla $a$ a $b$ z odposlechnutých hodnot A a B, k tomu aby je vypočetl by musel vyřešit problém diskrétního logaritmu. DH protokol nám však nezaručí autentizaci jednotlivých entit, z toho důvodu jej lze napadnout útokem typu „man in the middle“. Kdy se útočník vydává za druhou stranu, Alice a Bob nemají tušení, že nekomunikují přímo spolu, ale prostřednictvím útočníka. Takže Alice a Bob mají domění že mají sestaven tajný klíč, ale ve skutečnosti mají klíč sestaven s útočníkem, který si pak zprávu dešifruje, přečte a případně i upraví poté ji znovu zašifruje a odešle druhé straně. Hlouběji je tato problematika popsána v[1]

## 5.2 RSA

RSA je jedním z nejrozšířeňejších kryptografických algoritmů pracujících s veřejným klíčem. Tento algoritmus byl vyvinut v roce 1977 skupinou složenou ze tří matematických expertů Rivesta, Shamira a Adlemana. Bezpečnost tohoto algoritmu je založena na problému faktorizace velkého čísla na prvočísla a tzv. RSA problému. Systém RSA je používán pro šifrování zpráv, vytvoření bezpečného kanálu pro přenos klíčů symetrické kryptografie dále poskytuje autentizaci dokumentů pomocí digitálního podpisu.
V následujícím textu bude popsáno jakým způsobem pracuje algoritmus pro generování soukromého a veřejného klíče, tento děj si rozdělíme do několika kroků:

1. Příjemce náhodně vygeneruje dvě velké prvočísla $p$ a $q$.

2. Nyní vypočte součin čísel $p$ a $q$: $n = pq$

3. V tomto kroku je spočtena hodnota eulerovy funkce $r = (p-1)(q-1)$.

4. Zvolí se celé číslo $e$ menší než $r$ tak, aby bylo s číslem $r$ nesoudělné. Typicky je $e$ malé prvočíslo například $e = 2^{16} - 1$ a nesoudělné s číslem $r$. Toto číslo je veřejný exponent. Nesoudělnost může být ověřena pomocí euklidova algoritmu.

5. Pomocí rozšířeného Euklidova algoritmu se vypočte soukromý exponent $d$ podle: 5.5

$$d = e^{-1} \bmod r \tag{5.5}$$

6. Veřejným klíčem je zde dvojice čísel $(e, n)$, kde $e$ se označuje jako veřejný nebo šifrovací exponent a číslo $n$ se označuje jako modul. Soukromým klíčem je dvojice čísel $(d, n)$, kde $d$ je označováno jako dešifrovací či soukromý exponent a $n$ je opět modul protože se jedná o stejné číslo. Více o principu fungování RSA naleznete v[1], [7] a [8].

Samotná komunikace po ustanovení klíčů probíhá následovně. Odesílatel si nejprve zjistí veřejný klíč příjemce, kterému chce odeslat zprávu $Z$. Tuto zprávu musí nejprve převést na číselný formát, k tomu bývá s výhodou využito ascii tabulky. V ascii tabulce ja každý znak zastoupen číslem od 0 po 127. Zpráva je následovně rozdělena na bloky stejné dálky. Délka každého takto vzniknutého bloku $z_i$ musí být menší než $n$. Matematicky to můžeme zapsat následovně $z_i < n$. Potom je vybrán každý $i$-tý blok a takto vybraný blok je zašifrován. Šifrování probíhá pomocí vzorce 5.6

$$c_i \equiv z_i^e \bmod n. \tag{5.6}$$

Bloky $c_i$ jsou následovně spojeny do jednoho celku jenž nazýváme kryptogram C. Příjemce příjme zprávu C a dešifruje ji využitím svého soukromého klíče čímž získá původní zprávu. Dešifrování probíhá podle vzorce 5.7, průběh dešifrování je obdobný šifrovacímu procesu.

$$z_i \equiv c_i^d \bmod n. \tag{5.7}$$

Pro lepší pochopení algoritmu pro vytvoření dvojice soukromého a veřejného klíče si zde uvedeme příklad s konkrétníma hodnotama.

1. Bob si zvolí prvočísla $p = 7$ a $q = 13$.
2. Modulus $n = 7 \cdot 13 = 91$.
3. $r = (7 - 1)(13 - 1) = 72$.
4. Bob zvolí hodnotu $e = 5$. protože číslo 5 je nesoudělné s číslem $r = 72$.
5. Bob nyní spočítá $d = 5^{-1} \bmod 72 = 29$.
6. $K_{pub} = (5, 72)$, $K_{priv} = (29, 72)$.

Nyní máme ustanovenu dvojici klíčů, takže můžeme přejít k samotnému šifrování.

1. Alice chce poslat zprávu, zpráva je libovolným číslem z intervalu $< 0, n-1 >$ zvolíme tedy číslo 59
2. Alice získá Bobův veřejný klíč $K_{pub} = (e, n)$, pro náš příklad: $K_{pub} = (5, 72)$
3. Alice vytvoří kryptogram $C$, Pro který platí: $C = Z^e \bmod n$, kde Z je zpráva. Takže $C = 59^5 \bmod 72 = 11$.
4. Kryptogram $C$ je odeslán Bobovi.
5. Pro dešifrování zprávy platí: $Z = C^d \bmod n$
6. Bob tedy zprávu dešifruje: $Z = 11^{29} \bmod 72 = 59$

Nyní si můžeme všimnout, že pokud budeme vícekrát přenášet stejnou zprávu s použitím stejných klíčů, bude při každém přenosu tato zpráva zašifrována jako stejný kryptogram. Proto se používá takzvaný padding, více v kapitole: 5.2.1. Více o paddingových schématech lze nalézt [1].

### 5.2.1 Padding v RSA

V praxi se užívá při šifrování RSA také padding, protože RSA šifra je náchylná na prolomení se znalostí části otevřeného textu. Proto se zprávy nešifrují přímo, ale je využito jisté paddingové schéma, které vnáší do zprávy určité náhodné hodnoty a tím zabraňbuje tomuto typu útoku. OAEP-Optimal Asymmetric Encryption Padding je paddingové schéma později používané spolu s RSA.

## 5.3 DSA

DSA je zkratka Digital Signature Algorithm což v překladu neznamená nic jiného než algoritmus digitálního podpisu. Někdy se označuje jako DSS(Digital Signature System) což standart americké vlády pro digitální podpis, DSS vznikl z DSA roku 1991 a je téměř totožný s DSA. DSS se od DSA liší jen omezením délky klíče na 1024 bitů. V dnešní době se ale již využívá klíčů délek 2048 bitů pro podpisování běžných uživatelů a až 4096 bitů pro certifikační autority. Algoritmus DSA je založen na problému diskrétního logaritmu, a je určen ke generování a ověřování digitálního podpisu. Takže se jeho bezpečnost odvíjí od toho, že nelze efektivně řešit diskrétní logaritmus. Stále však není prokázáno že je tento problém dostatečnou podmínkou pro bezpečnost tohoto algoritmu. Dále bude uvedeno jak funguje mechanizmus DSA algoritmu. Před samotným podepisováním je třeba ustanovit dvojici klíčů. Veřejného a soukromého. A to probíhá následujícím způsobem:[4]

1. Zvolí se prvočíslo $n$, pro které platí: $2^{511+64t} < n < 2^{512+64t} (0 < t < 24)$
2. Zvolí se prvočíslo $q$, toto číslo musí být celočíselným dělitelem $n-1$ (zpravidla má toto číslo délku 160-512 bitů).
3. Nyní je třeba zvolit parametr $h$ tak, aby $h < n-1$ a zároveň $h^{\frac{n-1}{q}} \bmod n > 1$.
4. Následně je třeba vypočítat generátor $b$ podle vztahu 5.8

$$b = h^{\frac{n-1}{q}} \bmod n \tag{5.8}$$

5. Je třeba zvolit náhodné čislo k, tak aby splňovalo podmínku: $1 < k < q$
6. Vypočte se Y podle 5.9.

$$y = b^k \bmod n \tag{5.9}$$

7. Soukromým klíčem je číslo $k$, veřejný klíč je tvořen $(n, q, b, y)$

Nyní když máme ustanovený veřejný a soukromý klíč, můžeme přejít k vytvoření samostatného podpisu, které probíhá tímto způsobem:

1. Prvním krokem je volba celého čísla x, pro které platí podmínka $0 < x < n$
2. Vypočte se $r$ podle vztahu 5.10

$$r = (b^x \bmod n) \bmod q \tag{5.10}$$

3. Vypočte se $s$ podle 5.11

$$s = x^{-1}[h(M) + kr] \bmod q \tag{5.11}$$

4. Digitálním podpisem pro zprávu M je dvojice čísel $(r, s)$

K vytvoření otisku (hashe) zprávy používá DSA algoritmus SHA - 1[1] , který vygeneruje sekvenci dat o délce 160 bitů, tato sekvence je pro potřeby výpočtu převedena na číslo. Ověření pravosti podpisu probíhá pole následujícího postupu:[4]

1. Uživatel získá veřejný klíč $(n, q, b, y)$ z ověřeného zdroje.

2. Přejde k ověřování podmínek: $0 < r < q$ a $0 < s < q$. Pokud jedna s těchto podmínek není splněna, je tento podpis odmítnut.

3. Vypočte se $w$ podle 5.12

$$w = s^{-1} \bmod q \tag{5.12}$$

[1]SHA-1 (Secure Hash Standard)

4. Vypočte $u_1$ a $u_2$ podle 5.135.14

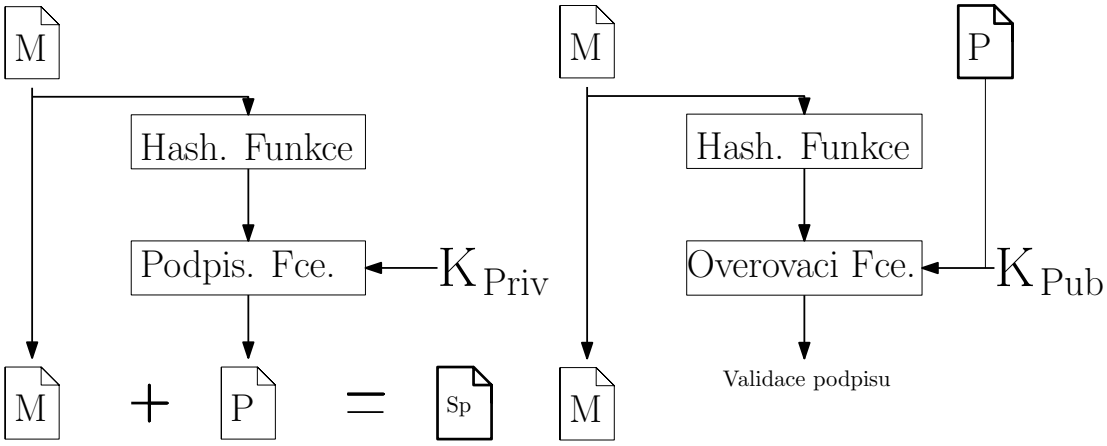
$$u_1 = w \cdot h(M) \bmod q \tag{5.13}$$

$$u_2 = r \cdot w \bmod q \tag{5.14}$$

5. Vypočte hodnotu $v$ podle 5.15

$$v = (b^{u_1} y^{u_2} \bmod n) \bmod q \tag{5.15}$$

6. Podpis je přijat v případě, že je splněna podmínka v=r, pokud tato podmínka není splněna je podpis odmítnut.




(a) Podepisování dokumentu. (b) Ověřování pravosti dokumentu

Obr. 5.1: Schéma pro znázornění průběhu podepisování a ověřování pravosti dokumentu.

Na obrázku a je znázorněn průběh podepisování zprávy, kde M je zpráva, $K_{Priv}$ je soukromý klíč, P je podpis a SP je již podepsaná zpráva.
Na obrázku b je následně zobrazen průběh ověření pravosti, kde $K_{Pub}$ je veřejný klíč odesílatele, pomocí kterého je dešifrován podpis. Hashovací hunkce vytvoří hash ze zprávy M. Tento hash je následně porovnán s dešifrovaným podpisem, pokud hash zprávy M, je stejný jako dešifrovaný podpis, je podpis přijat. V opačném případě je zamítnut.

# 6 APLIKACE PRO VIZUALIZACI ASYMETRICKÝCH KRYPTOSYSTÉMŮ

Součástí této práce je také návrh pomůcky pro podporu výuky na téma asymetrických kryptografických systémů. Cílem této aplikace bude co nejlépe vizualizovat činnost vybraných kryptografických systémů, názorně a interaktivně zobrazit průběh ustanovení klíčů v protokolu Diffie-Hellman. Objasnit průběh šifrování v RSA a objasnit funkci podepisování v DSA.

## 6.1 Platforma použitá pro aplikaci

Cílem projektu je vytvořit jednoduchou přenositelnou aplikaci, která nebude vyžadovat instalaci ani konfiguraci a bude nezávislá na platformě ve které bude provozována. Z tohoto důvodu jsem zvolil jako implementační nástroj pro vývoj této aplikace jazyk JAVA. Čímž mám zajištěnu přenositelnost a nezávislost aplikace na operačním systému, na kterém je provozována. To díky tomu že java obsahuje JVM(Java Virtual Machine), což je virtuální stroj ve kterém aplikace běží a tudíž běží na jakékoli platformě stejně. Pro návrh grafického uživatelského rozhrání (Graphic User Interface- GUI) bude použita platforma javaFX, pro usnadnění vývoje použiju pro návrh rozložení GUI nástroj SceneBuilder. Jako vývojové prostředí použiju nástroj Eclipse.

## 6.2 Použité technologie

### 6.2.1 Programovací jazyk Java

Javu vyvinuli v roku 1991 James Gosling, Patrick Naughton, Ed Frank, Chris Warth a Mike Sheridan ze společnosti Sun Microsystems(Oracle). Jazyk byl původně nazýván Oak, roku 1995 došlo k jeho přejmenování na dodnes známý název Java. Již od počátku vývoje byli vývojaři zaměřeni na nezávislost jazyka na platformě tak aby byl použitelný pro široké spektrum zařízení.

Java je objektově orientovaný programovací jazyk (OOP-Object Oriented Programming), který se podobá například jazykům C, C++ a C#. Oproti těmto jazykům je java jednodušší, protože má stručnou, kompaktní sadu rysů usnadňujících používání. Zdrojový kód je při kompilaci převeden na strojově nezávislý bajtový kód (bytecode), který je následně spouštěn na virtuálním stroji JVM (Java Virtual Machine). A proto aplikace napsaná v jazyku Java bude spustitelná na jakékoli počítačové platformě podporující JRE (Java Runtime Environment).

Mezi hlavní výhody Javy patří také velké množství knihoven, kdy programátor už nemusí programovat každou funkci ve svém programu úplně od začátku, ale využije k tomu již napsaný program napsaný jiným programátoram nebo si napíše svou vlastní knihovnu, která je potom snadno přenositelná v kódu a tím je celkový vývoj efektivnější a rychlejší. Knihovny pro Javu tvoří jak vývojaři Sun Microsystems, tak rozsáhlá komunita programátorů, což dodává jazyku Java jako programovacím nástroji velkou sílu. Problematika je dále popsána v [12].

### 6.2.2 JavaFX

JavaFX je je významná a bohatá platforma usnadňující vývoj a tvorbu interaktivních a multimediálních aplikací s grafickým uživatelským rozhraním. Hlavní účel pro který byla tato platforma vytvořena je umožnění vývoje RIA(Rich Internet Applications), které budou funkční bez ohledu na platformu (PC, mobilní telefony, nebo IP TV). JavaFX je jakousi základní sadou nástrojů a knihoven, které umožňují vývojařům tvořit aplikace s multimediálním obsahem. Více o java fx můžete nalézt [11].

### 6.2.3 FXML

FXML je skriptovatelný značkovací jazyk založený na XML vyvinutý společností Oracle. Hlavním účelem, pro který byl jazyk FXML vytvořen je efektivní tvorba Grafického rozhraní aplikací, kdy je FXML soubor provázán s JavaFX projektem a definuje rozložení(layout) uživatelského rozhraní. O FXML dále pojednává text [3].

### 6.2.4 JavaFX Scene Builder

JavaFX Scene Builder je pomůcka pro vizuální tvorbu GUI. Dá se považovat za WYSIWYG editor, fungující na principu drag and drop. Tím pádem se této fázi vývoje obejdeme bez programování. Výstupem Scene Builderu je FXML soubor, který je potřeba provázat s JavaFX projektem, ve kterém už stačí jen naprogramovat logickou část projektu a přiřadit tlačítka k daným funkcím.

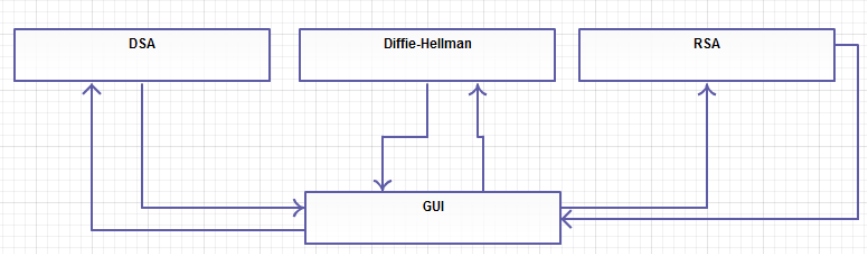
## 6.3 Třída BigInteger

Třída **java.math.BigInteger** umožňuje práci teoreticky s libovolně velkými čísly, dále poskytuje operace analogické ke všem operacím s primitivními datovými typy. Dále však poskytuje modulární aritmetiku, kalkulaci GCD(gratest common divisor-

největší společný dělitel), testy prvočíselnosti, generování prvočísel a spoustu dalších operací. Z toho plyne, že třída BigInteger bude v podstatě základem aplikace, protože dokáže provádět většinu operací potřebných pro výpočty klíčů a další operace u asymetrických kryptosystémů.

## 6.4 Struktura aplikace

Aplikaci budu tvořit jako jeden celek, skládající se ze 4 základních obslužných tříd. Hlavní spouštěcí třída bude zajišťovat běh aplikace, dále bude v této třídě nadefinováno GUI. Dále pak každý kryptosystém bude mít svou vlastní třídu ve které bude definován obslužný algoritmus. Třídy jednotlivých algoritmů kryptosystémů budou na sobě nezávislé tzn. třída pro kryptosystém DSA bude nezávislá na třídě pro RSA. Tato skutečnost bude výhodná pro případ, kdyby bylo aplikaci potřeba rozdělit na samostatné funkční celky a v tomto případě to nebude problémem.




Obr. 6.1: Diagram komunikace mezi třídami.

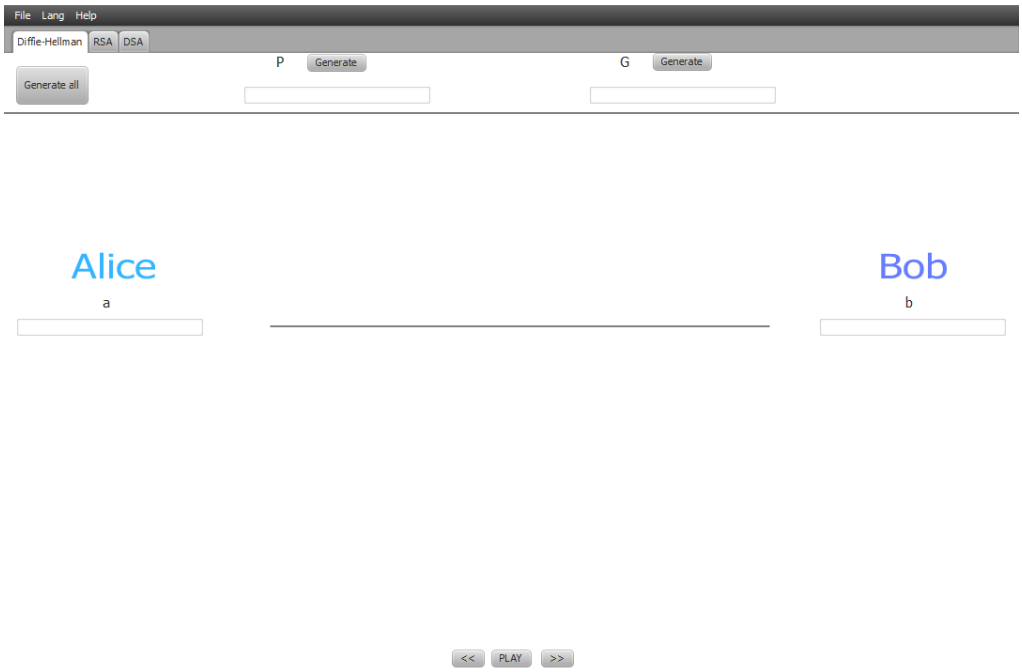
Aplikace bude vyvíjena v prostředí E(fx)clipse, určeném pro vývoj JavaFX aplikací, grafické rozhraní bude vytvořeno pomocí nástroje JavaFX Scene Builder.

Protože cílem aplikace je co nejlépe objasnit funkci asymetrických kryptosystémů, je důležitou volbou rozložení ovládacích prvků aplikace tak, aby její ovládání bylo intuitivní a nenáročné. Dále pak je velice důležitá interaktivita aplikace, pro toto téma ale jsou bohužel možnosti interaktivity aplikace omezeny povahou tématu, jelikož většina probíhajících operací jsou jen výpočty podle vzorců.

## 6.5 Grafické rozhraní

Základní okno aplikace bude obsahovat 3 záložky, každá záložka bude vyhrazena jednomu kryptosystému. Rozhraní bude obsahovat textová pole pro vložení hodnot ze kterých má program vycházet při výpočtech. Samozřejmě bude k dispozici i tlačítko generate, stiskem tohoto tlačítka uživatel vygeneruje náhodnou hodnotu do

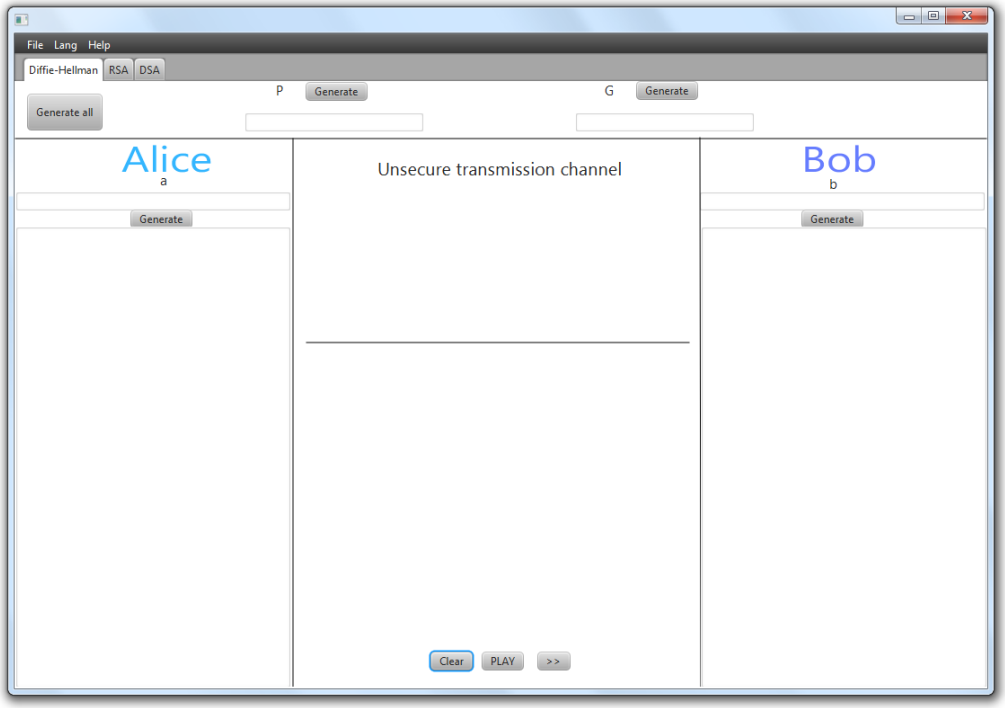
tohoto pole, při stisku tlačítka generate all budou náhodně vygenerovány všechny hodnoty, které jsou vyžadovány na počátku prezentace. Dále pak budou k dispozici tlačítka play, vpřed a vzad, pomocí těchto tlačítek může uživatel buď procházet postup krok po kroku, nebo pomocí tlačítka play nechat plynule přehrávat postup, který bude přerušen jen v případě pokud bude požadováno vložení nějaké vstupní hodnoty a nebo pokud uživatel prezentaci zastaví.




Obr. 6.2: Návrh grafického rozhraní aplikace.

## 6.6 Modul Diffie-Hellmanův protokol

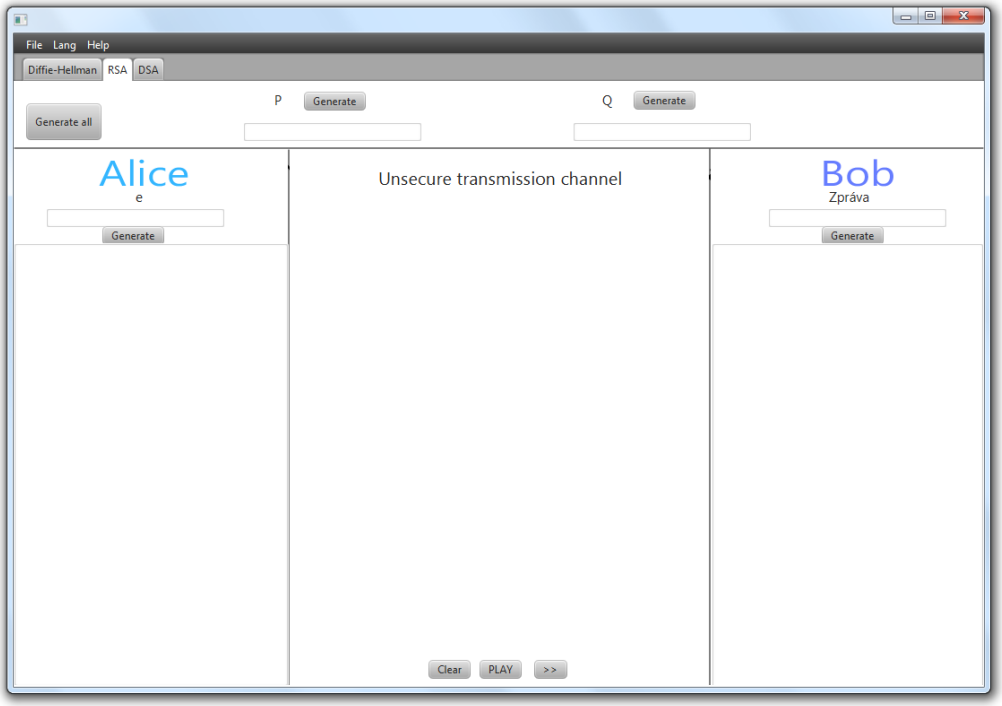
První modul bude zaměřen na demonstraci průběhu ustanovení klíčů pomocí Diffie-Hellmanova protokolu po nezabezpečeném přenosovém kanálu. Pomocí průvodce bude celý tento proces intuitivně znázorněn a vysvětlen. Grafické prostředí je rozděleno na 4 části. První částí je horní lišta, ve které bude zadáno prvočíslo $p$ a generátor $g$, tyto hodnoty jsou veřejné. Pod touto lištou budou umístěny další 3 části, kde na levé a pravé straně budou znázorněny komunikující entity Alice a Bob. Uprostřed je znázorněn přenosový kanél, po kterém budou přenášeny informace. Jednotlivé hodnoty potřebné k výpočtům budou zadávány uživatelem, nebo mohou být vygenerovány stiskem tlačítka generate. Pokud bude zadáno složené čisla namísto prvočísla bude spuštěn algoritmus, který čisla nahradí nejbližším prvočíslem. Grafické rozložení naleznete na obr. 6.3




Obr. 6.3: Grafické prostředí modulu Diffie-Hellmanova protokolu

## 6.7 Modul RSA

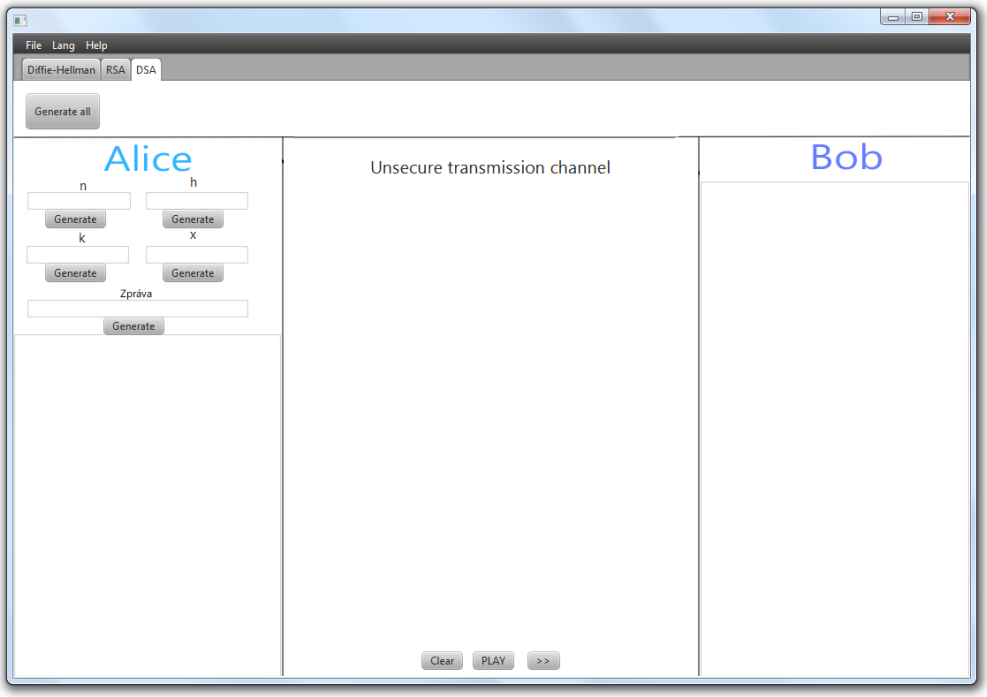
Modul algoritmu RSA stejně jako předchozí modul je zaměřen na vysvětlení problematiky při šifrování metodou RSA. Průvodce v tomto modulu bude postupně zobrazovat kroky výpočtu jednotlivých hodnot a zjednodušeně bude zobrazeno zašifrování a následný přenos zprávy. Co se týká rozložení grafického prostředí, bude z důvodu jednotnosti aplikace téměř shodné s prostředím využitým pro Diffie-Hellmanův protokol, bude jen rozdíl v zadávaných hodnotách a následné animaci.




Obr. 6.4: Grafické prostředí modulu pro RSA

## 6.8 Modul DSA

Tento modul bude sloužit pro krátké seznámení s algoritmem pro podepisování zpráv. Krok za krokem bude předveden proces ustanovení klíčů a vytvoření podpisu a následné autentifikace. Grafické prostřdí bude opět souhlasné s předchozími moduly aby byla zajištěna jednotnost aplikace a tím i větší přehlednost. Rozdíl bude jen ve vyžadovaných hodnotách a následném průběhu prezentace.




Obr. 6.5: Grafické prostředí modulu pro DSA

## 6.9 Hlavní nabídka

Hlavní nabídka je součástí, která propojuje jednotlivé moduly, dále bude obsahovat jednoduchou nápovědu a základní nastavení programu ve kterém bude možno nastavit jazyk ve kterém bude zobrazován interface programu. Hlavní nabídka bude stále viditelná v horní části okna aplikace. A bude v ní možno kdykoliv změnit jazyk programu, otevřít soubor nápovědy a nebo se přepnout na jiný modul, popřípadě zavřít aplikaci.


Obr. 6.6: Hlavní nabídka aplikace

# 7 IMPLEMENTACE

## 7.1 Volba platformy pro implementaci

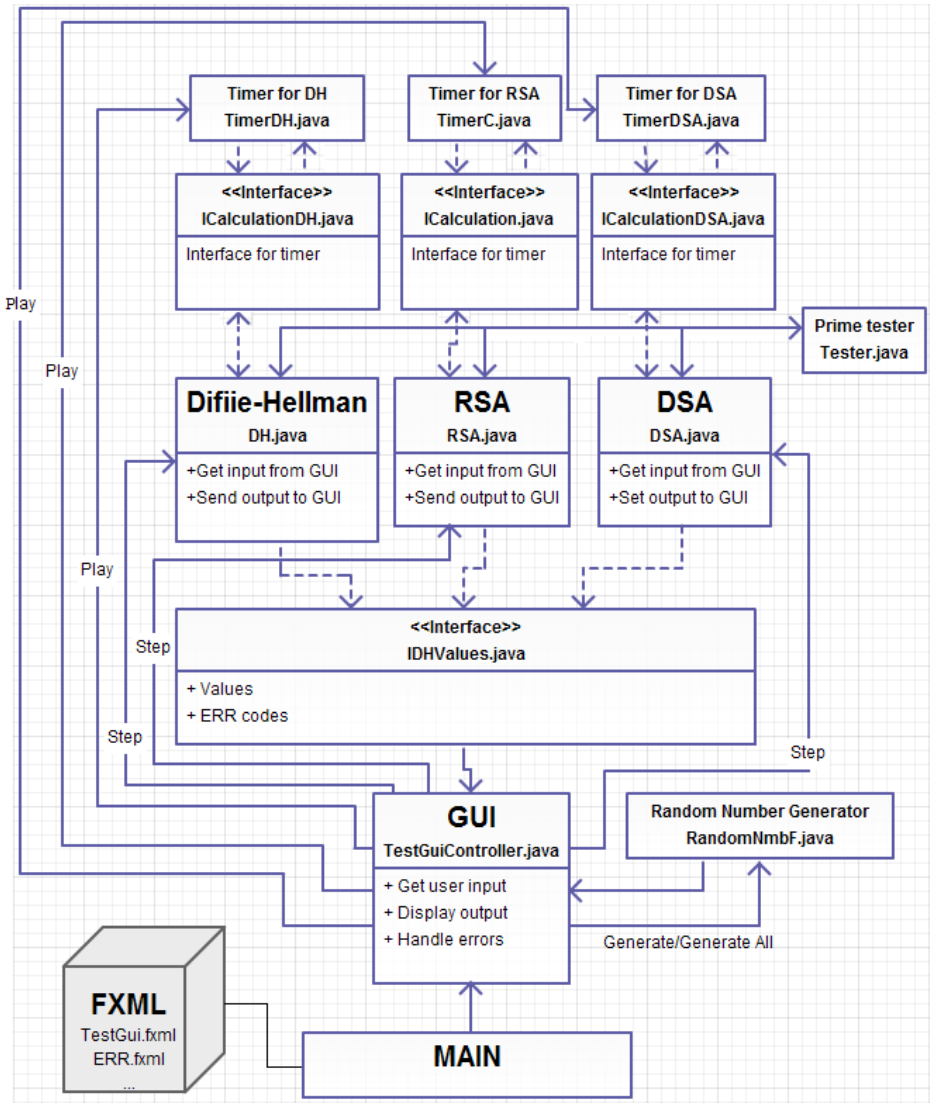
Jak již bylo výše uvedeno, aplikace je implementována na platformě JAVA, která je specifická tím, že při překladu je program převeden na takzvaný byte kód. Tento kód je pak interpretován pomocí (JVM) Java Virtual Machine. Tento způsob má velkou výhodu v tom, že výslednou aplikaci lze spustit na libovolném systému. Nevýhodou může být o něco menší výkon, protože musí být spuštěn virtuální stroj. Mezi výhody také patří množství knihoven. Použití knihoven ušetří programátorovi spoustu času, protože již nemusí vyvíjet funkce, které už jsou vytvořeny a pouze se soustředí na vývoj aplikace jako takové.

## 7.2 Komunikace mezi třídami

V části návrhu již byl uveden diagram komunikace mezi třídami z čistě teoretického hlediska. Ve skutečnosti je však potřeba daleko více elementů pro zajištění funkčnosti aplikace. Z tohoto důvodu zde přikládám na obrázku 7.1 UML diagram znázorňující propojení a komunikaci mezi jednotlivými třídami. Na uvedeném UML diagramu[1] můžete vidět průběh komunikace mezi třídami v závislosti na uživatelském vstupu. Při spuštění programu třída Main načte soubor definicí rozložení grafického uživatelského rozhraní TestGui.fxml. Další operace již probíhají především ve třídě TestGuiController, DSA, DH, RSA. Tyto třídy jsou navzájem propojené prostřednictvím Interface IValues, přes který jsou přenášeny proměnné, které byly zadány do políček uživatelem, nebo vygenerovány stiskem tlačítka. Výsledky výpočtu jsou zpracovány a odeslány prostřednictvím IValues zpět do GUI, kde se zobrazí v předem připravených polích spolu s popisem co bylo provedeno za výpočet a o jakou hodnotu se vlastně jedná.

Nyní si popíšeme co se děje v programu v konkrétních případech. Například pokud uživatel klikne na tlačítko generate je třídou TestGuiController zavolána třída RandomNmbF, což je generátor náhodných čísel. Tato funkce navrátí náhodné číslo, které je vyplněno do příslušného políčka. Jakmile je stisknuto tlačítko Generate All, spustí se ve třídě TestGuiController cyklus-for, který opakovaně volá náhodný generátor a postupně vyplňuje všechna políčka.

[1] Unified Modeling Language




Obr. 7.1: UML diagram

Má-li uživatel aktuálně rozkliknutou záložku pro Diffie-Hellmanův protokol a stiskne tlačítko play, je z třídy TestGuiController zavolán konstruktor třídy TimerDH. Jak již z názvu vyplývá jedná se o jednoduchý časovač. Tento časovač odešle prostřednictvím Interface ICalculationDH požadavek do třídy DH na provedení prvního kroku výpočtu. Třída DH získá prostřednictvím IValues všechny hodnoty zadané v polích v záložce DH a přejde k ověření jejich správnosti. Jsou-li v polích zadány platné hodnoty (jsou zadána celá čísla, pole nejsou prázdná), odešle třída DH třídě Prime tester čísla, u kterých je vyžadováno aby byly prvočísly. Pokud jsou zadaná čisla prvočísly, pokračuje výpočet normálně dál. Pokud některé ze zadaných čísel není prvočíslo, je automaticky upraveno a zároveň je odeslán chybový kód do TestGui controlleru, který zobrazí chybovou hlášku a tím informuje uživatele o této skutečnosti. Při této chybě však není běh výpočtu narušen. Po úspěšném ověření, popřípadě i úpravě vstupních hodnot třída DH informuje časovač návratovou hodnotou typu boolean-true že může načasovat další krok. Pokud by některé ze vstupních políček bylo prázdné, nebo by obsahovalo neplatnou hodnotu je časovači navrácena hodnota false. Což způsobí zastavení časovače.

Třída DH dále odešle chybový kód do TestGuiControlleru, který zobrazí chybovou hlášku a tím pádem je uživatel vyrozuměn o této skutečnosti. Časovač vlastně odesíla pouze impulzy k provedení operace, ale časovač ïneví ö tom který výpočet probíhá (zda-li probíhá první nebo poslední krok výpočtu). O toto se již stará samotná třída DH tím způsobem, že při provedení každého kroku je inkrementována servisní proměnná o 1. V případě výskytu závažné chyby (napr. neplatné číslo, prázdné pole) je tato proměnná vynulována. Kdyby servisní proměnná nebyla nulována nebyla by chyba vařešena, protože by výpočet pokračoval i po přepisu políček stále dál s původním neplatným vstupem a pomohl by jen restart programu. Jestliže uživatel zadal správné hodnoty, výpočet pokračuje plynule dál, s tím že na konci každého kroku je navrácena do časovače hodnota true, výsledky výpočtu jsou odesílány prostřednictvím IValues do TestGuiController, kde je k ním přidán popis a následně jsou zobrazeny v příslušné části okna. Jakmile proběhne celý výpočet je navrácena do časovače hodnota false, to způsobí jeho zastavení. Servisní proměnná je vynulována aby mohl uživatel zadát jiné hodnoty a spustit protokol znovu. Při opětovném stisku tlačítka play, jsou před započetím výpočtu vyčištěna výstupní textová pole.

Stiskne-li uživatel tlačítko pro krokování (»), je třídou TestGuiController odeslán požadavek na provedení prvního kroku výpočtu přímo do DH. Výpočet probíhá stějně jako při stisku tlačítka play, jen je jiný spouštěcí mechanismus. Již zde není využito časovače ale pouze vstupu od uživatele. Sled početních operací určuje opět servisní proměnná, která je při každém stisku inkrementována o 1. V tomto případě je ale přepínací mechanismus a servisní proměnná umístěna v TestGuiControlleru

a volá postupně jednotlivé metody z DH. Je zde nastaven počet kliků, po kterých je servisní proměnná vynulována, tak aby mohl výpočet probíhat opakovaně. S každým započetím prvního kroku jsou vyčištěny výstupy aby neobsahovaly informace z předchozích výpočtů, což by bylo nepřehledné a matoucí.

Dále je uživateli k dispozici tlačítko vyčistit. Toto tlačítko vymaže veškeré hodnoty obsažené ve vstupních a výstupních polích. Tato funkce je velice jednoduchá, protože pro každé pole provede operaci `textovepole.setText=""`.

Výpočty jsou implementovány s využitím funkcí třídy BigInteger, z toho důvodu, že třída BigInteger již má implementovánu většinu potřebných matematických operací, které lze s výhodou využít pro demonstraci kryptografických protokolů.

## 7.3 Generátor náhodných čísel

Generátor náhodných čísel je spuštěn pokaždé při kliku uživatele na tlačítko sloužící pro vygenerování náhodného čísla. V této třídě se nachází tři metody, v každé metodě je implementován náhodný generátor. Tyto tři metody se liší pouze délkou vygenerovaného čísla. Nastavení délky vygenerovaného čísla není uživatelovi zpřístupněno, ke každém tlačítku a výstupním poli je napevno připojena metoda, která generuje číslo o požadované délce. Například v DSA jsou generována krátká čísla z důvodu náročnějších výpočtů.

```
public static BigInteger getRandomBigInteger() {
  Random rand = new Random();
  BigInteger rndNum = BigInteger.probablePrime(99, rand);
  return rndNum;
}
public static BigInteger getRandomBigIntegerSmallest() {
  Random rand = new Random();
  BigInteger rndNumSmall = BigInteger.probablePrime(9, rand);
  return rndNumSmall;
}
```

Obr. 7.2: Ukázka kódu generujícího náhodná čisla o různé délce

## 7.4 Tester prvočíselnosti

Protože je nutné, aby některé vstupní hodnoty byly prvočíselného typu, jsou tyto vstupy kontrolovány testerem prvočíselnosti. Tester ověří zda je zadané čislo prvočíslem. Pokud ano je číslo ponecháno beze změny a výpočet může pokračovat. Pokud zadané číslo není prvočíslem je inkrementováno o 1 a znovu testováno. Číslo je inkrementováno tak dlouho dokud není prvočíselným testem prohlášeno za prvočíslo. Jakmile je číslo prohlášeno za prvočíslo je uživatel upozorněn na to že zadal špatné číslo, ale výpočet pokračuje dál. Upozornění uživatele je implementováno ve výpočetních třídách, ne v testeru. Tester rovněž využívá třídu BigInteger konkrétně metodu java.math.BigInteger.isProbablePrime tato metoda navrací hodnotu boolean pokud je testované číslo prvočíslo navrátí hodnotu true, v opačném případě je navrácena hodnota false.

```
import java.math.BigInteger;

public class tester {

public static BigInteger getNextPrime(String ans){
  BigInteger test = new BigInteger(ans);
  while (!test.isProbablePrime(99))
    test = test.add(BigInteger.ONE);
  return test;
}
}
```

Obr. 7.3: Ukázka kódu prvočíselného testu

V ukázce lze zpozorovat že je zde použitý cyklus while, který se opakuje tak dlouho dokud není navrácena hodnota true.

## 7.5 Časovač

Aby po kliknutí na tlačítko play, program nezobrazil nepřiměřeně mnoho číselných údajů najednou, je výpočet řízen časovačem, jenž mezi jednotlivými kroky uměle vatváří zpoždění. Výše již bylo zmíněno, že je zavedena i zpětná vazba, pomocí které je časovač v případě potřeby zastaven. Nastavení délky trvání časovače není uživateli přístupno. Zpoždění je pevně nastaveno ve zdrojovém kódu. V programu jsou celkem tři časovače, každý pro jeden protokol. Například Diffie-Hellmanův protokol má svůj časovač, RSA má svůj časovač. Díky tomu lze nastavit pro každý protokol jiné zpoždění, tak aby uživatel měl dostatek času na přečtení.

```
public class TimerDH {
  private final ICalculationDH calculation;
  private final Timer casovac;

  public TimerC(final ICalculationDH calculation) {
    this.calculation = calculation;

    TimerListener listener = new TimerListener();
    casovac = new Timer(990, listener);
    casovac.setRepeats(true);
    return;
  }

  public void start() {
    casovac.start();
  }

  private class TimerListener implements ActionListener {

    @Override
    public void actionPerformed(ActionEvent e) {
      if (!calculation.makeStep())
        casovac.stop();
    }
  }
}
```

Obr. 7.4: Ukázka kódu časovače

## 7.6 Výpočetní část programu

Pomocné třídy programu jsou v předchozím textu již dostatečně popsány. Nyní je nařadě výpočetní část programu. Každý protokol má svoji vlastní třídu s výpočetní částí. Zde jsou zpracovány vstupní hodnoty a výsledky jsou následně zobrazeny na výstupu. Ve výpočetních třídách je také řešeno zobrazování některých chybových hlášení.

Jako názornou ukázku si zde rozebereme výpočetní část Diffie-Hellmanova protokolu:

První krok, který je proveden po stisknutí tlačítka » nebo Play, je nastavení vstupních proměnných. Program ze vstupních polí zkopíruje vložené údaje do pracovních proměnných textita, b, p, g. Tyto proměnné jsou typu BigInteger. U proměnné textitp je proveden test prvočíselnosti. Následně je porovnáno číslo p s číslem pReff. Číslo pReff je hodnota p před provedením prvočíselného testu a číslo p je hodnota po provedení prvočíselného testu. Pokud jsou hodnoty stejné výpočet pokračuje dál, jestli se hodnoty liší je zobrazeno vyskakovací okno, které upozorní uživatele že zadané číslo p bylo upraveno a výpočet pokračuje. Poslední řádek nastaví upravenou hodnotu do políčka pro p.

```
a = values.getA();
b = values.getB();
pReff = new BigInteger(values.getP());
p = tester.getNextPrime(values.getP());
g = values.getG();
if (p.compareTo(pReff) != 0) {
  Stage ERR113 = new Stage();
  ERR113.initStyle(StageStyle.UTILITY);
  Scene ERRs113 = new Scene(new Group(new Text(25, 25, "Cislo p : "
      + pReff + "neni prvocislo a bylo upraveno na hodnotu" +p
      + "\n")));
  ERR113.setScene(ERRs113);
  ERR113.show();
}
values.setPrimalizedP(p.toString());
```

Obr. 7.5: Ukázka prvního kroku výpočtu DH protokolu

## 7.7 GUI

V této části textu bude popsána ta část programu kterou může uživatel vidět. A to je Grafické Uživatelské Rozhraní. tato část se stará o zobrazení programu, dále

Dále je proveden test znaménka vstupních hodnot. Pokud některá z hodnot není kladná je poslán chybový kód do GUI, které zobrazí chybovou hlášku. Pokud nebyl průběh programu přerušen chybou je spočítána hodnota $resultA = g^a \bmod p$ tato hodnota je následně odeslána do GUI, kde je zobrazena ve výstupním poli Alice. Navíc ja k této hodnotě přidán popis, přidání popisu má za úkol GUI, tedy třída TestGuiController.

```
    if ((a.signum() == -1) || (b.signum() == -1) || (g.signum() == -1))
        {
      values.sendERRcode(012);
    } else {
      values.sendERRcode(0);
    }
    resultA = g.modPow(a, p);
    values.setResultA(resultA);
```

Obr. 7.6: Ukázka prvního kroku výpočtu DH protokolu část 2

Alice vypočtené číslo A pošle bobovi, toto je v kódu provedeno jednoduše nastavením stejné hodnoty do výstupního textového pole Boba: `values.AforBOB(resultA);`
V dalším kroku je spočteno číslo B a jeho hodnota je odeslána do GUI, kde je zobrazeno ve výstupním textovém poli Boba:

```
resultB = g.modPow(b, p);
values.setResultB(resultB);
```

Toto číslo je následně posláno do GUI kde je vytištěno do výstupního pole Alice: `values.BforAlice(resultB);`

Nakonec je spočten klíč pro Alici a Boba, tato klíče musí vyjít stejně, protože to je podstata DH protokolu. Poslední řádek opět odesílá vypočtené hodnoty do GUI kde jsou tyto hodnoty vytištěny na příslušný výstup.

```
    BigInteger KeyAlice = resultB.modPow(a, p);
    BigInteger KeyBob = resultA.modPow(b, p);
    values.setKey(KeyAlice, KeyBob);
```

Ostatní protokoly jsou implementovány podobným způsobem, samozřejmostí je, že mají jiný počet kroků a také odlišné výpočty. Koncept je ale stejný neboť jsem nejdřív implementoval DH protokol, ze kterého jsem vycházel u ostatních protokolů.

odesílá zadané hodnoty do výpočetních tříd a také naopak příjmá výsledné hodnoty které zobrazuje. GUI také zajišťuje funkčnost veškerých tlačítek, zobrazování většiny chybových hlášek atd. Jak již bylo popsáno výše layout programu je definován pomocí FXML souboru. Nyní si popíšeme jak je implementována funkce tlačítka generovat vše:

Po stisknutí tlačítka Generovat vše se spustí cyklus for, který postupně inkrementuje proměnnou i. Pokud má proměnná hodnotu 1 zavolá se generátor náhodných čísel a výstupní hodnota generátoru se vloží do prvního pole. Pokud je hodnota proměnné 2 opět se zavolá generátor a jeho výstup je vložen do druhého políčka. Tato akce je opakována pro všechna pole. V našem případě 4.

```
public void btnGenerateDHAllFired(ActionEvent event) {
    for (int i = 1; i < 5; i++) {
      BigInteger res1 = RandomNmbF.getRandomBigInteger();
      BigInteger res2 = RandomNmbF.getRandomBigIntegerSmall();
      String randd = (res1.toString());
      if (i == 1) {
        inputAInDH.setText(randd);

      ...

      if (i == 4) {
        inputDInDH.setText(randd);
      }
    }
  }
```

Obr. 7.7: Ukázka funkce pro vyplnění všech polí náhodným číslem.

Pro tlačítka generovat je tato funkce obdobná, jen je podstatně jednodušší. Již zde není potřeba cyklus-for ale stačí zde pouze zavolat generátor čísel a jeho výstup následně vložit do textového pole.

### 7.7.1 Tlačítko Play

Tlačítko play obsahuje velice jednoduchý kód. Tento kód po stisknutí tlačítka nejprve smaže obsah ve výstupních polích a následně zavolá časovač.

### 7.7.2 Tlačítko »

Toto tlačítko při každém stisku inkrementuje servisní proměnnou a spouští jednotlivé kroky výpočtu. Servisní proměnná textitServValDH je při každém stisku inkremen-

```
public void btnPlay1Fired(ActionEvent event) {
    AliceArea.setText("");
    BobArea.setText("");
    timerDH.start();
  }
```

Obr. 7.8: Ukázka obslužného kódu reagujícího na stisk tlačítka Play.

tována o 1. Pokud má servisní proměná hodnotu 1 je zavolána metoda calculateA třídy DH. Pokud ServValDH=2 je zavolána metoda afb třídy DH. Tímto způsobem jsou prováděny jednotlivé kroky při každém kliku až dokud ServValDH nenabyde hodnoty 6, kdy je vynulována, v našem případě je nastavena na hodnotu -1, čímž je vatvořen jeden prázdný krok Kdyby byla nastavena na 0 fungoval by kód stejně dobře jen bez prázdného kroku.

```
public void btnNextDHFired(ActionEvent event) {
    ServValDH++;
    if (ServValDH == 1) {
        dh.calculateA();
    }
    if (ServValDH == 2) {
      dh.afb();
    ...
    }
    if (ServValDH == 6) {
      ServValDH = -1;
    }
```

Obr. 7.9: Ukázka obslužného kódu reagujícího na stisk tlačítka ».

## 7.8 Chybové hlášky

Protože nikdo není neomylný je třeba, zajistit odolnost programu vůči chybám na vstupu a dále upozornit uživatele na tyto chyby. Z tohoto důvodu jsou v programu zavedeny chybové hlášky, které uživatele upozorí na to, že zadal neplatnou hodnotu nebo zapomněl některé vstupní pole prázdné. Tato funkce je zajištěna samostatnou metodou umístěnou v GUI. V případě chyby je tato metoda volána z výpočetního algoritmu.

```
//Showing errors(if any)
@Override
public void sendERRcode(int ERR) {
  if (ERR == 011) {
    Parent root = null;
    try {
      root = FXMLLoader.load(getClass().getResource(
          language + ".fxml"));
    } catch (IOException ex) {
      ex.printStackTrace();
    }
    Stage chyba = new Stage();
    chyba.setTitle("Chyba 011");
    chyba.setScene(new Scene(root, 170, 50));
    chyba.show();

  }
```

Obr. 7.10: Ukázka kódu chybového hlášení

Pokud chybový kód odeslaný z výpočetní části (ERR) má hodnotu 011 je načten nový FXML soubor, který je definován proměnnou language. Název souboru je definován proměnnou z toho důvodu, aby když uživatel změní jazyk v hlavní nabídce byl změněn i jazyk chybových hlášek. Pro každý jazyk je obsažen samostatný fxml soubor pro chybové hlášky. Po načtení FXML souboru je vatvořena nová javaFX Stage, dále je nastaven titulek a vytvořena nová scéna. Poslední příkaz zobrazí okno s chybou.

### 7.8.1 Ošetření exception při provádění prvního kroku

Při spouštění prvního kroku algoritmu je vstup ošetřen tak, že je volání metody calculateA() třídy DH obaleno do try a catch. V případě zachycení výjimky je již známým způsobem vytvořeno okno s chybovou hláškou.

```
public void btnNextDHFired(ActionEvent event) {

      try {

        dh.calculateA();

      } catch (Exception e) {
          Parent root = null;

          try {
            root = FXMLLoader.load(getClass().getResource(
              language + ".fxml"));
        } catch (IOException ex) {
              e.printStackTrace();

        }

        Stage chyba = new Stage();
        chyba.setTitle("Chyba 011");
        chyba.setScene(new Scene(root, 170, 50));
        chyba.show();
      }
    }
```

Obr. 7.11: Ukázka kódu chybového hlášení

Vnitřní fungování programu již bylo vysvětleno dostatečně. Fungování a vzhled programu z pohledu uživatele, naleznete v příloze A, v této příloze naleznete ukázky modulu pro DH protokol. Ukázku modulu pro RSA naleznete v B. V příloze C naleznete ukázku DSA modulu.

# 8 ZÁVĚR

V úvodu této práce je v krátkosti představen vývoj kryptografie od prvopočátku až po dnešní moderní kryptografické systémy. Dále jsou shrnuty základní pojmy a používané v kryptografii, jsou popsány základní typy kryptosystémů.

Tato práce je zaměřena především na asymetrické kryptosystémy. Konkrétně na kryptosystémy Diffie-Hellman, RSA, a algoritmus digitálního podpisu DSA. Tyto kryptosystémy jsou v projektu podrobně popsány a je zde také uveden matematický aparát na základě kterého jednotlivé algoritmy pracují.

Výstupem projektu je také aplikace pro podporu výuky asymetrických kryptosystémů. V rámci práce bylo navrženo Grafické uživatelské rozhraní této aplikace a následně byla tato aplikace vytvořena. Aplikace je napsána v programovacím jazyku Java.

Aplikace slouží jako demonstrační pomůcka, která uživateli pomůže pochopit funkci asymetrických kryptosystémů DH, RSA, DSA. Aplikace je také vhodná pro ověření výpočtů, díky tomu, že je zpřístupněno uživateli zadávat vlastní hodnoty do programu. Díky uživatelskému rozhraní je ovládání aplikace snadné a nevyžaduje speciální znalosti.

Implementace aplikace je podrobně popsána v 7. kapitole této práce. Je zde na ukázkách zdrojového kódu vysvětleno jak jsou naprogramovány jednotlivé funkce tohoto programu.

V osmé kapitole naleznete ukázku hotové aplikace jakožto výsledku této práce.

# LITERATURA


[1] BURDA, Karel. *Bezpečnost informačních systémů.* Brno : FEKT Vysokého učení technického v Brně., 2013. 149 s.

[2] BĚDAJÁNEK, O. *Infrastruktura veřejných klíčů.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. 67 s.

[3] BROWN, Greg. *Introducing FXML*[online]. 2011-8-15 [cit. 2013-12-26]. Dostupné z URL: <http://fxexperience.com/wp-content/uploads/2011/08/Introducing-FXML.pdf>.

[4] KŘÍŽ, J. *Softwarová podpora výuky kryptosystémů založených na problému diskrétního logaritmu.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 74 s.

[5] LEPKA, Karel. *Malá fermatova věta*[online]. [cit. 2013-12-26]. Dostupné z URL: <http://bart.math.muni.cz/~fuchs/ucitel/clanky/1_3_5.pdf>.

[6] Ľ. Balková. *Úvod do kryptologie.* FJFI ČVUT, Praha, 2011.

[7] MENEZES, Alfred, VAN OORSCHOT, Paul, VANSTONE, Scott. *Handbook of applied cryptography.* Boca Raton : CRC Press, 1997. 780 s. ISBN 0849385237.

[8] McCURLEY, Kevin, S.. *The Discrete Logarithm Problem* [online]. 1990 [cit. 2013-12-26].

[9] MIČKA, Pavel. *Java pro začátečníky*[online]. [cit. 2013-12-26]. Dostupné z URL: <http://www.algoritmy.net/article/21340/Uvod-1>.

[10] MOČKA, Pavel. *Eratosthenovo síto*[online]. [cit. 2013-12-26]. Dostupné z URL: <http://www.algoritmy.net/article/65/Eratosthenovo-sito>.

[11] PREMKUMAR, Lawrence, MOHAN, Praveen. *Beginning JavaFX.* Apress, 2010. 107 s. ISBN 978-1-4302-7199-4.

[12] SCHILDT, Herbert. *Java7 Výukový kurz* . Albatros : Computer Press, Brno 2012. 640 s. ISBN 978-80-251.

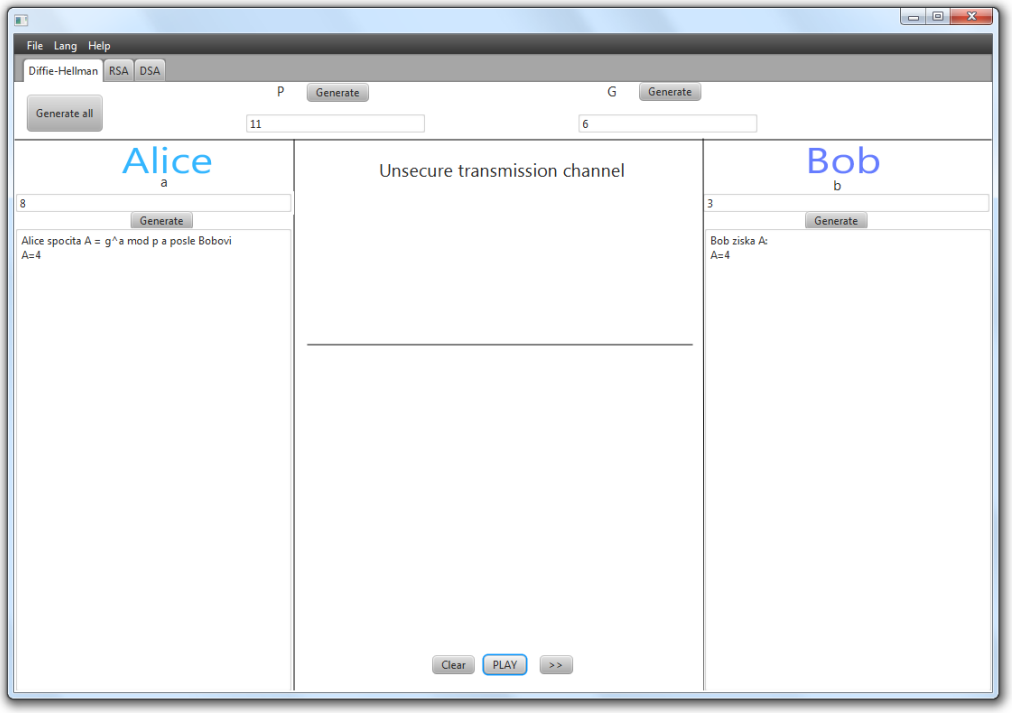
[13] TILL, Michal. *Historie kryptografie I*[online]. [cit. 2013-12-26]. Dostupné z URL: <http://www.krypta.cz/articles.php?ID=55>.

[14] WALEK, V. *Moderní asymetrické kryptosystémy.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2011. 63 s.

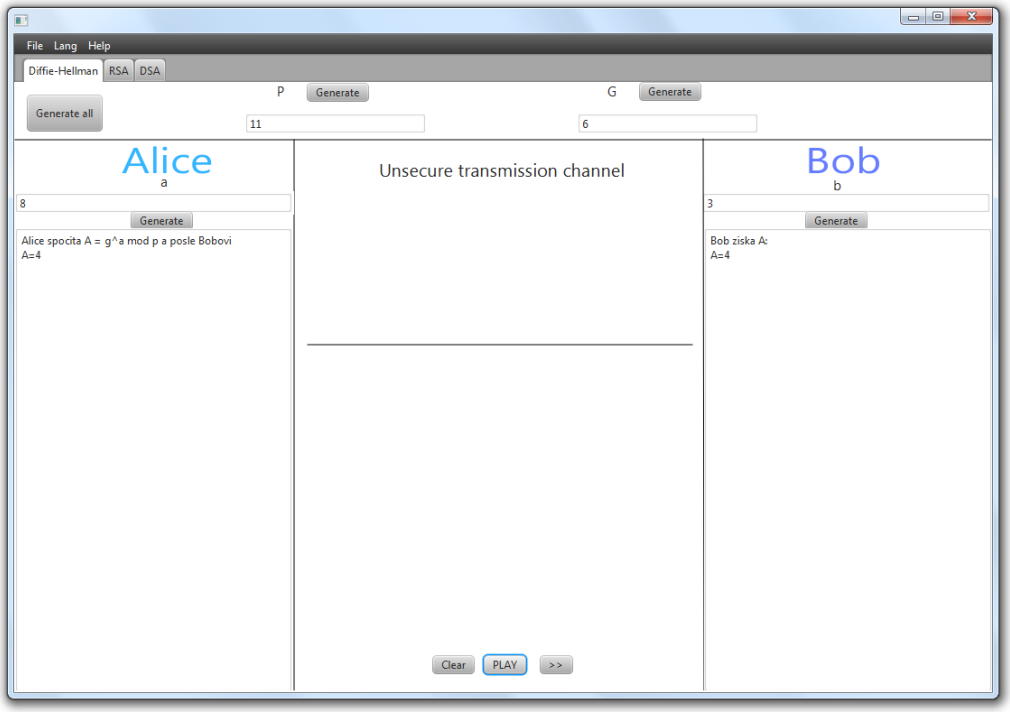
# SEZNAM PŘÍLOH

# A UKÁZKA PROGRAMU MODULU PRO DH PROTOKOL

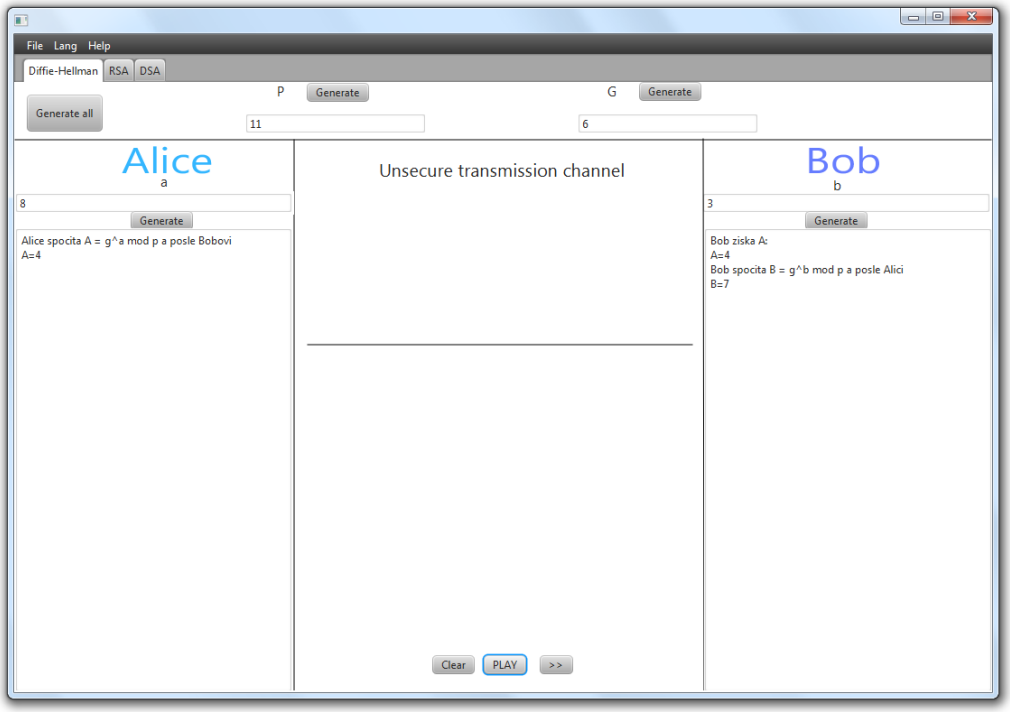
Pro názornost jsem do programu zadal stejné hodnoty, které byly použity v příkladu v teoretické části. Při porovnání výsledku s příkladem, zjistíme že klíč vyšel stejně. Program tedy pracuje správně.



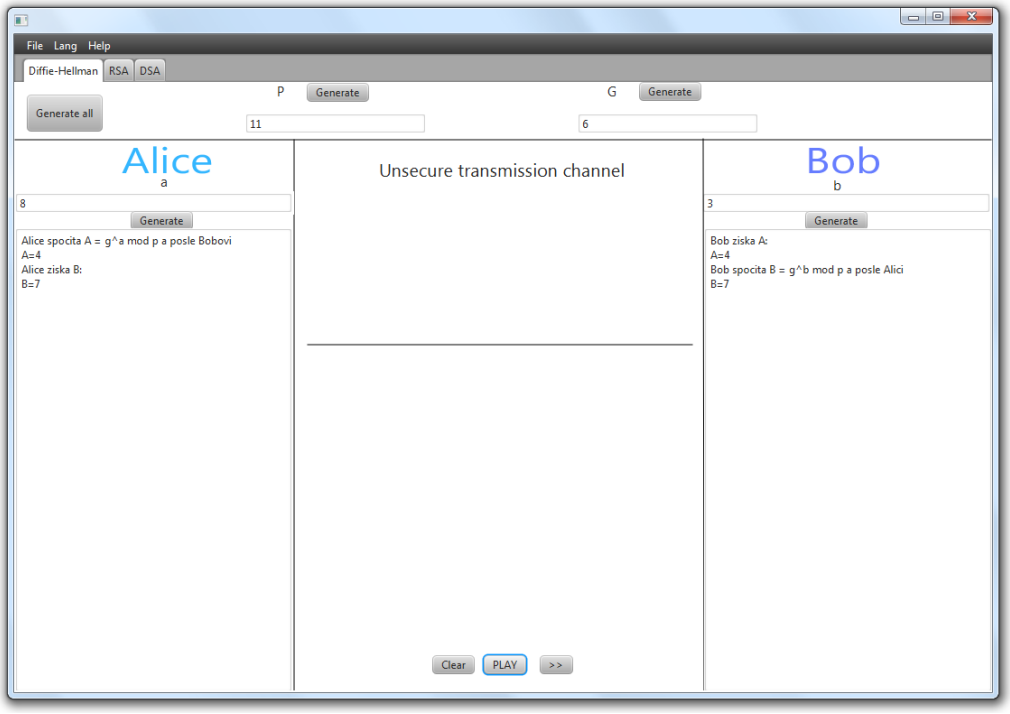

Obr. A.1: Výpočet čísla A.



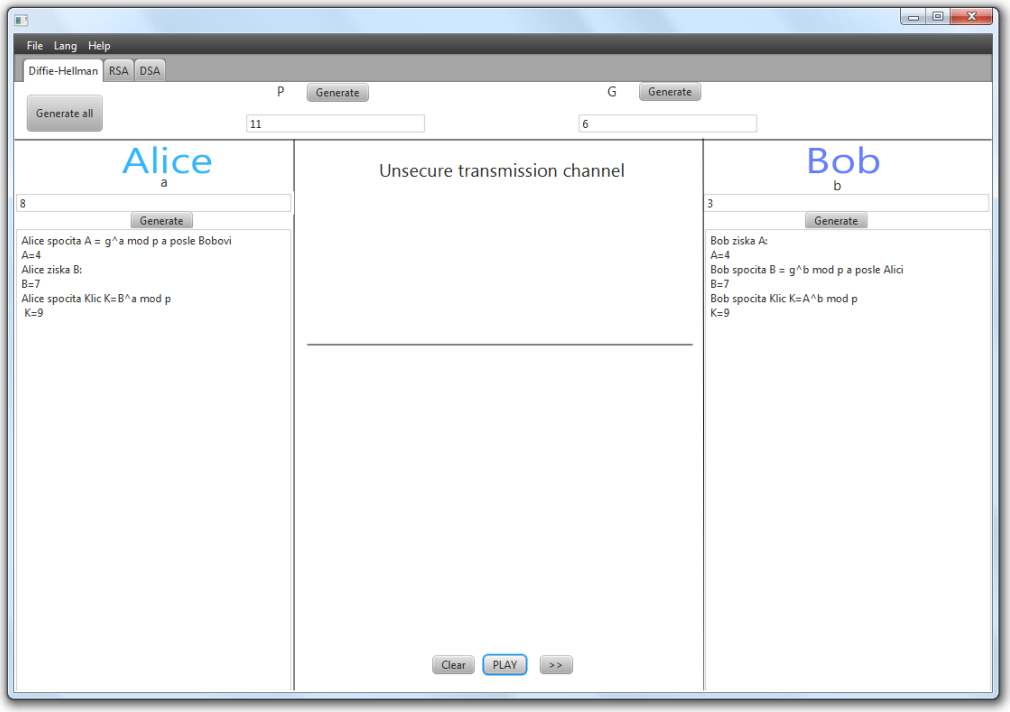

Obr. A.2: Předání Bobovi.




Obr. A.3: Výpočet čísla B.



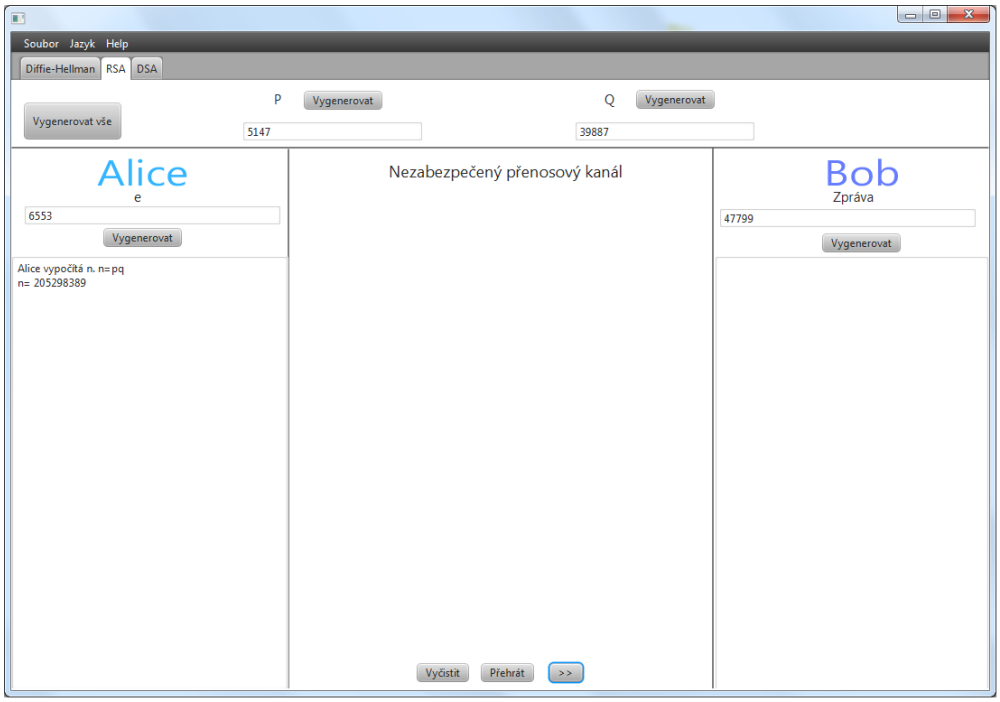

Obr. A.4: Předání Alici.



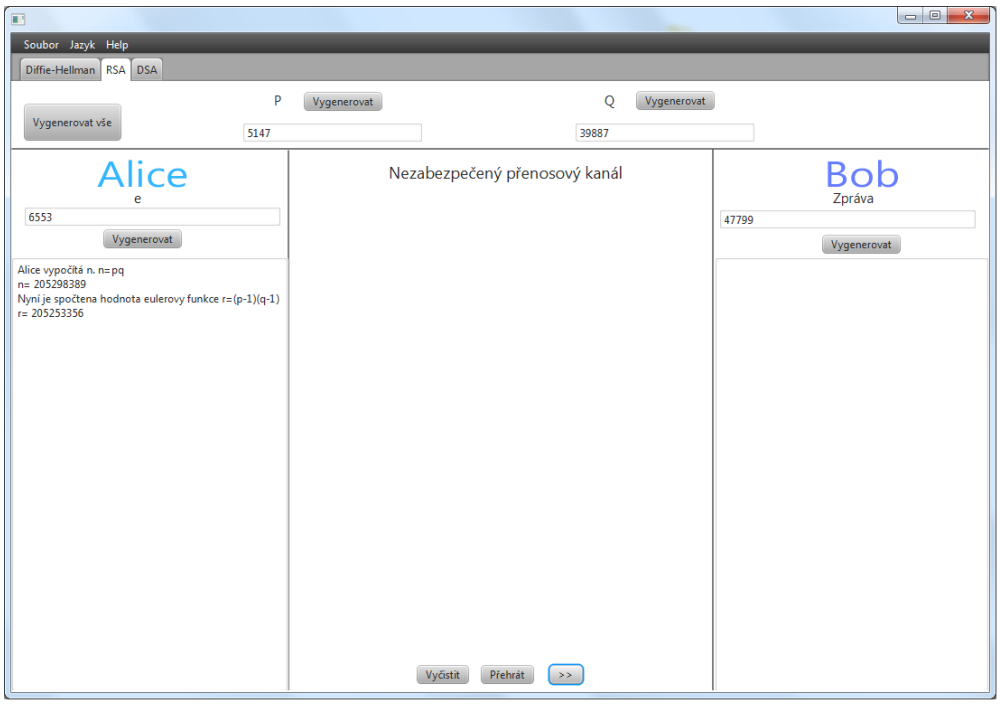

Obr. A.5: Výpočet klíčů.

# B UKÁZKA MODULU PRO RSA

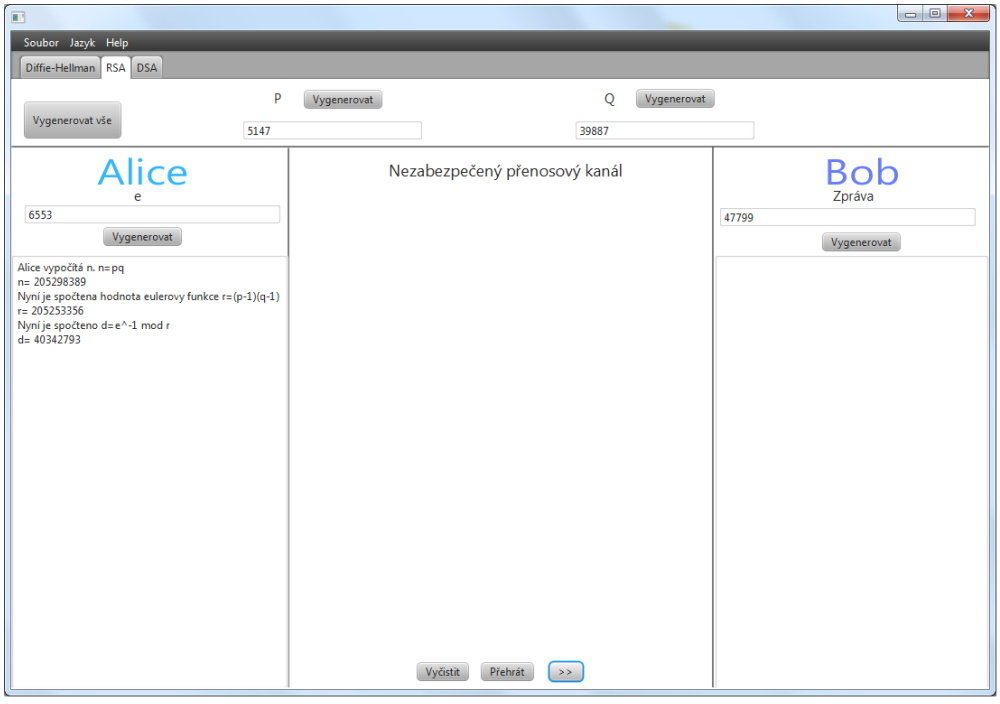
Program disponuje také funkcí pro výpočet RSA klíčů a šifrování zprávy, která je zde reprezentována číselnou hodnotou. U Alice je pak tato zpráva dešifrována.



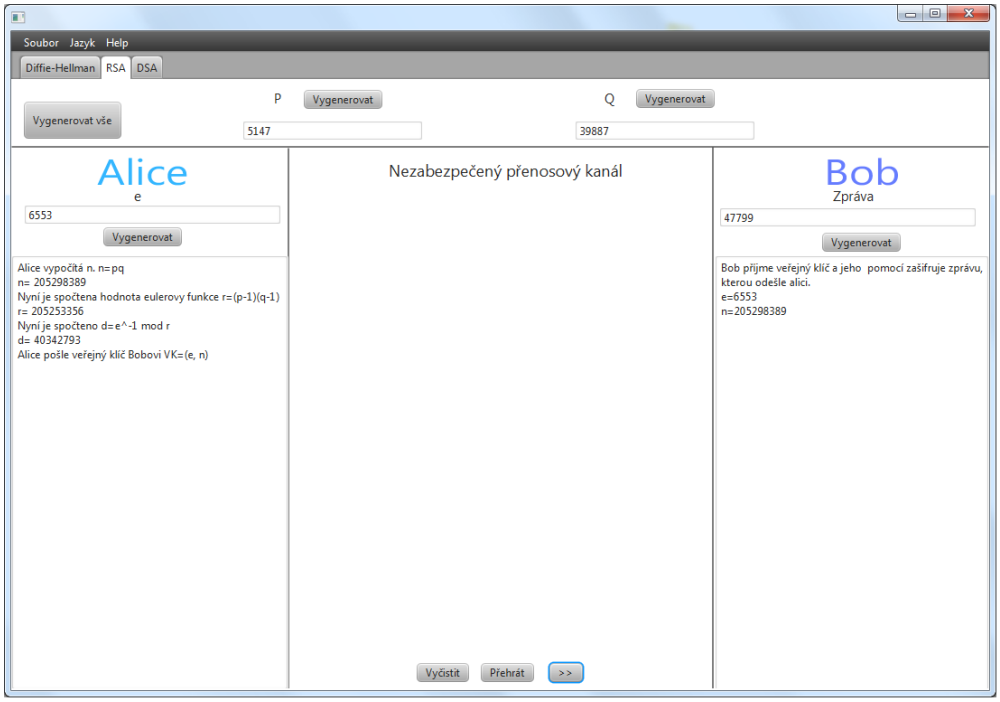

Obr. B.1: Výpočet hodnoty n



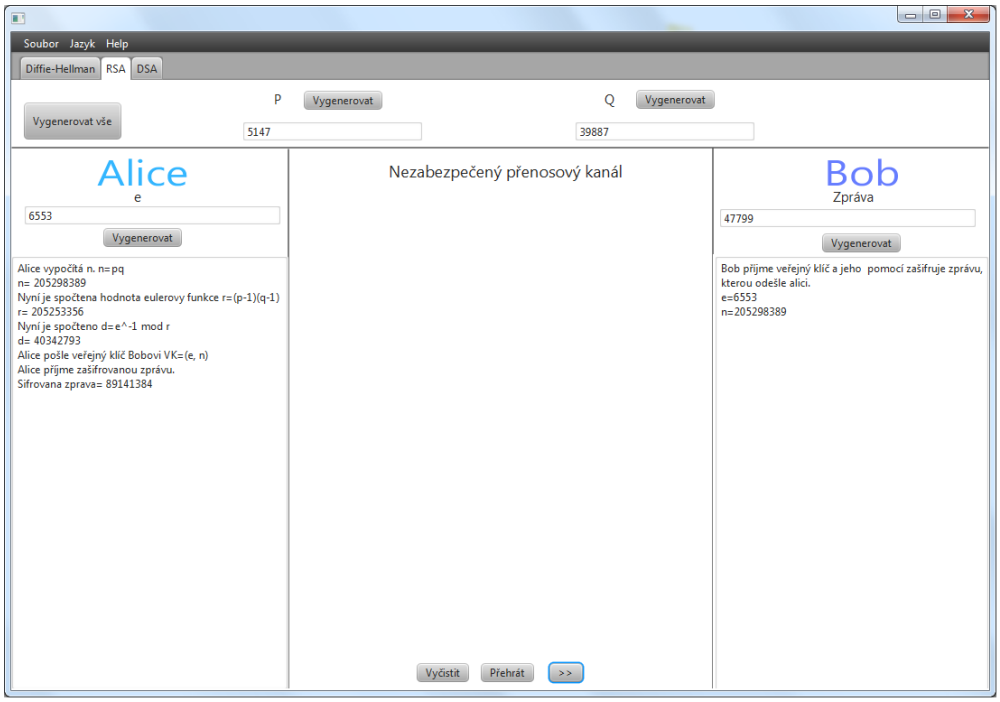

Obr. B.2: Výpočet r



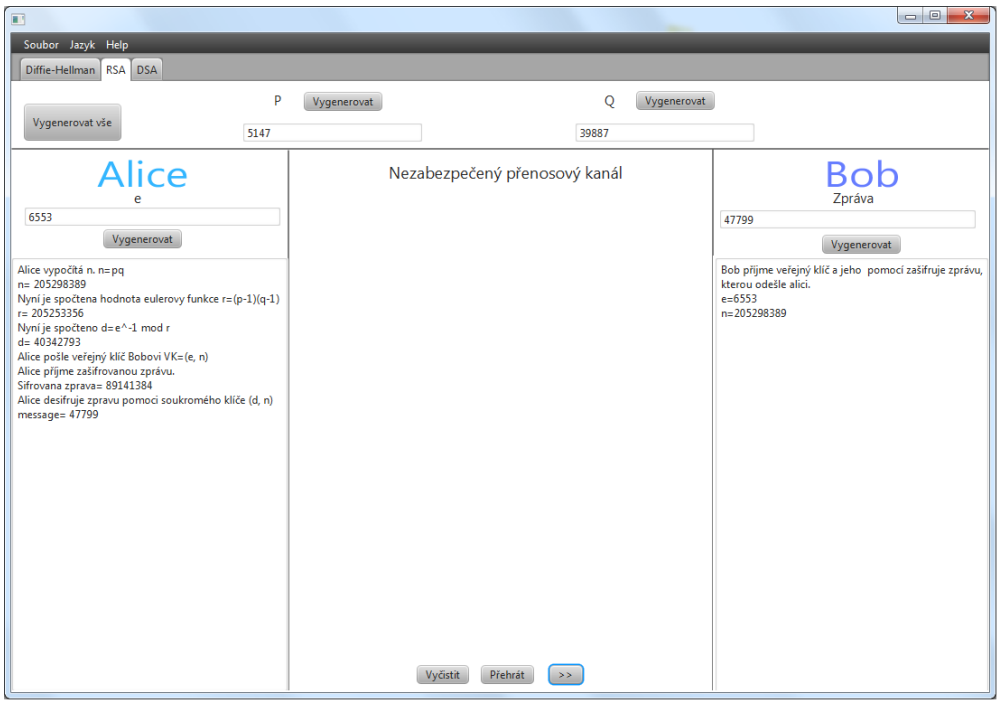

Obr. B.3: Výpočet čísla d.



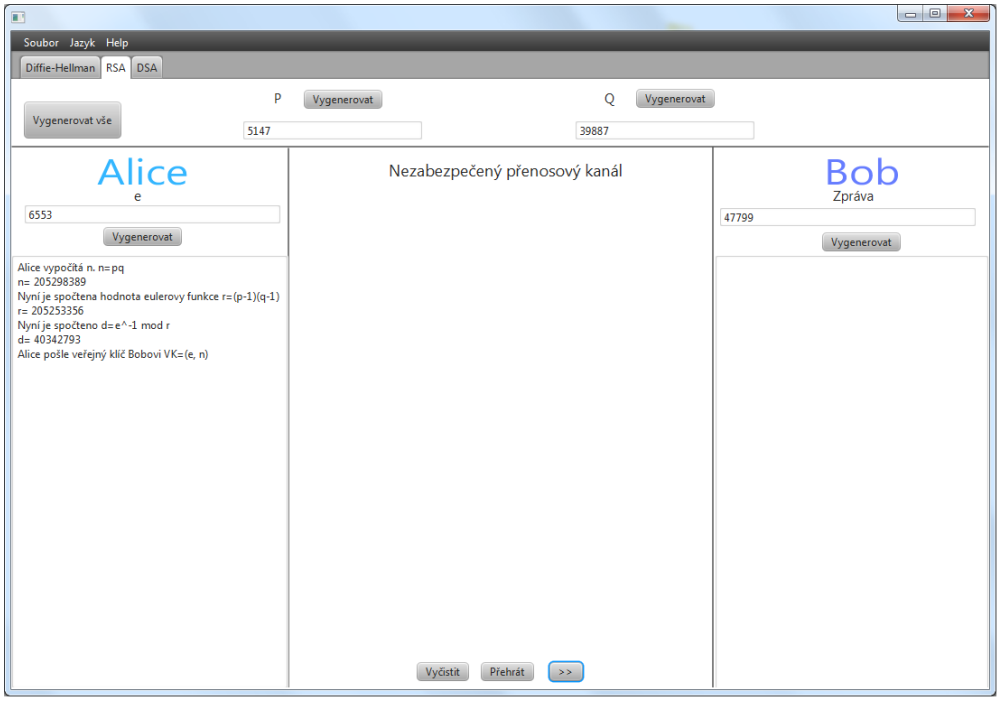

Obr. B.4: Předání klíče bobovi.



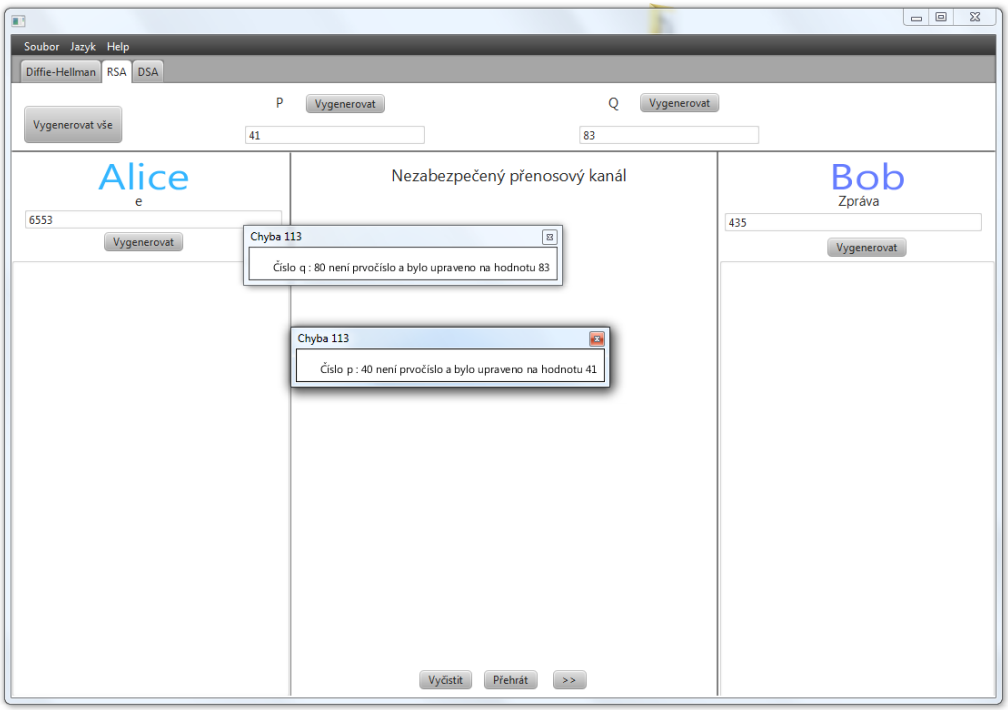

Obr. B.5:




Obr. B.6: Bob šifruje zprávu.



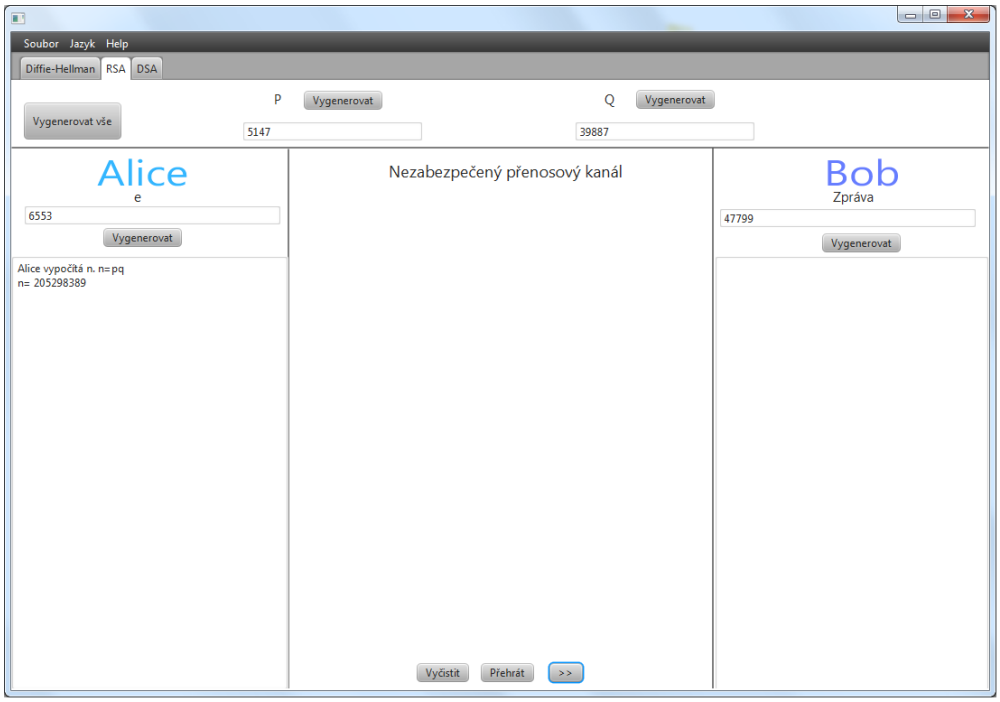

Obr. B.7: Alice dešifruje.




Obr. B.8: Chybová hláška v případě nesprávných vstupních hodnot vstupu.

# C UKÁZKA MODULU PRO DSA

Dále je v programu obsažen modul pro DSA. Na obrázkuC.1 je ukázka již hotového výpočtu. Pro četnost kroků zde není uveden postup jako u ostatních protokolů.




Obr. C.1: Ukázka DSA