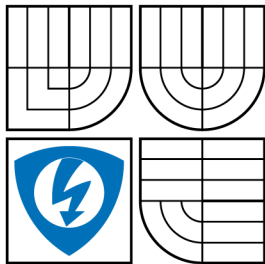


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ



FACULTY OF ELECTRICAL ENGINEERING AND
COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

NÁVRH ZABEZPEČENÍ SYSTÉMU DÁLKOVÉHO MĚŘENÍ KVALITY DODÁVKY ELEKTRICKÉ ENERGIE

SECURITY DESIGN FOR SYSTEM OF REMOTE QUALITY MEASUREMENT OF
ELECTRIC POWER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MICHAL JAKUBÍČEK

VEDOUcí PRÁCE
SUPERVISOR

Ing. PETR MLÝNEK, Ph.D.

BRNO 2013



VYSOKÉ UČENÍ
TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student: Michal Jakubiček

ID: 125232

Ročník: 3

Akademický rok: 2012/2013

NÁZEV TÉMATU:

Návrh zabezpečení systému dálkového měření kvality dodávky elektrické energie

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte protokol s veřejným klíčem pro zabezpečení komunikace v energetice, zaměřte se pouze na ustanovení klíčů komunikujících stran. Realizujte navržený protokol pro ustanovení klíčů komunikujících stran v jazyce C. Velikost klíčů musí odpovídat normám. Implementace by měla být přizpůsobena pro nízkoenergetické procesory (nevyužívat knihovny pro práci s velkými čísly, ale využít optimalizaci). Zaměřte se především na asymetrickou kryptografii, RSA a Diffie-Hellman a eliptické křivky.

DOPORUČENÁ LITERATURA:

[1] Menezes, A. J., van Oorschot, P. C., Vanstone, S. A.: Handbook of Applied Cryptography, CRC Press, 1997, ISBN 0-8493-8523-7.

[2] Herout, P.; Učebnice jazyka C, Kopp 2001, České Budějovice, s. 280, ISBN: 80-7232-220-6

Termín zadání: 11.2.2013

Termín odevzdání: 5.6.2013

Vedoucí práce: Ing. Petr Mlýnek, Ph.D.

Konzultanti bakalářské práce:

prof. Ing. Kamil Vrba, CSc.

Předseda oborové rady

UPOZORNĚNÍ:

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

ABSTRAKT

Bakalářská práce v první kapitole objasňuje pojem kvalita elektrické energie a některé její parametry. Dále jsou uvedeny používané systémy dálkového sběru dat a aktuální směr vývinu těchto systémů.

Ve třetí kapitole jsou popsány symetrické a asymetrické kryptografické systémy a jejich hlavní představitelé. Z asymetrických systémů je popsán princip výpočtu RSA a Diffie–Hellmana.

Čtvrtá kapitola se zabývá vybranými knihovnami, které umí počítat s velkými čísly. Je provedena realizace ustavení klíčů pomocí Diffie–Hellmana s těmito knihovnami a následné měření, zda optimalizovaná knihovna je rychlejší, než ostatní knihovny.

Na závěr je provedeno zhodnocení použitých knihoven z hlediska jejich rychlosti a paměťové náročnosti.

V příloze je také návod na instalaci knihoven BigDigits, GMP a OpenSSL ve Visual Studiu 2010.

KLÍČOVÁ SLOVA

Kvalita elektrické energie, kryptografie, asymetrický kryptosystém, Diffie–Hellmanův protokol, BigDigits, GMP, OpenSSL.

ABSTRACT

Bachelor thesis in the first chapter deals with the concept of quality of electric power and its parameters. The next chapter is focused on systems that used remote data collection and the current direction of evolution of these systems.

The third of the thesis discusses the symmetric and asymmetric cryptographic systems and their main representatives. The asymmetric system is described by the principle of calculating RSA and Diffie–Hellman.

The next part deals with selected libraries that can count large numbers. Through the establishment of key using the Diffie–Hellman with these libraries and the subsequent measurement should be proved whether optimized library is faster than the other libraries. The evaluation of the used library in terms of speed and memory consumption is included in the final chapter.

The appendix include installation instructions for the libraries BigDigits, GMP and OpenSSL in Visual Studio 2010.

KEYWORDS

Quality of electric power, cryptography, asymmetric cryptosystem, Diffie–Hellmann protocol, BigDigits, GMP, OpenSSL.

JAKUBÍČEK, Michal *Návrh zabezpečení systému dálkového měření kvality dodávky elektrické energie*: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2013. 42 s. Vedoucí práce byl Ing. Petr Mlýnek, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Návrh zabezpečení systému dálkového měření kvality dodávky elektrické energie“ 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

Děkuji vedoucímu bakalářské práce Ing. Petrovi Mlýnkovi, Ph.D. za odborné vedení, konzultace, trpělivost, poskytnuté studijní materiály a podnětné návrhy k práci. Dále děkuji své rodině za podporu při tvorbě práce a také v průběhu celého studia.

Brno

.....

(podpis autora)



Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic
<http://www.six.feec.vutbr.cz>

PODĚKOVÁNÍ

Výzkum popsáný v této bakalářské práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno

.....

(podpis autora)



EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI



OBSAH

Úvod	11
1 Systémy dálkového měření kvality elektrické energie	12
1.1 Kvalita elektrické energie	12
1.2 Systémy dálkového sběru dat	13
1.3 Chytrá síť	15
2 Zabezpečení přenosu	16
2.1 Kryptografická ochrana	16
3 Kryptografie	17
3.1 Symetrické kryptosystémy	19
3.1.1 AES (Advanced Encryption Standard)	19
3.2 Asymetrické kryptosystémy	20
3.2.1 RSA	20
3.2.2 Diffie–Hellmanův protokol	21
3.2.3 Kryptosystémy na bázi eliptických křivek	23
4 Volba knihoven pro práci s velkými čísly	24
4.1 Knihovna BigDigits	24
4.2 Knihovna GMP	24
4.3 OpenSSL	25
5 Realizace kryptosystému pro výměnu klíčů	26
5.1 Popis kódu	26
5.2 Srovnání knihoven	27
6 Závěr	30
Literatura	31
Seznam příloh	33
A Knihovny	34
A.1 BigDigits	35
A.1.1 Použité funkce	35
A.1.2 Postup při instalaci knihovny BigDigits ve Visual Studiu 2010	36
A.2 GMP	36

A.2.1	Použité funkce	36
A.2.2	Postup při instalaci knihovny GMP ve Visual Studiu 2010 . . .	36
A.3	OpenSSL	37
A.3.1	Použité funkce	37
A.3.2	Postup při instalaci knihovny OpenSSL ve Visual Studiu 2010	37
B	Zdrojový kód	38
B.1	BigDigits	38
B.2	GMP	39
B.3	OpenSSL	41

SEZNAM OBRÁZKŮ

3.1	Šifrovací systém	17
3.2	Rozdělení metod pro získání klíče [3]	18
3.3	Symetrický kryptosystém	19
3.4	Asymetrický kryptosystém	20
3.5	Ustavení klíče pomocí protokolu Diffie–Hellman	22
5.1	Výpis z konzole	27
5.2	Srovnání rychlosti výpočtu pro počet cyklů 100 000	28
5.3	Srovnání využití paměti	29

SEZNAM TABULEK

5.1	Srovnání rychlosti výpočtu v [h:m:s]	28
5.2	Srovnání maximálního využití paměti v kB	28

ÚVOD

Informace a data nejrůznějšího druhu jsou dnes např. v oblasti informačních technologií velmi důležité, proto se také při jejich přenosu klade velký důraz na dostatečné zabezpečení. Pro přenos přes nezabezpečený kanál se používá různých druhů šifrování. Snažíme se data šifrovat tak, aby útočník nemohl zachycenou komunikaci převést do otevřené podoby, nebo alespoň aby mu to nebylo v reálném čase umožněno. Proto jsou dnešní používané kryptografické systémy založeny na tom, že neoprávněná osoba nemůže v rozumném čase zašifrovaná data převést na dnes běžných počítačích do srozumitelné podoby.

Data ze systému dálkového měření kvality dodávky elektrické energie je nutné zabezpečit, jelikož jsou cenným materiálem pro distributory elektrické energie. Pomocí nich mohou předvídat dění v distribuční síti a podle toho se zachovat.

Tato práce se věnuje systémům dálkového měření kvality elektrické energie, zabezpečení komunikace a kryptografii. Praktická část je zaměřena na výměnu klíčů protokolem Diffie–Hellman. Tento protokol je implementován pomocí tří knihoven: BigDigits, GMP a OpenSSL. Dále bude provedeno srovnání rychlosti a paměťové náročnosti všech knihoven. Na závěr bude vybrána knihovna s nejvhodnějšími parametry z hlediska použití pro nízkoenergetické procesory.

1 SYSTÉMY DÁLKOVÉHO MĚŘENÍ KVALITY ELEKTRICKÉ ENERGIE

Dálkové měření se začalo vyvíjet s masivním nasazením obnovitelných zdrojů elektrické energie do její výroby. Elektrárna již nemusí být pouze velká továrna na energii, nýbrž může do sítě dodávat i pouze malé množství elektrické energie, což je případ především výroby elektřiny z obnovitelných zdrojů. Tyto elektrárny jsou obvykle decentralizované. Tok energie se tedy může s časem měnit. Taková soustava však zhoršuje kvalitu dodávané elektrické energie a je náchylná na rozsáhlý výpadek proudu (blackout), zejména při prudkých klimatických změnách.

1.1 Kvalita elektrické energie

Elektrická energie se zhoršenými parametry kvality je závažným problémem pro průmysl a firmy poskytující služby.

Množství vyrobené elektrické energie, pokud zanedbáme ztráty, se musí v každém časovém okamžiku rovnat energii spotřebované. Proto každé připojení nebo odpojení zátěže se projeví na kvalitě elektrické energie. Dále na ni mají vliv vlastnosti distribuční sítě a v nemalé míře i počasí.

Na straně spotřeby je však třeba zajistit určitou kvalitu elektrické energie charakterizovanou zejména velikostí napětí a jeho frekvencí, neboť i menší změna kteréhokoli z těchto parametrů může vyvolat chybnou funkci nebo dokonce poškození připojeného elektrického spotřebiče. Proto musí regulační mechanismy soustavy neustále udržovat co nejmenší odchylky velikosti napětí a frekvence od jmenovitých hodnot [2].

Kvalita elektřiny je definována normou ČSN EN 50160 - Charakteristiky napětí elektrické energie dodávané z veřejné distribuční sítě [4]. Její hlavní parametry jsou:

- **Velikost napájecího napětí**

Velikost napětí je udávána jako jmenovité napětí, pro veřejnou síť nn je fázové napětí 230 V_{ef}, sdružené napětí je 400 V_{ef}.

Pro síť vn jsou jmenovitá sdružená napětí 3, 6, 10, 22 a 35 kV, pro síť vvn jsou fázová i sdružená napětí 110 kV.

- **Kmitočet napětí**

Jmenovitý kmitočet pro Českou republiku je 50 Hz, její střední hodnota musí být 50 Hz ± 1 % během 99,5 % roku.

- **Odchylky napájecího napětí**

Během každého týdne musí být za normálních podmínek 95 % průměrných

efektivních hodnot napájecího napětí v měřicích intervalech 10 minut v rozsahu $U_{jm} \pm 10 \%$ a všechny průměrné efektivní hodnoty napájecího napětí v měřicích intervalech 10 minut musí být v rozsahu $U_{jm} +10 \%/ -15 \%$. Pro vvn je stanoveno nejvyšší napětí 123 kV.

- **Flikr**

Pod pojmem Flikr myslíme rychlé viditelné blikání svítidel, i když se napětí nalézá v dovozených hodnotách. Příčinou bývají změny napětí vyvolané rychlými změnami zatížení sítě.

- **Krátkodobé poklesy napájecího napětí**

Krátkodobé poklesy napájecího napětí způsobují poruchy v distribuční síti nebo chybné instalace u odběratelů. Většina z nich trvá méně než 1 sekundu s hloubkou poklesu méně než 60% U_{jm} .

- **Krátkodobá a dlouhodobá přerušení napájecího napětí**

Přerušení napájecího napětí je stav, kdy napětí klesne pod prahovou hodnotu, která je podle normy 1% U_{jm} . Ročně se jich vyskytuje v rozsahu od několika desítek až do několika stovek. Asi 70% krátkodobých přerušení napájecího napětí má dobu trvání kratší než 1 sekunda. Roční četnost dlouhodobých poruchových přerušení napětí (delších než 3 minuty) může být menší než 10, avšak v závislosti na oblasti může dosahovat až hodnot okolo 50 [9].

- **Dočasná přepětí o síťovém kmitočtu**

Tato přepětí mezi živými vodiči a zemí vznikají při zemních poruchách.

- **Úrovně napětí signálů v napájecím napětí**

K přenosu dat můžeme využít distribuční síť. Jsou to systémy pracující do frekvence 2 kHz, a dále systémy PLC (Power Line Communication) pracující na kmitočtech přes 100 kHz. Systém hromadného dálkového ovládání (HDO), který se v ČR také používá, pracuje na kmitočtu 216,67 Hz. V případě hodnocení vlivu tohoto signálu na kvalitu napětí jsou použity meziharmonické kmitočty, z kterých počítáme průměrné hodnoty za dobu 3 sekund [10].

1.2 Systémy dálkového sběru dat

V současné době používáme pro dálkový odečet hodnot z měřičů různé způsoby. Většinou se jedná o vlastní komutační okruhy na metalickém vedení, ale čím dál častěji se objevuje i optická síť, a to nejen ta páteřní. Také se může jednat o telefonní síť nebo mobilní GSM síť. Vždy musíme dbát na zabezpečení a spolehlivost komunikace, která je odolná proti napadení, a autorizovaný přístup k těmto informacím.

- **AMR (Automatic Meter Reading)**

Systém slouží k odečtu dat o odebrané energii u zákazníka. Data jsou odesí-

lána do datové centrály, kde se dále zpracovávají. Jedná se tedy o vylepšení klasických odečtů, není zde potřeba pověřená osoba, která navštívuje dané místo a provádí odečet manuálně.

- **AMM (Automatic Meter Management)**

Tyto elektroměry dokážou data nejen odesílat, ale i přijímat. Tím se rozšíří funkčnost zařízení např. o lepší cenovou tarifikaci - flexibilní cena, připojení nebo odpojení vzdáleného odběrného místa, nastavení maximálního příkonu atd. Tyto systémy mohou nahradit sloučené systémy AMR a hromadného dálkového ovládání (HDO)

- **AMI (Advanced Metering Infrastructure)**

Systém je chytřejší než systémy AMR a AMM, protože v reálném čase dokáže řídit některé spotřebiče odběratele na základě přijatých a vyhodnocených dat z měřičů. Úspěšně tak udržuje stabilitu celé sítě.

Společnosti zabývající se distribucí elektrické energie v současné době začínají hojně využívat systémy AMM/AMR/AMI. Oblast je velmi progresivní, systémově se řeší hlavně tyto technologie:

- **Smart Metering** – jsou to měřidla, ve kterých jsou implementované nové způsoby měření. Na rozdíl od mechanických měřidel mohou mít vyšší přesnost měření, odolnost vůči poruchám a jsou rezistentní proti pozměnění hodnot měřidel ze strany zákazníků. Velkou výhodou je v podstatě neomezené množství zaznamenaných hodnot. Lze sledovat např. spotřebu energie, maximální výkon v daném období, jalovou energii, zaznamenávat kvalitativní parametry, používat složité časově rozlišené tarify nebo rovnou ukládat několikaminutové zátěžové profily a řadu dalších parametrů. Bližší informace můžeme použít pro jednotlivé služby a zefektivnit spotřebu.
- **On-line komunikace s měřidly** – systémy AMR, AMM a AMI umožňují připojení on-line měřidel, a tím komunikaci mezi nimi. Tímto se otevírá možnost řady aplikací jako např. vzdálený odečet spotřeby, modifikace měřičů, změny tarifů, dálkové vypínání či zapínání dodávky elektrické energie při řízení spotřeby v síti (klimatizace, topení, ohřev vody). Při obousměrné komunikaci můžeme spojit i bezpečnostní systémy, vzdálené řízení pro inteligentní domy, topení nebo klimatizace.
- **Datové sklady a analýza dat** – díky tomu, že jsou dostupná přesná data ze všech odběrných míst, můžeme tato data uchovávat v datových skladech. Z nich jsme schopni následně vytvářet analýzy, které pomáhají při optimalizaci distribuční sítě. Můžeme tak předpovědět spotřebu elektrické energie v čase, identifikovat černé odběry, poruchy v síti apod.
- **Mobilní technologie** – velmi pomáhají pracovníkům při orientaci v terénu.

System GPS pomáhá najít nejen přesnou lokaci distribučních míst a transformátorů, ale také sledování vlastních pracovníků. Nyní již existují standardy pro odečítání a správu měřicích přístrojů, kalibraci a další funkce smart meteringu.

Tyto technologie umožňují sběr různorodých dat, která odečítáme z AMM systémů a smart měřidel z velkého počtu měřicích míst. Máme tak možnost získat informace o efektivním řízení dodávky elektrické energie v požadované kvalitě a určení problémových míst. Tato analýza se však v praxi používá celkem málo, především v předávacích místech. Je problém s ovládním tisíců až milionů měřidel, musíme automaticky reagovat na události a řídit spolehlivě a bezpečně celou komunikaci. Uplatňují se zde nástroje pro definici pravidel, musí být schopny zavést určité požadavky a podmínky konkrétního distributora.

1.3 Chytrá síť

Problém s dodávkou elektrické energie v požadované kvalitě lze vyřešit a zároveň uspořit nezanedbatelné množství energie pomocí chytré sítě neboli Smart Grids. Jde o typ elektrické a komunikační sítě, která v reálném čase umožňuje regulovat výrobu a spotřebu elektrické energie. Její rozsah může být jak v celosvětovém, tak i pouze v místním měřítku.

Používá se hlavně pro řízení vyrobené elektrické energie od menších dodavatelů: z větrných elektráren, fotovoltaických panelů, kogeneračních jednotek a dalších tak, aby nedocházelo k přetížení sítě. V případě nedostatku elektrické energie se vypínají zbytečné elektrické spotřebiče, naopak při přebytku mohou tyto spotřebiče odebírat elektřinu za výhodnější tarif, a tím sníží náklady za energii. Chytrá síť určuje pomocí získaných dat z různých míst rozvodné sítě aktuální stav, ihned jej vyhodnotí a podle něj řídí dodávku elektrické energie. K tomu využívá další prostředky, jako jsou tzv. smart meters, tedy „chytrá měřidla“.

2 ZABEZPEČENÍ PŘENOSU

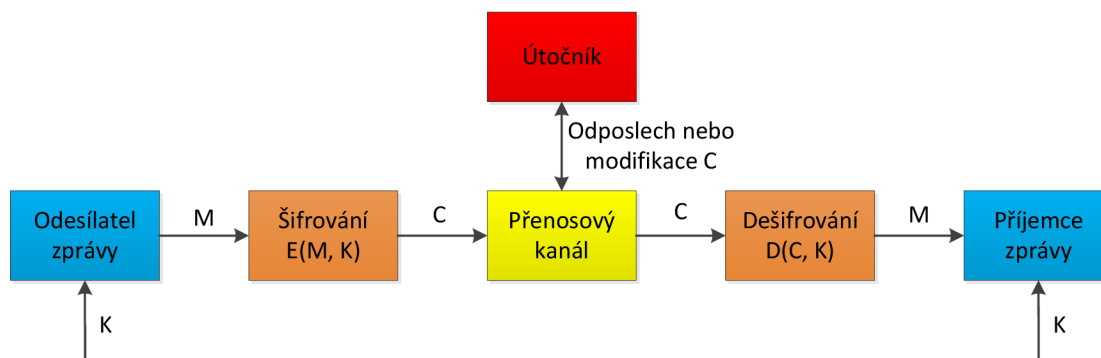
Data, která chceme poslat přes přenosový kanál, musíme určitým způsobem chránit, aby nemohlo dojít ke zneužití - neoprávněnému čtení nebo úpravě dat. Zejména v rádiovém, ale i metalickém komunikačním kanálu se musíme účinně bránit proti úmyslnému rušení či odposlouchávání kanálu. Pokud nedokážeme ochránit komunikační kanál přímo fyzicky, je vhodné použít jiný druh ochrany, např. kryptografickou ochranu.

2.1 Kryptografická ochrana

Pro bezpečnost informačních systémů je důležitá především kryptografická ochrana. Můžeme ji považovat za určitou technickou ochranu dat, která je založena na obtížnosti řešení matematických problémů. Je to nejrozšířenější druh ochrany v informačních systémech, protože její implementace je poměrně snadná, levná a má vysokou odolnost proti útokům. Nemůžeme však pomocí ní zajistit dostupnost daného zařízení [3].

3 KRYPTOGRAFIE

Kryptografie se zabývá tím, že zprávu M (message) převádí do šifrované zprávy C (cypher). Je to tedy nauka o metodách utajování smyslu zpráv převodem do podoby, která je čitelná jen se speciální znalostí. Je zde charakteristické to, že přístup ke zprávě je umožněn na základě vyřešení nějakého matematického problému, který je volen tak, aby ho útočník nemohl být schopen vyřešit za rozumnou dobu [3].



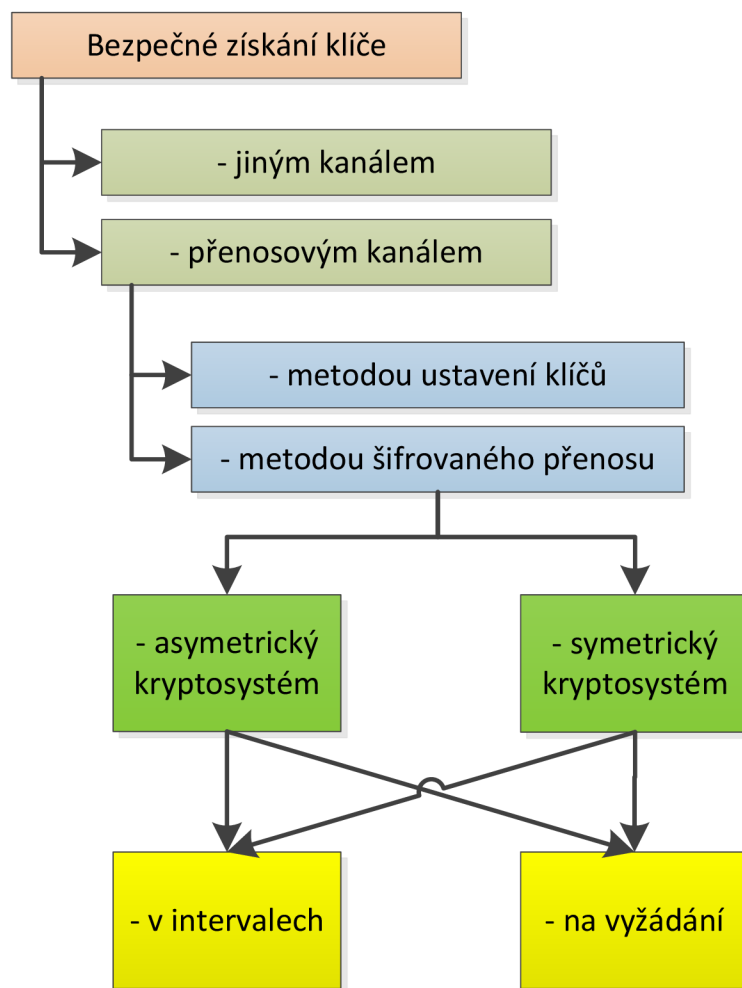
Obr. 3.1: Šifrovací systém

Kryptoanalýza je opačný postup. V kryptoanalýze se snažíme ze zašifrované zprávy C dostat zprávu M . Obvykle se zde používá hrubá síla – vyzkouší se všechny možnosti, popř. se pomůže nějakým algoritmem, protože rychlosti současných počítačů jsou již poměrně vysoké.

Kryptologie je disciplína, která sjednocuje kryptografii a kryptoanalýzu.

V literatuře o kryptografii se obvykle nacházejí tři osoby: Alice, Bob a proradná Eva, která chce číst cizí zprávy. Alice chce poslat Bobovi zprávu, a to tak, aby si ji nikdo jiný nemohl přečíst. Je možné dvojí řešení. Buď si Alice s Bobem předem domluví systém, jak se bude šifrovat, zde mluvíme o tzv. symetrické kryptografii, nebo si to předem nedomluví, a přesto chtějí šifrovat, což je v dnešní době častější. Tento systém nazýváme asymetrický.

Nejdůležitější položkou v kryptografii je klíč, což je jistá zpráva nebo informace, kterou má Alice s Bobem k dispozici. Pomocí klíče může zprávu šifrovat nebo přeložit do srozumitelné podoby. V moderní kryptografii se udává délka klíče v bitech či bajtech – ta určuje úroveň zabezpečení. Délka klíče volená pro určitou úroveň zabezpečení také závisí na použitém algoritmu šifrování [11].



Obr. 3.2: Rozdělení metod pro získání klíče [3]

Moderní kryptografie zajišťuje následující bezpečnostní cíle:

- Důvěrnost dat – utajení informace před neoprávněnými uživateli. Používá se např. při řízení fyzického přístupu k datům nebo při kryptografických metodách, kdy se data převedou do nesrozumitelné podoby šifrováním.
- Integrita dat – takové zajištění dat, aby je nemohl neoprávněný uživatel úmyslně či neúmyslně pozměnit.
- Autentizace entit – ověření pravosti daného uživatele, počítače, zařízení, programu, či procesu.
- Autentizace dat – ověření pravosti dat (obsah, čas, původ dat).
- Nepopiratelnost – zajišťuje, že daný subjekt nemůže později popřít, co vykonal [8].

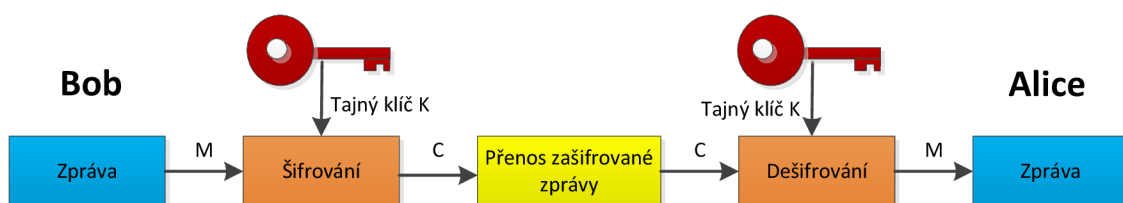
3.1 Symetrické kryptosystémy

U symetrického šifrování se šifrovací a dešifrovací klíč shoduje, nebo se z šifrovacího klíče dá snadno odvodit klíč dešifrovací. Odesílatel a příjemce zprávy tudíž musí klíče držet v tajnosti a bezpečí alespoň takovém, v jakém drží data posílaná ve zprávách. Při vyzrazení klíče by mohl kdokoli dešifrovat jejich zprávy a zjistit tak jejich obsah, anebo by mohl zprávy dešifrovat, pozměnit jejich obsah a opět zašifrovat, a příjemce by tak obdržel nepravou zprávu. Toto šifrování je časově a výpočetně málo náročné, avšak nastává zde problém s bezpečnou distribucí klíčů mezi odesílatelem a příjemcem.

Označme šifrovací funkci jako E_K , dešifrovací funkci jako D_K , volnou zprávu M , šifrovanou zprávu C za použití klíče K , potom bude šifrování reprezentovat rovnice č. 3.1 a dešifrování získání původní zprávy rovnice č. 3.2.

$$E_K(M) = C, \quad (3.1)$$

$$D_K(C) = M. \quad (3.2)$$



Obr. 3.3: Symetrický kryptosystém

Symetrické kryptosystémy se dělí na proudové a blokové šifry. Proudové šifry mění hodnotu příslušného bitu zprávy v závislosti na hodnotě klíče. Oproti tomu blokové šifry mění hodnotu příslušného bitu v závislosti na hodnotě klíče a navíc i v závislosti na dalších bitech dané zprávy. Proto je blokové šifrování obecně bezpečnější.

3.1.1 AES (Advanced Encryption Standard)

AES je velmi rozšířený symetrický blokový kryptosystém. Jeho původní název Rijndael vznikl ze jmen jejich autorů Daemena a Rijmena. V roce 2001 byl schválen Americkým úřadem pro standardizaci jako nejvhodnější návrh z patnácti různých systémů. Jeho předchůdcem je dnes již prolomený kryptosystém DES a 3DES. Má

pevně danou velikost bloku na 128 bitů, délka šifrovacího klíče je 128, 192 nebo 256 bitů.

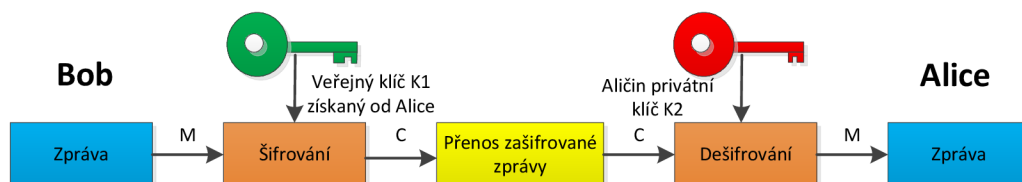
Používá se především pro šifrování větších objemů dat díky jeho malé výpočetní náročnosti. Problémem je však výměna klíčů, protože např. uložit klíč přímo do daného zařízení není bezpečné [3].

3.2 Asymetrické kryptosystémy

Asymetrické šifrování používá dva odlišné klíče. Pro šifrování zprávy veřejný klíč, který je všem známý, a pro dešifrování klíč tajný s tím, že se z veřejného klíče nedá odvodit v reálném čase. To znamená, že zprávu může zašifrovat a poslat kdokoli, ale pouze osoba vlastnící tajný klíč může tuto zprávu dešifrovat a přečíst. Tímto je také zajištěna důvěrnost a integrita přenesené zprávy [1]. Používá se i opačný případ, kdy zprávu zašifrujeme tajným klíčem a dešifrujeme veřejným klíčem. Takto si zprávu může přečíst každý, ale máme jistotu o autentičnosti přenesené zprávy (tzv. digitální podpis). Šifrování reprezentuje rovnice č. 3.3 a dešifrování rovnice č. 3.4.

$$E_{K_1}(M) = C \quad (3.3)$$

$$D_{K_2}(C) = M. \quad (3.4)$$



Obr. 3.4: Asymetrický kryptosystém

Při šifrování veřejným klíčem používáme tzv. jednosměrnou funkci. Jednosměrná funkce je taková, která se jedním směrem provede snadno a rychle, opačným směrem se provádí obtížně. Této funkci pouze věříme, že je jednosměrná, protože u většiny těchto funkcí zatím nebylo prokázáno, že jsou skutečně jednosměrné.

3.2.1 RSA

Asymetrický kryptosystém RSA byl v roce 1977 navržen Rivestem, Shamirem a Adlemanem, z jejichž iniciálů se skládá zkratka RSA. Je založen na problematice faktORIZACE čísla n – rozkladu čísla n na součin mocnin prvočísel. V opačném pořadí (při

šifrování) násobení dvou velkých čísel není příliš složité. V současnosti se používá n o velikosti 768, 1024, 2048 a 3072 bitů. Nevýhoda je v tom, že musíme pracovat s velkými čísly, proto je tento systém poměrně náročný na výpočet, a tudíž pomalý. Další slabina je v tom, že není jisté, jak dlouho bude ještě vyhovovat klíč dané délky, předpokládá se totiž, že dojde k prolomení klíčů postupně od jejich nejkratší délky.

Sestavení kryptosystému:

1. Určíme si dvě velká prvočísla p a q .
2. Vypočítáme čísla $n = (p \cdot q)$, $r = (p - 1) \cdot (q - 1)$.
3. Zvolíme veřejný šifrovací klíč e . Toto číslo musí být nesoudělné s číslem r .
4. Vypočítáme tajný šifrovací klíč $d = e^{-1} \bmod r$.
5. Parametry e a n zveřejníme. Ostatní parametry jsou tajné.

Postup šifrování:

1. Zprávu Z rozdělíme na bloky symbolů o stejné délce. Každý i -tý blok zprávy se chápe jako číslo z_i . Musí platit, že $z_i < n$.
2. Každý blok zprávy z_i zašifrujeme: $c_i = z_i^e \bmod n$.
3. Z bloků c_i poskládáme kryptogram C a odešleme jej k adresátovi.

Postup dešifrování:

1. Kryptogram C rozdělíme na původní bloky c_i .
2. Každý blok kryptogramu dešifrujeme $z_i = c_i^d \bmod n$.
3. Z bloků z_i poskládáme zprávu Z [3].

Pokud použijeme čísla d a e opačně, jedná se o jednu z možností digitálního podpisu. Kvůli již zmíněné nízké provozní rychlosti tohoto systému se proto používá hlavně pro autentizaci formou elektronického podpisu.

3.2.2 Diffie–Hellmanův protokol

Diffie–Hellmanův protokol slouží k distribuci vytvořeného klíče přes nezabezpečený přenosový kanál. Jeho vlastností je, že i když bude útočník odposlouchávat veškerou komunikaci, není schopen určit, jaký klíč byl ustaven.

Sestavení Diffie–Hellmanova protokolu k ustavení klíčů (odesílatel A – Alice, příjemce B – Bob):

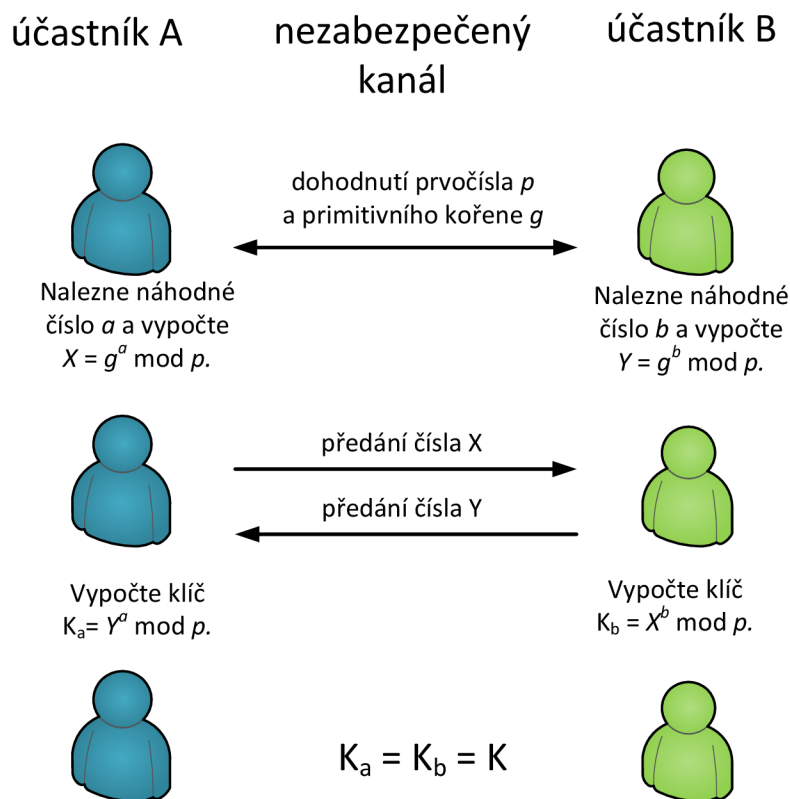
1. Zvolíme si veřejné parametry: velké prvočíslu p a primitivní kořen g , který je definován tak, že všechna čísla $y = (g^x) \bmod p$ jsou pro $x = 1, 2, \dots, (p - 1)$ různá.
2. Odesílatel zvolí náhodně velké číslo a a příjemce zvolí náhodně velké číslo b . Tato čísla jsou tajná.
3. Účastník A vypočítá číslo $X = g^a \bmod p$. Podobně účastník B vypočítá číslo $Y = g^b \bmod p$. Vypočítaná čísla si účastníci přenosovým kanálem navzájem

předají.

- Účastník A vypočítá klíč $K' = Y^a \bmod p$. Účastník B vypočítá stejnou hodnotu $K' = X^b \bmod p$. Hodnota K' je stejná jako K a může být použita jako klíč pro rychlejší symetrický kryptosystém.

Skutečnost, že oba účastníci vypočítají stejný klíč, plyne z rovnice č. 3.5 [3].

$$K = Y^a \bmod p = (g^b)^a \bmod p = (g^a)^b \bmod p = X^b \bmod p = K'. \quad (3.5)$$



Obr. 3.5: Ustavení klíče pomocí protokolu Diffie–Hellman

Diffie–Hellmanův protokol využívá obtížnost řešení diskretního logaritmu. Útočník zná veřejná prvočísla p a g , dále zná vyměněné $g^a \bmod p$ a $g^b \bmod p$. Aby se dozvěděl klíč K , musí zjistit alespoň jedno z tajných čísel a nebo b . Může k tomu využít vztah $X = g^a \bmod p$ nebo $Y = g^b \bmod p$. Tento výpočet je však prakticky neřešitelný, protože pro používané hodnoty p z oboru hodnot 2^{768} až $2^{1024} \approx 10^{231}$ až 10^{308} nejsme schopni úlohu v rozumném čase vyřešit.

Při rovnosti $p = n$ kryptosystémy využívající diskretního logaritmu jsou srovnatelné s kryptosystémem RSA modulo n [3].

3.2.3 Kryptosystémy na bázi eliptických křivek

Šifrování pomocí eliptických křivek je založeno na algebraických strukturách eliptických křivek nad konečnými poli. Na toto použití přišli v roce 1985 Neal Koblitz a Victor Saul Miller.

Za hlavní výhodu eliptických křivek můžeme považovat vyšší bezpečnost použitého klíče. Mají tedy nižší výpočetní náročnost než klasické kryptosystémy, a tím postačí menší nároky na hardware při dodržení odpovídající bezpečnosti jako u klasických kryptosystémů. Další výhodou je v tom, že systémy založené na obtížném počítání diskretního logaritmu jsou aplikovatelné také pomocí eliptických křivek, tím se staly jejich výhodnou alternativou. Nicméně v současné době ještě nejsou eliptické kryptosystémy masově v praxi využívány. Jednou možnou příčinou může být využívání starých kryptosystémů, které mají mnohem delší historický vývoj než ono šifrování pomocí eliptických křivek [7].

4 VOLBA KNIHOVEN PRO PRÁCI S VELKÝMI ČÍSLY

Pro realizaci výměny klíčů po otevřeném kanále byl zvolen protokol Diffie-Hellman, podle normy [6] smí být použit pro výměnu klíčů v energetice. Problém zde nastává při použití velkých čísel, protože programovací jazyky mají velikostně omezené datové typy. Největší datový typ např. v jazyce C je `unsigned long long`, který má 64 bitů, a tím je pro kryptografické výpočty nevyhovující. Pokud chceme pracovat s většími typy, musejí se spojit jednotlivé `unsigned long long` nebo jiné datové typy a napsat funkce pro práci s nimi. Jednodušší je použít již existující knihovny, které mají open source kódy (kromě GMP, který má licenci GNU).

Jsou tedy použity knihovny, které mají vlastní datové typy, obvykle libovolné velikosti, a umí s nimi vykonávat různé aritmetické a jiné matematické operace. Byly zvoleny knihovny BigDigits, GMP a OpenSSL.

4.1 Knihovna BigDigits

Jedná se o volnou knihovnu psanou v ANSI C. Používá se pro provádění výpočtů velkých přirozených čísel, kterých je potřeba např. v kryptografických výpočtech. Knihovna obsahuje funkce pro klasické aritmetické operace (sčítání, odčítání, násobení a dělení) a také funkce pro modulární násobení, umocnění, inverzi čísla či ověření, zda je dané velké číslo prvočíslo.

Knihovna má dvě rozhraní:

1. Kompletní sadu funkcí zvanou BIGD („bd“ knihovna), zde se místo v paměti alokuje automaticky.
2. Sadu základních funkcí („mp“ knihovna), se kterou je sice obtížnější pracovat, ale zato je v zásadě ve výsledku výpočetně jednodušší a má možnost vynulovat celou paměť naráz (`NO_ALLOCS`). Toto rozhraní je rychlejší oproti spoléhání se na automatickou alokaci paměti v knihovně „bd“, protože je známo dopředu, jak velká pole se mají v paměti alokovat, a nemusí se při běhu programu čekat na dynamickou alokaci[14].

Knihovna BigDigits obsahuje navíc samostatné rozhraní pro generování náhodných čísel, což může být využito při testování a ukázkových programech.

4.2 Knihovna GMP

GMP je volná knihovna v jazyce C a C++ pro práci s libovolně velkými čísly, a tedy i pro práci s libovolně velkou přesností, omezení je pouze v dostupné paměti

zařízení. Má velkou sadu funkcí a funkce mají svá rozhraní. Velkým plusem je vysoká výkonnost při manipulaci s velkými čísly. Je distribuována pod licencí GNU LGPL (Leader General Public Licence), knihovnu lze tedy volně používat, sdílet a vylepšovat. Hlavní cílenou platformou pro použití jsou systémy typu Unix, jako GNU/Linux, Solaris, Mac OS X, atd. Běží také pod 32 i 64bitovými Windows [15].

4.3 OpenSSL

OpenSSL obsahuje knihovny s otevřeným zdrojovým kódem psané v jazyce C. Může se používat pro komerční i nekomerční účely s výhradou jistých jednoduchých podmínek – tzv. Apache licence. Má v sobě funkce pro základní kryptografické operace. Používá dynamickou alokaci paměti. Základní objekt je BIGNUM, jeho velikost není teoreticky omezená [16].

5 REALIZACE KRYPTOSYSTÉMU PRO VÝMĚNU KLÍČŮ

5.1 Popis kódu

Byl realizován program pro výměnu klíče pomocí protokolu Diffie–Hellmann s využitím tří knihoven pracujících s velkými čísly viz. Příloha B. Vstupní proměnné a , b , p a g jsou pevně zadané v kódu programu. V praxi se využívá (pseudo)náhodné generování těchto čísel, na což má knihovna BigDigits funkci `rand()` ve své knihovně `bigdRand.h`, GMP knihovna zase funkci `mpz_random` a knihovna OpenSSL obsahuje svou knihovnu `rand.h` s funkcí `RAND`.

Byla napsána optimalizace funkce `multiplyModulo` s knihovnou BigDigits. Postup byl dle následujícího pseudokódu z webové stránky z literatury č. [13]:

```
function powermod(base, exponent, modulus) {
  if (base < 1 || exponent < 0 || modulus < 1)
    return -1

  result = 1;
  while (exponent > 0) {
    if ((exponent % 2) == 1) {
      result = (result * base) % modulus;
    }
    base = (base * base) % modulus;
    exponent = floor(exponent / 2);
  }
  return result;
}
```

Knihovny GMP a OpenSSL využívají při vypočítávání této funkce již napsanou funkci `mpz_powm` viz. příloha B.2 a `BN_mod_exp` viz. B.3.

Výstupem každého kódu je vidět na obr. 5.1, tedy vstupní proměnné a , b , p a g , dále z nich vypočítané hodnoty proměnných X a Y podle rovnice č. 3.5. Nakonec se vypíší získané klíče K_a a K_b , které se musejí rovnat.

Klíče jsou tvořeny o velikosti 1024 bitů a prvočíslo má velikost 160 bitů, čísla (viz. příloha A) jsou pro kontrolu převzaty ze standardu RFC 5114 [12]. Tento standard popisuje skupiny Diffie–Hellmana, které mohou být použity ve spojení

```
C:\Users\Michal\Documents\Visual Studio 2010\Projects\DH_GMP\Release\DH_GMP.exe

Diffie-Hellmanuv protokol pro vmenu klice
pomoci knihovny GMP

Michal Jakubicek - Bakalarska prace

A = b9a3b3ae8fefc1a2930496507086f8455d48943e
B = 9392c9f9eb6a7a6a9022f7d83e7223c6835hbdda
P = b10b8f96a080e01dde92de5eae5d54ec52c99fbcfb06a3c69a6a9dca52d23b616073e28675a2
3d189838ef1e2ee652c013ecb4aea906112324975c3cd49b83bfaccbdd7d90c4bd7098488e9c219a
73724effd6fae5644738faa31a4ff55bcc0a151af5f0dc8b4bd45bf37df365c1a65e68cfda76d4d
a708df1fb2bc2e4a4371
G = a4d1cbd5c3fd34126765a442efb99905f8104dd258ac507fd6406cff14266d31266fea1e5c41
564b777e690f5504f213160217b4b01b886a5e91547f9e2749f4d7fbd7d3b9a92ee1909d0d2263f8
0a76a6a24c087a091f531dbf0a0167b6a28ad662a4d10e73af32d779d5918d00bc8858f4dcef97c
2a24855e6eeb22b3b2e5
X = 2a853b3d92197501b9015b2deb3ed84f5e021dcc3e52f109d3273d2b7521281cbabe0e76ff57
27fa8acce26956ba9a1fca26f20228d8693feb10841d84a7360054ece5a7f5b7a61ad3dfb3c60d2e
43106d8727da37df9cce95b478755d06bcea8f9d45965f75a5f3d1df3701165fc9e50c4279ceb07f
989540ae96d5d88ed776
Y = 717a6cb053371ff4a3b932941c1e5663f861a1d6ad34ae66576dfb98f6c6cbf9ddd5a56c7833
f6bcbdfdf095582ad868e440e8d09fd769e3c9ccdc3d3b1e4cfa057776caaf9739b6a9fee8e7411f8
d6dac09d6a4edb46cc2b5d5203090eae6126311e53fd2c14b574e6a3109a3da1be41bdceaa186f5c
e06716a2b6a07b3c33fe
Ka = 5c804f454d30d9c4df85271f93528c91df6b48ab5f80b3b59caac1b28f8acha9cd3e39f3cb6
14525d9521d2e644c53b807b810f340062f257d7d6fbfe8d5e8f072e9b6e9afda9413eafb2e8b069
9b1fb5a0caceddeaead7e9cfbb36ae2b420835bd83a19fb0b5e96bf8fa4d09e345525167ecd91554
16f46f408ed31b63c6e6d
Kb = 5c804f454d30d9c4df85271f93528c91df6b48ab5f80b3b59caac1b28f8acha9cd3e39f3cb6
14525d9521d2e644c53b807b810f340062f257d7d6fbfe8d5e8f072e9b6e9afda9413eafb2e8b069
9b1fb5a0caceddeaead7e9cfbb36ae2b420835bd83a19fb0b5e96bf8fa4d09e345525167ecd91554
16f46f408ed31b63c6e6d
```

Obr. 5.1: Výpis z konzole

s IETF protokoly k zabezpečení komunikace na internetu. Různé skupiny Diffie-Hellmana se používají k různé kryptografické účinnosti, tj. délky základních prvočísel používaných během procesu výměny klíčů.

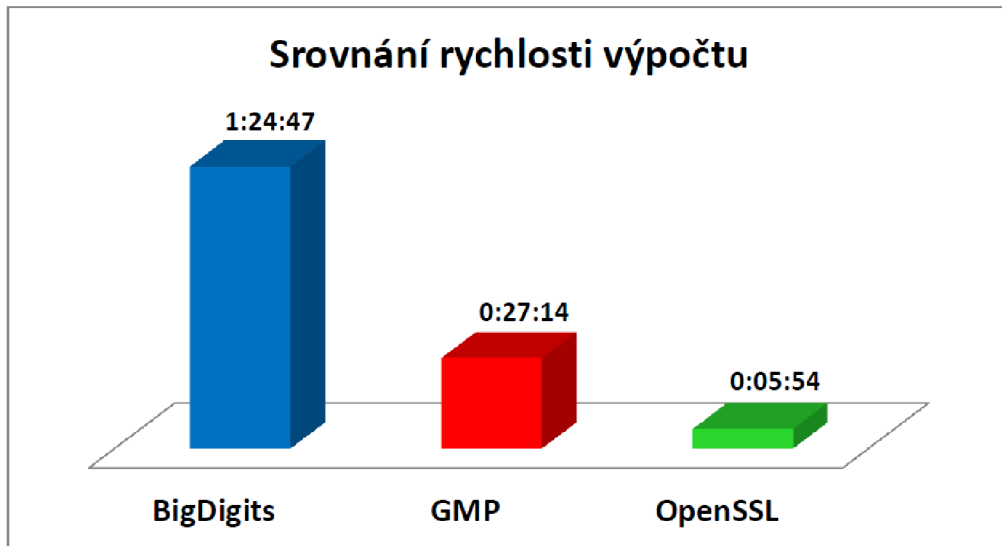
5.2 Srovnání knihoven

Srovnání knihoven bylo provedeno z hlediska rychlosti výpočtu v závislosti na počtu vykonaných cyklů. To bylo realizováno pomocí cyklu `for`, který vykonával celý program od začátku do konce. Bylo zvoleno 1000, 10 000 a 100 000 takovýchto cyklů, aby byly časové hodnoty jednoduše měřitelné. Pro zjištění měřených hodnot byl použit program Správce úloh v systému Windows 7. Při spuštění smyčky se začal počítat čas procesoru a běžel až do dokončení všech cyklů. Zjištěné údaje z měření rychlosti jsou uvedeny v tabulce č. 5.1.

Z obrázku č. 5.2 vidíme srovnání jednotlivých knihoven z hlediska rychlosti výpočtu. V knihovně BigDigits byla vytvořena vlastní funkce *MultiplyModulo*, která

Tab. 5.1: Srovnání rychlosti výpočtu v [h:m:s]

Počet cyklů	BigDigits	GMP	OpenSSL
1 000	0:00:50	0:00:16	0:00:03
10 000	0:08:29	0:02:43	0:00:35
100 000	1:24:47	0:27:14	0:05:54



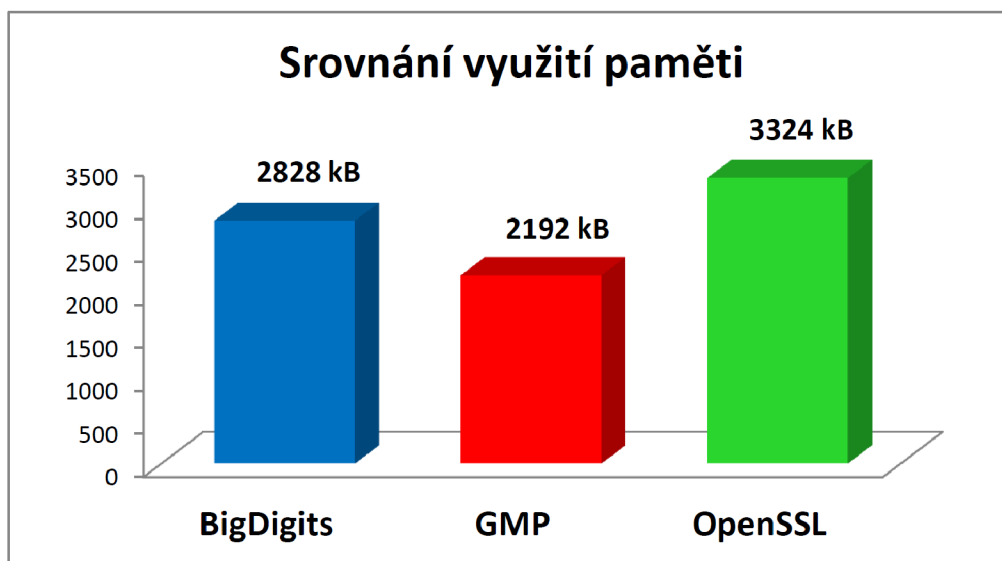
Obr. 5.2: Srovnání rychlosti výpočtu pro počet cyklů 100 000

měla optimalizovat rychlost výpočtu použitím knihovny „mp“. Ta má omezenou sadu funkcí a je rychlejší na výpočet. I přesto je nakonec algoritmus s knihovnou BigDigits nejpomalejší ze všech testovaných knihoven. Vidíme, že hlavně knihovna OpenSSL, ale i GMP jsou již pro dané výpočty velmi optimalizované. OpenSSL spočítala stejné výsledky přibližně patnáctkrát rychleji než BigDigits a více než čtyřikrát rychleji než GMP.

Tab. 5.2: Srovnání maximálního využití paměti v kB

Počet cyklů	BigDigits	GMP	OpenSSL
1 000	2848 kB	2188 kB	3352 kB
10 000	2852 kB	2184 kB	3348 kB
100 000	2828 kB	2192 kB	3324 kB

Dále bylo uskutečněno srovnání z hlediska paměťové náročnosti. V tomto případě byl brán v potaz pouze vrchol pracovní sady – tzn. nejvyšší využitá paměť při běhu programu. Tabulka č. 5.2 obsahuje toto srovnání – paměťovou náročnost algoritmu



Obr. 5.3: Srovnání využití paměti

pro každou použitou knihovnu.

Z obrázku č. 5.3 vidíme, že nejrychlejší knihovna zabírá také nejvíce paměti. Proto, při omezené operační paměti, může být knihovna GMP výhodným řešením, neboť potřebuje ke své funkci v našem případě o 1 MB méně. Zde se opět projevila optimalizace knihovny GMP.

6 ZÁVĚR

V této bakalářské práci jsou popsány základní parametry pro stanovení kvality elektrické energie. Je zřejmé, že na kvalitu může působit více faktorů a není vždy jednoduché určit danou příčinu.

Systémy AMM/AMR/AMI pro dálkový sběr dat působí v energetice jako nástroj pro zjednodušení získání dat z měřičů a pro optimalizování rovnováhy rozvodné soustavy.

Hlavní část práce se zabývá kryptografií, symetrickou a asymetrickou šifrou, zvláště pak Diffie–Hellmanovým protokolem pro ustavení klíčů.

Tento protokol je realizován prakticky v jazyce C se třemi knihovnamí pracujícími s velkými čísly: BigDigits, GMP a OpenSSL. V knihovně BigDigits je napsána vlastní funkce pro optimalizaci operace umocnění a modulo. Bylo provedeno srovnání všech tří knihoven z hlediska rychlosti provedení výpočtu, dále bylo změřeno maximální využití paměti. Nejrychlejší knihovna se ukázala být OpenSSL, která byla více než čtyřikrát rychlejší než GMP a téměř patnáctkrát rychlejší než BigDigits. Naproti tomu se GMP knihovna ukázala jako paměťově nejúspornější, oproti OpenSSL potřebovala o 1 MB paměti méně. Pro menší paměťovou náročnost je tedy doporučována knihovna GMP. Pokud je kladen důraz spíše na rychlost výpočtu, je doporučována knihovna OpenSSL.

Optimalizace knihovny BigDigits se ukázala jako nevýhodná oproti již optimalizovaným knihovnám OpenSSL a GMP.

LITERATURA

- [1] BAYER, T. Návrh hardwarového šifrovacího modulu. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 62 s.
- [2] BLAŽEK, Vladimír a Petr SKALA. Distribuce elektrické energie. Skriptum VUT FEKT v Brně, 140 s.
- [3] BURDA, K. Bezpečnost informačních systémů. Skripta FEKT VUT v Brně, 2005.
- [4] ČSN EN 50160. Charakteristiky napětí elektrické energie dodávané z veřejné distribuční sítě. Praha: Český normalizační institut, 2011. 32 s.
- [5] FORMAN, T. Portál pro podporu výuky kryptografie. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2010.
- [6] IEC 62351-1, Power systems management and associated information exchange – Data and communications security – Part 3: Communication network and system security – Profiles TCP/IP.
- [7] OCHODKOVÁ, Eliška. Přínos teorie eliptických křivek k řešení moderních kryptografických systémů. VŠB - TUO [online]. 2003 [cit. 2013-05-25]. Dostupné z: http://www.cs.vsb.cz/arg/workshop/files/ecc_eli.pdf
- [8] KLÍMA, V. Moderní kryptografie I. CRYPTO-WORLD [online]. 11. 4. 2007, 4, [cit. 2012-11-05]. Dostupné z: http://www.cryptoworld.info/klima/mffuk/Symetricka_kryptografie_I_2007.pdf
- [9] Parametry elektřiny: vlastnosti výrobku. E.ON: O energii [online]. [cit. 2012-11-05]. Dostupné z: <http://www.eon.cz/cs/info/parameters.shtml>
- [10] ŠENK, M. Měření kvality elektrické energie . Brno : Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2010. 50 s.
- [11] TRUNEČEK P. Kryptografické protokoly v praxi. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 48 s.
- [12] RFC 5114 - Additional Diffie-Hellman Groups for Use with IETF Standards. *IETF Tools* [online]. 2008 [cit. 2013-04-20]. Dostupné z: <http://tools.ietf.org/html/rfc5114>
- [13] Modulus power of big numbers: Some pseudo code. *Stack Overflow* [online]. 2011 [cit. 2013-05-18]. Dostupné z: <http://stackoverflow.com/questions/8287144/modulus-power-of-big-numbers>

- [14] BigDigits multiple-precision arithmetic source code. *DI Management Home Page* [online]. 2001 [cit. 2013-04-21]. Dostupné z: <http://di-mgt.com.au/bigdigits.html>
- [15] The GNU MP Bignum Library. *GMP* [online]. 2000 [cit. 2013-04-12]. Dostupné z: <http://gmplib.org>
- [16] OpenSSL: Documents, bn(3). *OpenSSL: The Open Source toolkit for SSL/TLS* [online]. 1999 [cit. 2013-04-16]. Dostupné z: <http://www.openssl.org/docs/crypto/bn.html>

SEZNAM PŘÍLOH

A	Knihovny	34
A.1	BigDigits	35
A.1.1	Použité funkce	35
A.1.2	Postup při instalaci knihovny BigDigits ve Visual Studiu 2010	36
A.2	GMP	36
A.2.1	Použité funkce	36
A.2.2	Postup při instalaci knihovny GMP ve Visual Studiu 2010 . .	36
A.3	OpenSSL	37
A.3.1	Použité funkce	37
A.3.2	Postup při instalaci knihovny OpenSSL ve Visual Studiu 2010	37
B	Zdrojový kód	38
B.1	BigDigits	38
B.2	GMP	39
B.3	OpenSSL	41

A KNIHOVNY

V kódu ve všech třech případech byly použity stejné vstupní proměnné v hexa tvaru:

A =

0xB9A3B3AE 8FEFC1A2 93049650 7086F845 5D48943E,

B =

0x9392C9F9 EB6A7A6A 9022F7D8 3E7223C6 835BBDDA,

P =

0xB10B8F96 A080E01D DE92DE5E AE5D54EC 52C99FBC FB06A3C6 9A6A9DCA 52D23B61
6073E286 75A23D18 9838EF1E 2EE652C0 13ECB4AE A9061123 24975C3C D49B83BF
ACCBDD7D 90C4BD70 98488E9C 219A7372 4EFFD6FA E5644738 FAA31A4F F55BCCCO
A151AF5F ODC8B4BD 45BF37DF 365C1A65 E68CFDA7 6D4DA708 DF1FB2BC 2E4A4371,
G =

0xA4D1CBD5 C3FD3412 6765A442 EFB99905 F8104DD2 58AC507F D6406CFF 14266D31
266FEA1E 5C41564B 777E690F 5504F213 160217B4 B01B886A 5E91547F 9E2749F4
D7FBD7D3 B9A92EE1 909D0D22 63F80A76 A6A24C08 7A091F53 1DBFOA01 69B6A28A
D662A4D1 8E73AFA3 2D779D59 18D08BC8 858F4DCE F97C2A24 855E6EEB 22B3B2E5.

Vypočítají se čísla X a Y:

X =

0x2A853B3D 92197501 B9015B2D EB3ED84F 5E021DCC 3E52F109 D3273D2B 7521281C
BABE0E76 FF5727FA 8ACCE269 56BA9A1F CA26F202 28D8693F EB10841D 84A73600
54ECE5A7 F5B7A61A D3DFB3C6 OD2E4310 6D8727DA 37DF9CCE 95B47875 5D06BCEA
8F9D4596 5F75A5F3 D1DF3701 165FC9E5 0C4279CE B07F9895 40AE96D5 D88ED776

Y =

0x717A6CB0 53371FF4 A3B93294 1C1E5663 F861A1D6 AD34AE66 576DFB98 F6C6CBF9
DDD5A56C 7833F6BC FDF0955 82AD868E 440E8D09 FD769E3C ECCDC3D3 B1E4CFA0
57776CAA F9739B6A 9FEE8E74 11F8D6DA C09D6A4E DB46CC2B 5D520309 0EAE6126
311E53FD 2C14B574 E6A3109A 3DA1BE41 BDCEAA18 6F5CE067 16A2B6A0 7B3C33FE.

Výstupem jsou klíče:

Ka =

0x5C804F45 4D30D9C4 DF85271F 93528C91 DF6B48AB 5F80B3B5 9CAAC1B2 8F8ACBA9
CD3E39F3 CB614525 D9521D2E 644C53B8 07B810F3 40062F25 7D7D6FBF E8D5E8F0
72E9B6E9 AFDA9413 EAFB2E8B 0699B1FB 5A0CATED DEAEAD7E 9CFBB36A E2B42083
5BD83A19 FB0B5E96 BF8FA4D0 9E345525 167ECD91 55416F46 F408ED31 B63C6E6D

a Kb =

0x5C804F45 4D30D9C4 DF85271F 93528C91 DF6B48AB 5F80B3B5 9CAAC1B2 8F8ACBA9
CD3E39F3 CB614525 D9521D2E 644C53B8 07B810F3 40062F25 7D7D6FBF E8D5E8F0

72E9B6E9 AFDA9413 EAFB2E8B 0699B1FB 5A0CACED DEAEAD7E 9CFBB36A E2B42083
5BD83A19 FB0B5E96 BF8FA4D0 9E345525 167ECD91 55416F46 F408ED31 B63C6E6D.

A.1 BigDigits

A.1.1 Použité funkce

- **int multiplyModulo(DIGIT_T output[], DIGIT_T base[], DIGIT_T exponent[], DIGIT_T modulo[])** – vlastní funkce, která počítá $output = base^{exponent} \bmod modulo$. Algoritmus byl převzat z pseudokódu ze stránky [13].
- **void mpSetEqual(DIGIT_T a[], const DIGIT_T b[], size_t ndigits)** – nastaví $a = b$
- **size_t mpConvFromDecimal** – překonvertuje řetězec v dekadickém tvaru do proměnné datového typu DIGIT_T
- **size_t mpConvFromHex (DIGIT_T a[], size_t ndigits, const char *s)** – překonvertuje řetězec v hexadecimálním tvaru do proměnné datového typu DIGITS_T
- **int mpIsZero(const DIGIT_T a[], size_t ndigits)** – vrátí hodnotu true když je $a == 0$, jinak vrátí hodnotu false
- **int mpModulo(DIGIT_T r[], const DIGIT_T u[], size_t udigits, DIGIT_T v[], size_t vdigits)** – spočítá $r = u \bmod v$, přičemž u je velké u -čísel a v je velké v -čísel
- **int mpEqual(const DIGIT_T a[], const DIGIT_T b[], size_t ndigits)** – vrátí hodnotu true když je $a == b$, jinak vrátí hodnotu false
- **int mpMultiply(DIGIT_T w[], const DIGIT_T u[], const DIGIT_T v[], size_t ndigits)** – spočte součin $w = u * v$, proměnné u a v jsou n -místné a velikost proměnné w je $2 * n$ -místná.
- **int mpSquare(DIGIT_T w[], const DIGIT_T x[], size_t ndigits)** – spočítá druhou mocnin $w = x^2$
- **DIGIT_T mpShiftRight(DIGIT_T a[], const DIGIT_T b[], size_t x, size_t ndigits)** – spočítá bitovou operaci $a = b \gg x$
- **void mpPrintHex(const char *prefix, const DIGIT_T *p, size_t len, const char *suffix)** – tisk v hexa formátu s volitelnými předponami či příponami k řetězci

A.1.2 Postup při instalaci knihovny BigDigits ve Visual Studiu 2010

- Založíme nový projekt, Win32 Console Application v jazyce C (C++), v Application Settings odznačíme Precompiled header, nakopírujeme do něj kód.
- Knihovnu stáhneme z adresy

<http://www.di-mgt.com.au/bigdigits.html#download>

a rozbalíme zip archiv, obsah knihovny nakopírujeme do složky projektu.

- V této složce označíme soubory bigd.h, bigd.c, bigdigits.h a bigdigits.c (popř. i bigdigitsRand.c a bigdigitsRand.h) a přetáhneme je do Visual Studia do našeho projektu.

A.2 GMP

A.2.1 Použité funkce

- **void mpz_init (mpz_t x)** – inicializace proměnné x datového typu `mpz_t` s nastavenou hodnotou 0
- **int mpz_set_str (mpz_t rop, char *str, int base)** – funkce, která nastaví hodnotu řetězce do proměnné `rop`, do `base` vložíme hodnotu základu dané číselné soustavy řetězce (desítková, šestnáctková)
- **char * mpz_get_str (char *str, int base, mpz_t op)** – funkce, která konvertuje data z `op` na řetězec, `base` určuje základ číselné soustavy
- **void mpz_powm_ui (mpz_t rop, mpz_t base, unsigned long int exp, mpz_t mod)** – vypočítá $rop = base^{exp} \bmod mod$
- **void mpz_clear (MP_INT *integer)** – uvolnění dynamicky alokované proměnné na konci programu

A.2.2 Postup při instalaci knihovny GMP ve Visual Studiu 2010

- Založíme nový projekt, Win32 Console Application v jazyce C (C++), v Application Settings odznačíme Precompiled header, nakopírujeme do něj kód.
- Knihovnu si stáhneme pomocí odkazu

<http://www.cs.nyu.edu/exact/core/gmp/gmp-dynamic-mingw-4.1.tar.gz>

a rozbalíme do složky projektu. Používám dynamickou knihovnu, protože statická mi nefungovala s žádným návodem.

- Ve vlastnostech projektu v konfiguračních vlastnostech ve VC++ Directories zadáme cestu pro Include Directories i pro Library Directories „\gmp-dynamic“, poté rozklikneme Linker, kolonku Input a zde přidáme do Additional Dependencies „gmp.lib“.
- Dále musíme přímo do složky projektu zkopírovat knihovnu gmp.dll a při spuštění exe souboru s ním musí být vždy ve stejné složce.

A.3 OpenSSL

A.3.1 Použité funkce

- `BN_CTX *BN_CTX_new(void)` – alokace a inicializace struktury `BN_CTX`, která uchovává dočasně proměnné užívané funkcemi knihovny
- `BIGNUM *BN_new(void)` – alokace a inicializace nové struktury `BIGNUM`
- `int BN_hex2bn(BIGNUM **a, const char *str)` – převede řetězec `str` obsahující hexadecimální číslo na `BIGNUM`
- `int BN_bn2hex(const BIGNUM *a)` – navrátí tisknutelný řetězec představující hexadecimální číslo
- `int BN_mod_exp(BIGNUM *r, BIGNUM *a, const BIGNUM *p, const BIGNUM *m, BN_CTX *ctx)` – vypočítá $r = a^p \bmod m$
- `void BN_free(BIGNUM *a)` – uvolnění dynamicky alokované proměnné na konci programu
- `void BN_CTX_free(BN_CTX *c)` – uvolnění pomocné proměnné `ctx`

A.3.2 Postup při instalaci knihovny OpenSSL ve Visual Studiu 2010

- Založíme nový projekt, Win32 Console Application v jazyce C (C++), v Application Settings odznačíme Precompiled header, nakopírujeme do něj kód.
- Knihovnu si stáhneme z odkazu

http://slproweb.com/download/Win32OpenSSL-1_0_1e.exe

a nainstalujeme do složky projektu

- Ve vlastnostech projektu v konfiguračních vlastnostech ve VC++ Directories zadáme cestu pro Include Directories „\OpenSSL-Win32\include“ a do Library Directories zadáme „\OpenSSL-Win32\lib“, poté rozklikneme Linker, kolonku Input a zde přidáme do Additional Dependencies "libeay32.lib".

B ZDROJOVÝ KÓD

B.1 BigDigits

```
#include <stdio.h>
#include "BigDigits\bigd.h"
#include "BigDigits\bigdigits.h"

#define KLIC 32

int multiplyModulo(DIGIT_T output[], DIGIT_T base[], DIGIT_T exponent[],
DIGIT_T modulo[])
{
    DIGIT_T exponentMod[KLIC], exponentModOutput[KLIC], two[1], one[KLIC],
    outputOutput[2*KLIC], baseMod[KLIC], baseModOutput[2*KLIC];

    mpSetEqual(exponentMod, exponent, KLIC);
    mpSetEqual(baseMod, base, KLIC);

    mpConvFromDecimal(output, KLIC, "1");
    mpConvFromDecimal(one, KLIC, "1");
    mpConvFromDecimal(two, 1, "2");

    while (!mpIsZero(exponentMod, KLIC)) {
        mpModulo(exponentModOutput, exponentMod, KLIC, two, 1);

        if (mpEqual(exponentModOutput, one, 1)) {
            mpMultiply(outputOutput, output, baseMod, KLIC);
            mpModulo(output, outputOutput, 2*KLIC, modulo, KLIC);
        }

        mpSquare(baseModOutput, baseMod, KLIC);
        mpModulo(baseMod, baseModOutput, 2*KLIC, modulo, KLIC);

        mpShiftRight(exponentModOutput, exponentMod, 1, KLIC);
        mpSetEqual(exponentMod, exponentModOutput, KLIC);
    }

    return 1;
}

int main(int argc, const char * argv[])
{
    printf("      -----\n");
    printf(" | Diffie-Hellmanuv protokol pro vymenu klice |\n");
    printf(" |           pomoci knihovny BigDigits           |\n");
}
```



```

int main(int argc, char *argv[])
{
    mpz_t a, b, p, g, X, Y, Ka, Kb;

    mpz_init(a);
    mpz_init(b);
    mpz_init(p);
    mpz_init(g);
    mpz_init(X);
    mpz_init(Y);
    mpz_init(Ka);
    mpz_init(Kb);

    printf("
-----\n");
    printf(" | Diffie-Hellmanuv protokol pro vymenu klice |\n");
    printf(" |           pomoci knihovny GMP           |\n");
    printf(" |                                           |\n");
    printf(" |   Michal Jakubicek - Bakalarska prace   |\n");
    printf(" |_____|\n\n");

    mpz_set_str(a, "B9A3B3AE8FEFC1A2930496507086F8455D48943E", 16);
    printf("A = %s\n\n", mpz_get_str(NULL, 16, a));

    mpz_set_str(b, "9392C9F9EB6A7A6A9022F7D83E7223C6835BBDDA", 16);
    printf("B = %s\n\n", mpz_get_str(NULL, 16, b));

    mpz_set_str(p, "B10B8F96A080E01DDE92DE5EAE5D54EC52C99FBCFB06A3C69A6A9DC
A52D23B616073E28675A23D189838EF1E2EE652C013ECB4AEA906112324975C3CD49B83
BFACCBDD7D90C4BD7098488E9C219A73724EFFD6FAE5644738FAA31A4FF55BCC0A151A
F5F0DC8B4BD45BF37DF365C1A65E68CFDA76D4DA708DF1FB2BC2E4A4371", 16);
    printf("P = %s\n\n", mpz_get_str(NULL, 16, p));

    mpz_set_str(g, "A4D1CBD5C3FD34126765A442EFB99905F8104DD258AC507FD6406CF
F14266D31266FEA1E5C41564B777E690F5504F213160217B4B01B886A5E91547F9E2749
F4D7FBD7D3B9A92EE1909D0D2263F80A76A6A24C087A091F531DBFOA0169B6A28AD662A
4D18E73AFA32D779D5918D08BC8858F4DCEF97C2A24855E6EEB22B3B2E5", 16);
    printf("G = %s\n\n", mpz_get_str(NULL, 16, g));

    mpz_powm(X, g, a, p);
    mpz_powm(Y, g, b, p);

    printf("X = %s\n\n", mpz_get_str(NULL, 16, X));
    printf("Y = %s\n\n", mpz_get_str(NULL, 16, Y));

    mpz_powm(Ka, Y, a, p);

```

```

mpz_powm(Kb, X, b, p);

printf("Ka = %s\n\n", mpz_get_str(NULL, 16, Ka));
printf("Kb = %s\n\n", mpz_get_str(NULL, 16, Kb));

mpz_clear(a);
mpz_clear(b);
mpz_clear(p);
mpz_clear(g);
mpz_clear(X);
mpz_clear(Y);
mpz_clear(Ka);
mpz_clear(Kb);

getchar();
return 0;
}

```

B.3 OpenSSL

```

#include <stdio.h>
#include <openssl/bn.h>

int main(int argc, const char * argv[])
{
    BN_CTX * ctx = BN_CTX_new();

    BIGNUM * a = BN_new();
    printf("-----\n");
    printf(" | Diffie-Hellmanuv protokol pro vymenu klice |\n");
    printf(" |           pomoci knihovny OpenSSL           |\n");
    printf(" |                                           |\n");
    printf(" |   Michal Jakubicek - Bakalarska prace   |\n");
    printf(" |-----|\n\n");

    BN_hex2bn(&a, "B9A3B3AE8FEFC1A2930496507086F8455D48943E");
    printf("A = 0x%s\n\n", BN_bn2hex(a));

    BIGNUM * b = BN_new();
    BN_hex2bn(&b, "9392C9F9EB6A7A6A9022F7D83E7223C6835BBDDA");
    printf("B = 0x%s\n\n", BN_bn2hex(b));

    BIGNUM * p = BN_new();
    BN_hex2bn(&p, "B10B8F96A080E01DDE92DE5EAE5D54EC52C99FBCFB06A3C69A6A9DCA
52D23B616073E28675A23D189838EF1E2EE652C013ECB4AEA906112324975C3CD49B83B
FACCBDD7D90C4BD7098488E9C219A73724EFFD6FAE5644738FAA31A4FF55BCCCOA151AF

```

```

05FDC8B4BD45BF37DF365C1A65E68CFDA76D4DA708DF1FB2BC2E4A4371");
printf("P = 0x%s\n\n", BN_bn2hex(p));

BIGNUM * g = BN_new();
BN_hex2bn(&g, "A4D1CBD5C3FD34126765A442EFB99905F8104DD258AC507FD6406CFF
14266D31266FEA1E5C41564B777E690F5504F213160217B4B01B886A5E91547F9E2749F
4D7FBD7D3B9A92EE1909D0D2263F80A76A6A24C087A091F531DBF0A0169B6A28AD662A4
D18E73AFA32D779D5918D08BC8858F4DCEF97C2A24855E6EEB22B3B2E5");
printf("G = 0x%s\n\n", BN_bn2hex(g));

BIGNUM * X = BN_new();
BIGNUM * Y = BN_new();

BN_mod_exp(X, g, a, p, ctx);
BN_mod_exp(Y, g, b, p, ctx);

printf("X = 0x%s\n\n", BN_bn2hex(X));
printf("Y = 0x%s\n\n", BN_bn2hex(Y));

BIGNUM * Ka = BN_new();
BIGNUM * Kb = BN_new();

BN_mod_exp(Ka, Y, a, p, ctx);
BN_mod_exp(Kb, X, b, p, ctx);

printf("Ka = 0x%s\n\n", BN_bn2hex(Ka));
printf("Kb = 0x%s\n\n", BN_bn2hex(Kb));

BN_free(a);
BN_free(b);
BN_free(p);
BN_free(g);
BN_free(X);
BN_free(Y);
BN_free(Ka);
BN_free(Kb);
BN_CTX_free(ctx);

getchar();
return 0;
}

```