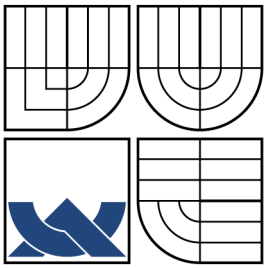
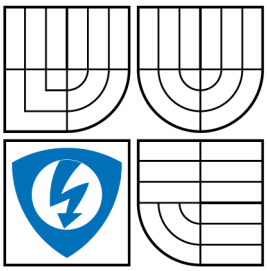


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



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

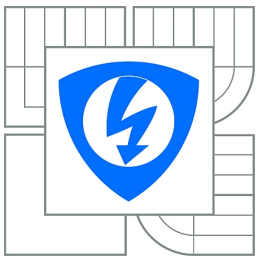
SECURITY DESIGN FOR SYSTEM OF REMOTE QUALITY MEASUREMENT OF ELECTRIC POWER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MICHAL JAKUBÍČEK
AUTHOR

VEDOUCÍ PRÁCE — Ing. PETR MLÝNEK, Ph.D.
SUPERVISOR

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Michal Jakubíček	***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:

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. Následně je provedeno zhodnocení použitých knihoven z hlediska jejich rychlosti, počtu cyklů a paměťové náročnosti.

V poslední kapitole je vysvětlena obecně optimalizace zaměřená mimo jiné na nízkovýkonové procesory a následně ukázáno Montgomeryho násobení a umocňování. Je podrobně popsán postup jeho výpočtu s popisem, proč je jeho výpočet rychlejší než klasické modulární násobení či umocňování.

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, optimalizace, Montgomeryho modulární násobení.

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. The evaluation of the used library in terms of speed, number of elementary cycles and memory consumption is inculded in the chapter 5.

The last chapter explains the optimization focused on Ultra-Low Power MCU and shown Montgomery multiplication and squaring. It describes in detail the procedure of calculation and described why is his calculation faster than conventional modular multiplication and exponentiation.

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–Hellman protocol, BigDigits, GMP, OpenSSL, optimization, Montgomery modular multiplication.

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. 49 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, Bc. Lucii Košárkové a Bc. Jakubu Onderkovi 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 popsaný 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

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

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, počtu zachycených vzorků a paměťové náročnosti všech knihoven. Bude vybrána knihovna s nejvhodnějšími parametry. Poslední část práce se zaměří na optimalizaci 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 $\pm$ 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} ± 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 dovolených 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 sítě nebo mobilní GSM sítě. 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 navštěvuje 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 elekrické 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.

Systém 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 technolgie 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 ovládáním 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

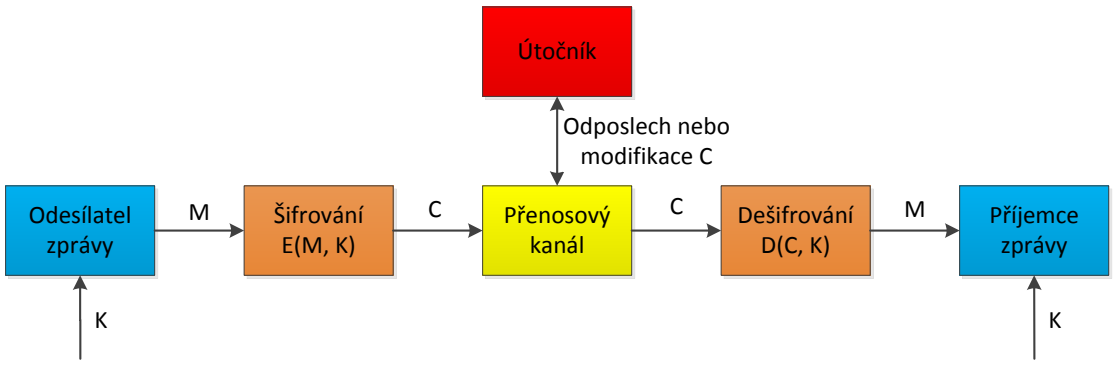
Data, která chceme poslat přes přenosový kanál, musíme určitým způsobem chránit, aby nemohlo dojít ke zneužití - neoprávněnému čtení nebo úpravě dat. Zejména v rádiovém, ale i metalickém komunikačním kanálu se musíme účinně bránit proti úmyslnému rušení či odposlouchávání kanálu. Pokud nedokážeme ochránit 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].



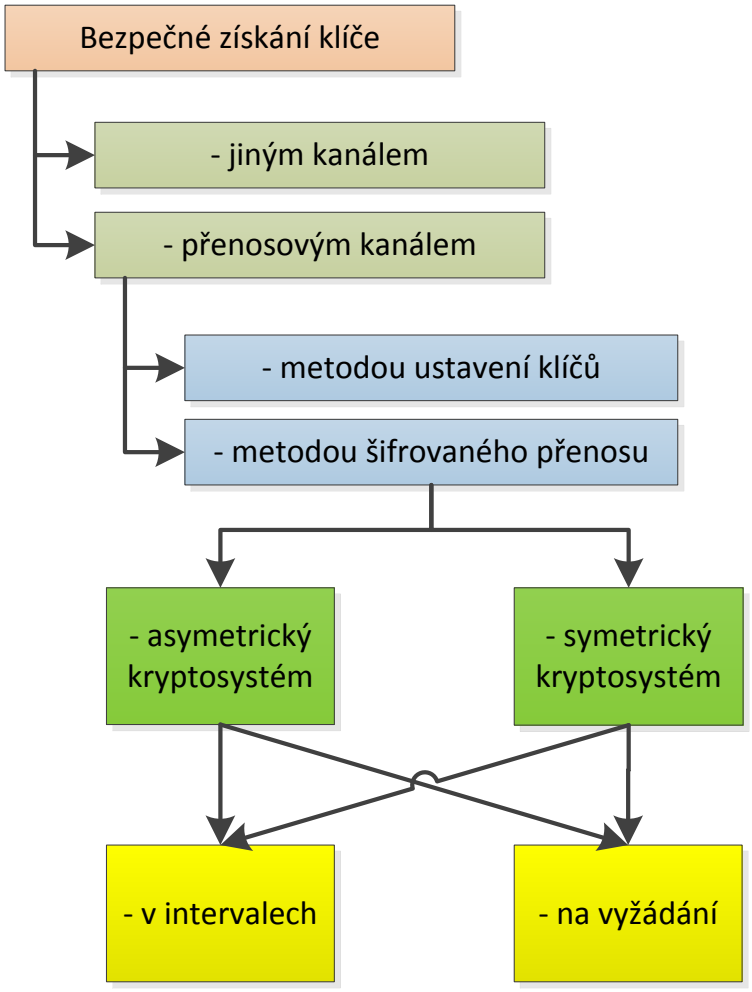
Obr. 3.1: Šifrovací systém

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].

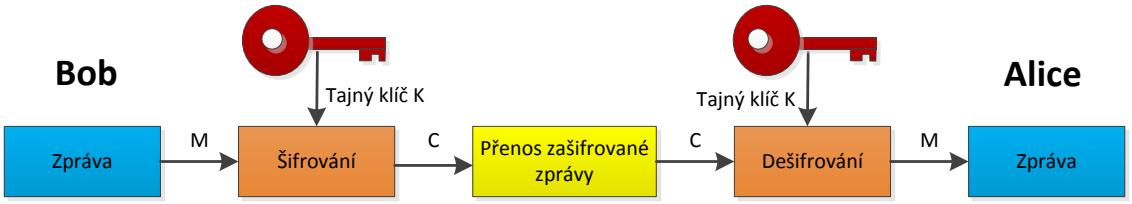
3.1 Symetrické kryptosystémy

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 kdokoliv 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.

$$E_{\mathrm{K}}(M) = C, \tag{3.1}$$

$$D_{\mathrm{K}}(C) = M. \tag{3.2}$$



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.

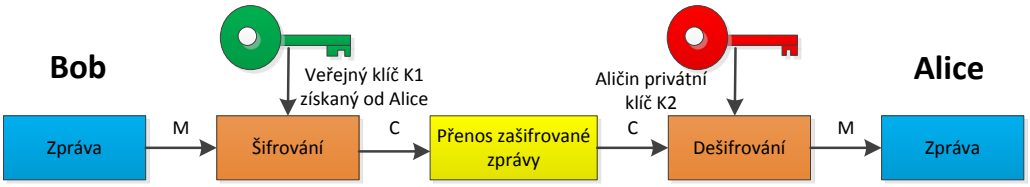
Používá se především pro šifrování větších objemů dat díky jeho malé výpočetní náročnosti. Problémem je však výměna klíčů, protože např. uložit klíč přímo do daného zařízení není bezpečné [3].

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.

$$E_{\mathrm{K1}}(M) = C \tag{3.3}$$

$$D_{\mathrm{K2}}(C) = M. \tag{3.4}$$



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 zi 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):

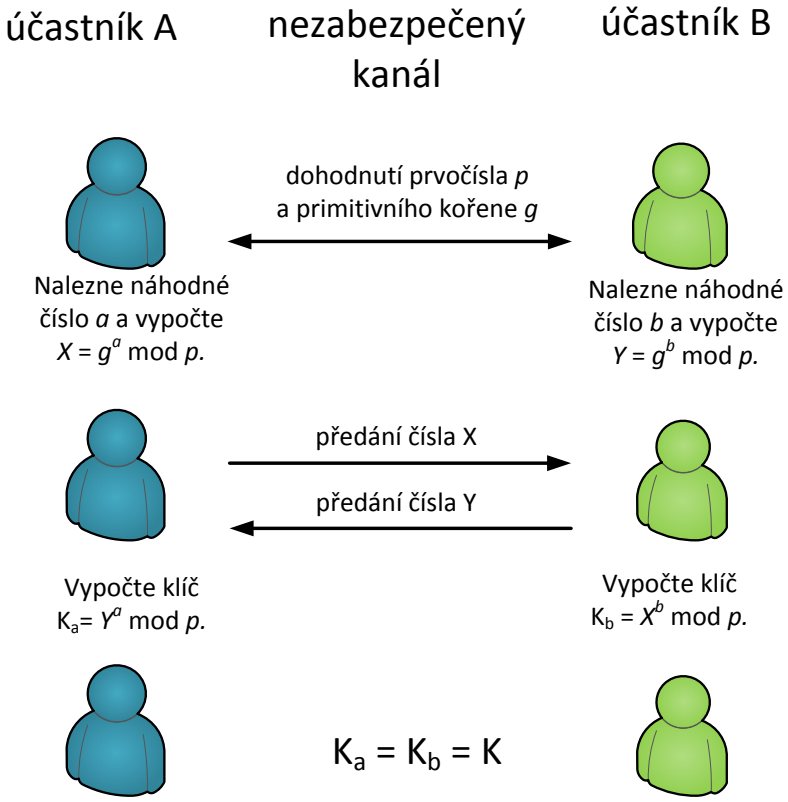
1. Zvolíme si veřejné parametry: velké prvočíslo p a primitivní kořen g, který je definován tak, že všechna čísla $y = (g^x) \bmod p$ jsou pro $x = 1, 2, \ldots, (p - 1)$ různá.
2. Odesílatel zvolí náhodně velké číslo a a příjemce zvolí náhodné velké číslo b. Tato čísla jsou tajná.
3. Účastník A vypočítá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

předají.

4. 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].

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



Obr. 3.5: Ustavení klíče pomocí protokolu Diffie–Hellman

Diffie–Hellmanův protokol využívá obtížnost řešení diskrétního 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 nerešitelný, 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 diskrétního 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 výhoda je v tom, že systémy založené na obtížném počítání diskrétního 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 otevřené zdrojové kódy (open source).

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

4.1 Knihovna BigDigits

Jedná se o otevřenou 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), u které 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 otevřená 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 množství dostupné

paměti zařízení. Má velkou sadu funkcí, které mají definované 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 je komplexní otevřená knihovna implementující různé šifrovací algoritmy psané v jazyce C. Může se používat pro komerční i nekomerční účely s výhradou jistých jednoduchých podmínek – tzv. Apache licence. Součástí OpenSSL je knihovna `bn`, která umožňuje provádět matematické operace s velkými čísly a je optimalizována pro výpočet kryptografických algoritmů. Používá dynamickou alokaci paměti. Základní objekt je BIGNUM, jehož 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()` definovanou v hlavičkovém souboru `bigdRand.h`, knihovna GMP zase funkci `mpz_random` a knihovna OpenSSL definuje v souboru `rand.h` funkci `RAND`.

Knihovny GMP a OpenSSL využívají při vypočítávání klíčů pomocí protokolu Diffie–Hellman funkci $g^a \bmod p$ existující funkci mpz_powm viz příloha B.2 resp. BN_mod_exp viz B.3.

Obr. 5.1: Výpis z konzole

Výstup každého mezivýsledku je zobrazen na obr. 5.1, tedy vstupní proměnné *a, b, p* a *g* a 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 číslem o velikosti 1024 bitů a prvočíslo má velikost 160 bitů, testovací čísla (viz příloha A) jsou pro kontrolu převzata ze standardu RFC 5114 [12]. Tento standard popisuje skupiny Diffie–Hellmana, které mohou být použity ve spojení s IETF protokoly k zabezpečení komunikace na Internetu. Různé skupiny Diffie–Hellmana se používají k různé kryptografické účinnosti, tj. délce základních prvočísel používaných během procesu výměny klíčů.

5.2 Srovnání knihoven

Srovnání bylo provedeno ve Visual Studiu 2010 pomocí Performance Explorer, a to dle Inclusive Samples využívající čítače výkonu (Performance Counters) a také dle času potřebného pro výpočet. Velikost Inclusive Samples je dána počtem odebraných vzorků, zde se zahrnuje i volání mimo danou funkci (narozdíl od Exclusive Samples). Pokud toto číslo vynásobíme vzorkovacím intervalem (např. 1 000 000), dostaneme celkový počet cyklů funkce, v našem případě funkce `main`, a tím i celého programu [17]. Poslední měřená veličina byla celková paměť využitá při běhu programu.

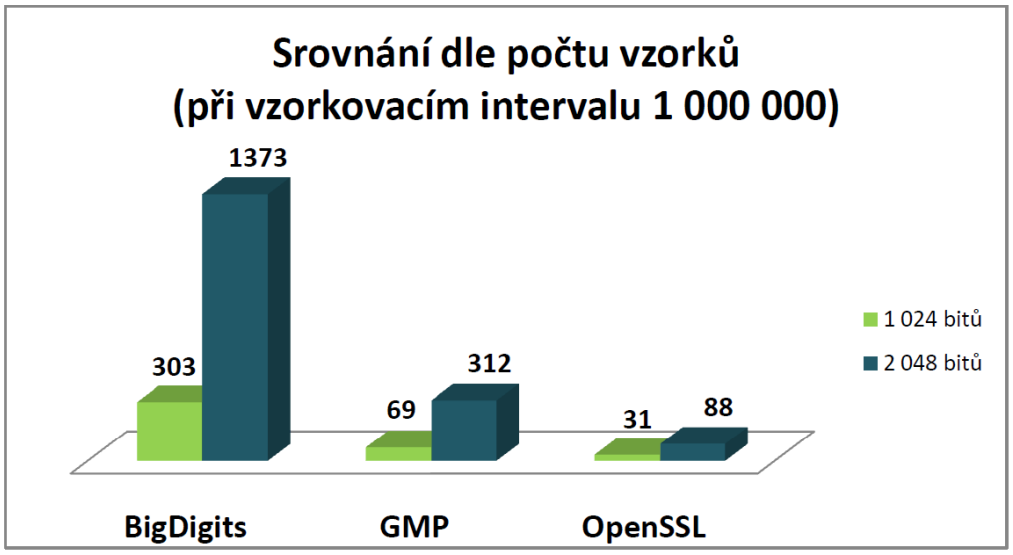
Zjištěné údaje z měření jsou uvedeny v tabulkách č. 5.1, 5.2 a 5.3.

Tab. 5.1: Počet zachycených vzorků při běhu programu

Velikost klíče	BigDigits	GMP	OpenSSL
1024	303	69	31
2048	1373	312	88

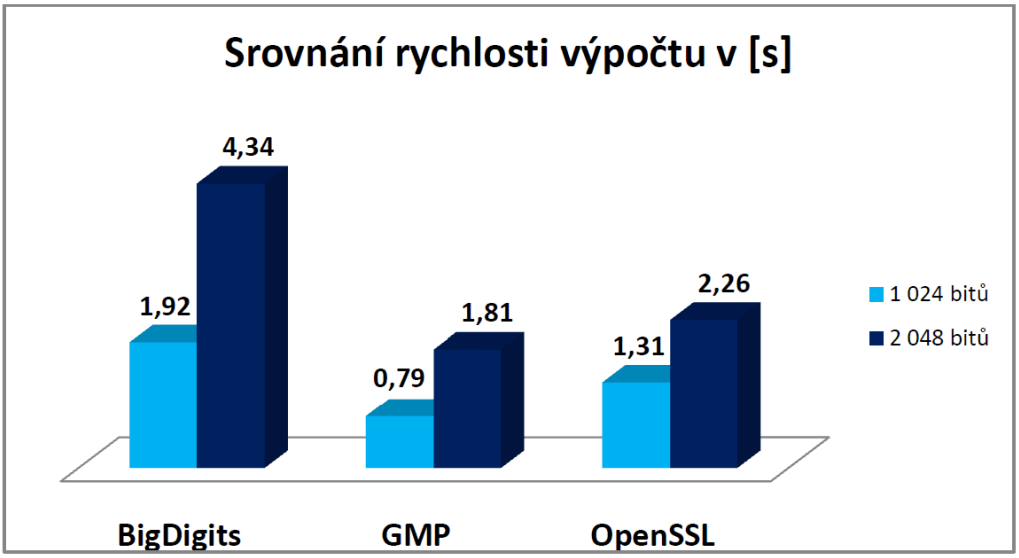
V režimu vzorkování je periodicky přerušen běh programu a odebrán vzorek, využívá se jeden z CPU čítačů. Ve výchozím nastavení je vzorkování nastaveno na 10 000 000, v našem případě byl vzorkovací interval nastaven na 1 000 000. Způsob nastavení relace pro zhodnocení rychlosti kódu: kliknout pravým tlačítkem myši na soubor v okně Performance Explorer, vybrat Properties a v záložce Sampling zvolit místo Clock cycles Performance Counter. Nyní si již můžeme zvolit vlastní vzorkovací interval. Visual Studio Profiler dokáže odebírat ve stejný čas vzorky z jediného čítače, proto můžeme využít pouze jeden čítač na jedno měření [17].

Na obrázku 5.2 vidíme srovnání jednotlivých knihoven z hlediska rychlosti výpočtu. V knihovně BigDigits byla vytvořena vlastní funkce *MultiplyModulo*, která měla zvýšit rychlost výpočtu použitím knihovny „*mp*“. Ta má omezenou sadu funkcí a je



Obr. 5.2: Srovnání knihoven podle počtu vzorků z čítače výkonu

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 desetkrát rychleji než BigDigits a více než dvakrát rychleji než GMP při počítání klíče o velikost 1024 bitů.



Obr. 5.3: Čas potřebný pro výpočet kódu

Čas potřebný pro výpočet odpovídá předchozímu měření. Avšak v tomto případě nejrychlejší byla knihovna GMP, zřejmě proto, že i když využívá více instrukcí pro výpočet, tyto instrukce jsou rychlejší, než které využívá OpenSSL.

Tab. 5.2: Srovnání knihoven podle času potřebného pro výpočet

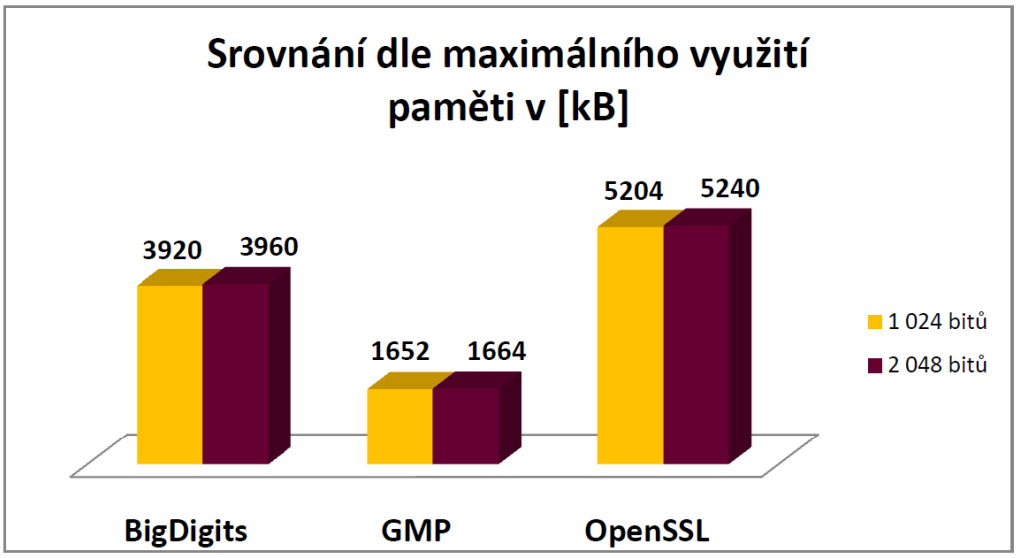
Velikost klíče	BigDigits	GMP	OpenSSL
1024	1,92	0,79	1,306
2048	4,34	1,81	2,26

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. 5.3 obsahuje toto srovnání – paměťovou náročnost algoritmu pro každou použitou knihovnu.

Měření bylo provedeno pomocí Správce úloh v systému Windows 7. Nejprve byl spuštěn program a na jeho konci pozastaven příkazem „getchar“, ve Správci úloh se v kartě Procesy odečetla hodnota ze sloupce „Vrchol pracovní sady (paměť)“. Na stránkách podpory operačního systému Microsoft Windows tento sloupec nazývají „Paměť – nejvyšší velikost pracovní sady“ s popisem: „Maximální velikost paměti pracovní sady používané procesem“ [18].

Tab. 5.3: Srovnání knihoven paměťové náročnosti

Velikost klíče	BigDigits	GMP	OpenSSL
1024	1,92	0,79	1,31
2048	4,34	1,81	2,26



Obr. 5.4: Srovnání využití paměti

Z obrázku č. 5.4 vidíme, že časově nejrychlejší knihovna GMP zabírá nejméně paměti. Knihovna s nejméně zachycenými vzorky (nejméně provedenými instrukcemi) využívá pro výpočet největší množství paměti. Proto, při omezené operační paměti, může být knihovna GMP výhodným řešením, neboť potřebuje ke své funkci v našem případě o více než 3 MB méně než OpenSSL. Zde se opět projevila optimalizace knihovny GMP.

6 OPTIMALIZACE ALGORITMU PRO NÍZKOENERGETICKÉ PROCESORY

Optimalizací myslíme zefektivnění kódu nebo snížení nároků na jeho výpočet. U nízkoenergetických procesorů je nutná, máme-li omezené výpočetní i paměťové prostředky, např. nízkoenergetický procesor MSP430 šesté série (F6xx) může pracovat maximálně na frekvenci 25 MHz, má velikost flash paměti 512 kB a RAM 67 kB. S tímto omezením není možno využít knihovny pro práci s velkými čísly, které si vyžádají v paměti i několik MB. Optimalizaci lze obecně rozdělit na:

- optimalizaci rychlosti,
- paměťové náročnosti,
- velikost zkompilovaného kódu,
- množství přístupů do paměti atd.,

s tím, že dané optimalizace mohou být vzájemně protichůdné - např. chceme-li zvýšit rychlost výpočtu, může se nám zvýšit velikost programu.

Optimalizace mohou být dále děleny podle toho, zda se zaměříme na instrukce, nebo na data programu. Pokud nevyužijeme knihovny s velkými čísly, je třeba si utvořit vlastní datovou strukturu, abychom byli schopni operovat s velkými čísly. Výhodné je vytvořit si strukturu podobnou BIGNUM z knihovny OpenSSL, která vypadá následovně:

```
{
struct bignum_st
        {
        BN_ULONG *d;    /* Pointer to an array of 'BN_BITS2'
                        bit chunks. */
        int top;        /* Index of last used d +1. */
        /* The next are internal book keeping for bn_expand. */
        int dmax;       /* Size of the d array. */
        int neg;        /* one if the number is negative */
        int flags;
        };
}.
```

`BN_ULONG` je ukazatel pole, kde se nachází velké číslo, `top` určuje pozici posledního použitého prvku pole, velikost pole je uložena do `dmax`, `neg` se použije v případě záporného čísla a `flag` je pomocná proměnná pro jiné příznaky [16].

Pokud využijeme tuto datovou strukturu (a nevyužijeme knihovnu pro práci s velkými čísly), je třeba napsat funkce pro základní operace s těmito velkými čísly, tj. sčítání, odčítání, násobení, dělení a operaci modulo.

Zaměřme se nyní na matematickou optimalizaci, tedy na optimalizaci instrukcí. Konkrétně u výpočtu vzorce $g^a \bmod p$ jde o složitou operaci. Například nízkonapěťový mikroprocesor z rodiny MSP430 (Low Voltage) nemá integrovanou hardwarovou násobičku, proto se operace násobení musí emulovat, což je jak časově tak i energeticky náročnější [19][20].

6.1 Klasické modulární násobení

U klasického modulárního násobení se jedná o metodu, jež vyžaduje jak násobení, tak i modulární redukci. Klasická metoda je ze všech nejjednodušší na pochopení, nejprve vynásobíme a a b a následně zjišťujeme zbytek po dělení.
Algoritmus klasického modulárního násobení:

- Vstup: a, b, n.
- Výstup: $X = a \cdot b \bmod n$.
- Postup:
  1. $t = a \cdot b$.
  2. $v = \lfloor t \div p \rfloor$, kde operace $\lfloor\ \rfloor$ vyjadřuje zaokrouhlení na nejbližší celé číslo, které je menší nebo rovno zaokrouhlovanému číslu.
  3. $X = t - v \cdot n$.

Nejsložitější operací je dělení ve 2. kroku, způsobuje velké zpomalení algoritmu a nehodí se tedy pro počítání s velkými čísly [21].

6.2 Montgomeryho modulární násobení

Montgomeryho násobení je metoda, která dokáže efektivně modulárně násobit bez toho, aniž by prováděla klasické modulární násobení. Jedná se o metodu, jež využívá výhod Montgomeryho redukce, což je zjednodušení algoritmu pro modulární redukci. Pro protokol Diffie–Hellman, kde máme základ g stále stejný, je provedeno předpočítání prvků r a r^{-1} pro různé exponenty a ty již zůstávají pro všechny výpočty stejné.

Mějme celá čísla a a b, přirozená čísla n a r. Číslo r je zvoleno tak, aby se dalo v dané číselné soustavě snadno dělit, tzn. ve dvojkové soustavě bude $r = 2^k$, v desítkové $r = 10^k$. Čísla musejí být zvolena tak, aby platilo $r > n$ a $\gcd(n, r) = 1$.[1] V Montgomeryho násobení je nutné převést vzor a na reziuum A a vzor b na reziduum B, to lze vypočítat jako:

$$A = (a \cdot r) \bmod n;\ B = (b \cdot r) \bmod n.$$

[1] gcd - největší společný dělitel

Z tohoto rezidua lze opět zpátky vypočítat vzor a a vzor b jako:

$$a = (A \cdot r^{-1}) \bmod n;\ b = (B \cdot r^{-1}) \bmod n,$$

kde r^{-1} je inverzní číslo k r v modulu n, to lze vypočítat pomocí rozšířeného Euklidova algoritmu. Montgomeryho násobení dvou reziduí A a B:

$$A \otimes B = A \cdot B \cdot r^{-1}.$$

V modulu n platí

$$C = A \otimes B = A \cdot B \cdot r^{-1} = (a \cdot r) \cdot (b \cdot r) \cdot r^{-1} = (a \cdot b) \cdot r = c \cdot r$$

a toto c je výsledkem modulárního součinu $c = (a \cdot b) \bmod n$. Dále je pro výpočet potřebná proměnná m, pro kterou platí

$$r \cdot r^{-1} - n \cdot m = 1,$$

lze ji tedy vypočítat jako

$$m = \frac{r \cdot r^{-1} - 1}{n}.$$

Algoritmus pro výpočet Montgomeryho modulárního násobení:

- Vstup: A, B, n, r, m.
- Výstup: C.
- Postup:
  1. $t = A \cdot B$.
  2. $u = (t \cdot m) \bmod r$.
  3. $C = (t + u \cdot n)/r$ [21].

Operace modulo ve 2. bodu a operace dělení ve 3. bodu se stává velmi snadnou, při operaci modulo je výsledkem posledních k číslic a u dělení dělenec po odebrání jeho posledních k číslic. Tento algoritmus provádí operace s čísly jako celky. Existují i jiné varianty, kde se provádějí operace po bitech [21].

Jediná knihovna, která Montgomeryho násobení využívá, je OpenSSL ve funkci `BN_mod_mul_montgomery`; pro převod z a do Montgomeryho reziduí slouží funkce `BN_from_montgomery` a `BN_to_montgomery`. Knihovna GMP obsahuje pouze funkce pro tento převod. V knihovně BigDigits tvůrci v minulosti experimentovali s Montgomeryho umocňováním pro rychlejší výpočet funkce `ModExp`, ale skončili s výsledkem o málo lepším nebo stejným než při počítání pomocí klasického modulárního umocňování. Kód se navíc stal více složitý a nepřehledný, proto od této metody nakonec ustoupili.

6.3 Montgomeryho modulární násobení s bitovým posunem

Následující algoritmus je založen na bitovém posunu, což patří mezi jednu z nejrychlejších operací. Montgomeryho rezidua A a B jsou uloženy v binárním tvaru, mohou být zapsána jako $A = \sum_i A_i 2^i$ a $B = \sum_i B_i 2^i$. Když počítáme $A \cdot B \cdot r^{-1}$, využíváme k-krát bitový posun doprava (číslo k může být v našem případě 1024). Protože číslo n je liché, může při operaci děleno 2 mod n nastat dvojí situace:

1. vznikne sudé číslo, které může být rovnou bitově posunuto,
2. vznikne číslo liché a potom je třeba přičíst n, aby byl poslední bit nulový a tím se číslo mohlo bitově posunout.

Pro proces násobení je třeba proměnná P pro zapamatování aktuálního výsledku, $P = \sum_i P_i 2^i$. Každý digit čísla B_i se násobí $\sum_i A_i 2^i$ a následně dělí 2. Když bude na začátku $B_i = 0$, bude i $P = 0$ (tyto začáteční hodnoty můžeme zahodit). Začínáme počítat až s bitem i_0, tj. až bude $B_{i_0} = 1$, potom přesuneme hodnotu A_i do P_i.

Od pozice i_0 je postup následující podle toho, zda je $B_i = 0$ nebo 1:

1. $B_i = 0 \rightarrow$ podělíme P 2 a přičteme n, pokud je P liché,
2. $B_i = 1 \rightarrow$ přidáme hodnotu A do P a to následně podělíme 2 a případně přičteme n, pokud je P liché.

Kód algoritmu, který využívá posun bitů [20]:

```
for(i=0;i<k;i++)
   P[i]=0;
for(i=0;i<k-1;i++){
   for(shift = 0x01; shift!=0; shift<<=1){
      if(b[i] & shift) add(P,a);
      if(P[0] & 0x01) add(P,n);

      for(j=0;j<k-1;j++){ /*Right bit shifting*/
         P[j] >>= 1;
         if(P[j+1] & 0x01) P[j] |= 0x80;
      }
      P[k-1] >>= 1;
   }
}
if(isGreaterEqual(P,n))
  sub(P,n);
```

6.4 Montgomeryho umocňování

Montgomeryho umocňování vychází z Montgomeryho redukce a metody pro umocňování velkých čísel. Platí stejné podmínky jako u Montgomeryho násobení viz výše.

- Vstup: $a, e = (e_t \dots e_0)_2, n = (n_{l-1} \dots n_0)_b$, popř. i r a r^{-1}.
- Výstup: $c = a^e \bmod n$.
- Postup:
  1. $A = a \otimes r^2; C = r$.
  2. For i from t down to 0 do the following:
     (a) $C = C \otimes C$
     (b) If $e_i = 1$ then $C = C \otimes A$.
  3. $c = C \otimes 1$ [21].

Očekávaný počet jednotlivých násobení v tomto algoritmu je dán vzorcem

$$3l(l+1)(t+1),$$

kde l je počet bitů modula n a t je počet bitů exponentu e.

Největší výhoda spočívá v tom, že místo klasického dělení pro získání zbytku po dělení je proveden bitový posun, což je jedna z nejrychlejších operací [22].

7 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 byl realizován prakticky v jazyce C se třemi knihovnami pracujícími s velkými čísly: BigDigits, GMP a OpenSSL. V knihovně BigDigits byla napsána vlastní funkce pro umocnění a modulo. Bylo provedeno srovnání všech tří knihoven z hlediska rychlosti provedení výpočtu, dále bylo změřeno maximální využití paměti a počet vzorků zachycených při běhu programu, to vše pro velikost klíče 1024 a 2048 bitů. Nejrychlejší knihovna se ukázala být GMP, z hlediska počtu odebraných vzorků vyšla nejlépe OpenSSL. To si můžeme vysvětlit tím, že některé instrukce trvají delší dobu a GMP využívá právě tyto jednodušší instrukce oproti OpenSSL. Knihovna GMP se ukázala jako paměťově nejúspornější, oproti OpenSSL potřebovala o více než 4 MB paměti méně. Pro menší paměťovou náročnost i pro rychlost výpočtu klíče je tedy doporučována knihovna GMP. V příloze je stručný návod na instalaci knihoven BigDigits, GMP a OpenSSL ve Visual Studiu 2010.

V poslední části práce byla obecně popsána optimalizace kódu pro nízkoenergetické procesory. Jelikož není možné, kvůli velké paměťové náročnosti, využít tyto knihovny, byla navrhnuta datová struktura, pomocí níž je možno ukládat a operovat s velkými čísly. Dále bylo podrobně popsáno Montgomeryho modulární násobení, jeho varianta s bitovým posunem a Montgomeryho umocňování, což značně zjednodušuje výpočet klíčů pomocí protokolu Diffie-Hellman. Pokud jsou splněny podmínky pro Montgomeryho násobení a jsou předpočítány prvky r a r^{-1}, lze snadno modulárně násobit, protože se ve výpočtu již neobjevuje dělení v operaci $\bmod n$, a tím je algoritmus méně náročný na výpočet. Stává se tak vhodnou optimalizací pro modulární násobení velkých čísel v nízkoenergetických procesorech.

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

[17] Beginners Guide to Performance Profiling. *MSDN Microsoft* [online]. 2013 [cit. 2013-08-13]. Dostupné z: http://msdn.microsoft.com/en-us/library/ms182372.aspx

[18] Význam sloupců udávajících velikost paměti v programu Správce úloh. *Microsoft Windows* [online]. [cit. 2013-08-12]. Dostupné z: http://windows.microsoft.com/cs-CZ/windows7/What-do-the-Task-Manager-memory-columns-mean

[19] MSP430 Ultra-Low-Power Microcontrollers. *Texas Instruments* [online]. 2013 [cit. 2013-08-22]. Dostupné z: http://www.ti.com/lit/sg/slab034w/slab034w.pdf

[20] Optimization of Public Key Cryptography (RSA and ECC) for 16-bits Devices based on 6LoWPAN [online]. Murcia (Španělsko) [cit. 2013-08-18]. Dostupné z: https://www.nics.uma.es/seciot10/files/pdf/ayuso_seciot10_paper.pdf. Computer Science Faculty, University of Murcia

[21] BURDA, Karel. Montgomeryho modulární násobení [Přednáška]. 2010 [cit. 15.8.2013].

[22] Menezes, A. J., van Oorschot, P. C., Vanstone, S. A.: Handbook of Applied Cryptography, CRC Press, 1996 [cit. 20.8.2013].

SEZNAM PŘÍLOH

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 F55BCCC0
A151AF5F 0DC8B4BD 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 1DBF0A01 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 0D2E4310 6D8727DA 37DF9CCE 95B47875 5D06BCEA
8F9D4596 5F75A5F3 D1DF3701 165FC9E5 0C4279CE B07F9895 40AE96D5 D88ED776
Y =
0x717A6CB0 53371FF4 A3B93294 1C1E5663 F861A1D6 AD34AE66 576DFB98 F6C6CBF9
DDD5A56C 7833F6BC FDFF0955 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 5A0CACED 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 mod 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 » 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 rozklikněme Linker, kolonku Input a zde přidáme do Additional Dependencies „gmp.lib“.
- Dále musíme přímo do složky projektu zkopírovat knihovnu gmp.dll a při spuštěni 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], exponentModOuput[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(exponentModOuput, exponentMod, KLIC, two, 1);

    if (mpEqual(exponentModOuput, 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(exponentModOuput, exponentMod, 1, KLIC);
    mpSetEqual(exponentMod, exponentModOuput, 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");
```

```
  printf("   |                                          |\n");
  printf("   |   Michal Jakubicek - Bakalarska prace    |\n");
  printf("   |__________________________________________|\n\n\n");

  DIGIT_T a[KLIC], b[KLIC], p[KLIC], g[KLIC], X[KLIC], Y[KLIC], Ka[KLIC],
  Kb[KLIC];

  mpConvFromHex(a, KLIC, "0xB9A3B3AE8FEFC1A2930496507086F8455D48943E");
  mpPrintHex("A: 0x", a, KLIC, "\n\n");

  mpConvFromHex(b, KLIC, "0x9392C9F9EB6A7A6A9022F7D83E7223C6835BBDDA");
  mpPrintHex("B: 0x", b, KLIC, "\n\n");

  mpConvFromHex(p, KLIC, "0xB10B8F96A080E01DDE92DE5EAE5D54EC52C99FBCFB06A
  3C69A6A9DCA52D23B616073E28675A23D189838EF1E2EE652C013ECB4AEA90611232497
  5C3CD49B83BFACCBDD7D90C4BD7098488E9C219A73724EFFD6FAE5644738FAA31A4FF55
  BCCC0A151AF5F0DC8B4BD45BF37DF365C1A65E68CFDA76D4DA708DF1FB2BC2E4A4371");
  mpPrintHex("P: 0x", p, KLIC, "\n\n");

  mpConvFromHex(g, KLIC, "0xA4D1CBD5C3FD34126765A442EFB99905F8104DD258AC5
  07FD6406CFF14266D31266FEA1E5C41564B777E690F5504F213160217B4B01B886A5E91
  547F9E2749F4D7FBD7D3B9A92EE1909D0D2263F80A76A6A24C087A091F531DBF0A0169B
  6A28AD662A4D18E73AFA32D779D5918D08BC8858F4DCEF97C2A24855E6EEB22B3B2E5");
  mpPrintHex("G: 0x", g, KLIC, "\n\n");

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

  mpPrintHex("X: 0x", X, KLIC, "\n\n");
  mpPrintHex("Y: 0x", Y, KLIC, "\n\n");

  multiplyModulo(Ka, Y, a, p);
  multiplyModulo(Kb, X, b, p);

  mpPrintHex("Ka: 0x", Ka, KLIC, "\n\n");
  mpPrintHex("Kb: 0x", Kb, KLIC, "\n\n");

  getchar();
  return 0;
}
```

B.2 GMP

```
#include "stdafx.h"
#include <stdio.h>
#include "gmp.h"
```

```
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\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
  BFACCBDD7D90C4BD7098488E9C219A73724EFFD6FAE5644738FAA31A4FF55BCCC0A151A
  F5F0DC8B4BD45BF37DF365C1A65E68CFDA76D4DA708DF1FB2BC2E4A4371", 16);
  printf("P = %s\n\n", mpz_get_str(NULL, 16, p));

  mpz_set_str(g, "A4D1CBD5C3FD34126765A442EFB99905F8104DD258AC507FD6406CF
  F14266D31266FEA1E5C41564B777E690F5504F213160217B4B01B886A5E91547F9E2749
  F4D7FBD7D3B9A92EE1909D0D2263F80A76A6A24C087A091F531DBF0A0169B6A28AD662A
  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\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
  FACCBDD7D90C4BD7098488E9C219A73724EFFD6FAE5644738FAA31A4FF55BCCC0A151AF
```

```
    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;
}
```