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

KRYPTOGRAFICKÝ PROTOKOL S VEŘEJNÝM KLÍČEM

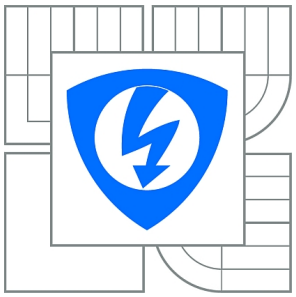
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. RADEK FUJDIAK
AUTHOR

BRNO 2013

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

KRYPTOGRAFICKÝ PROTOKOL S VEŘEJNÝM KLÍČEM

CRYPTOGRAPHY PROTOCOL WITH PUBLIC KEY

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

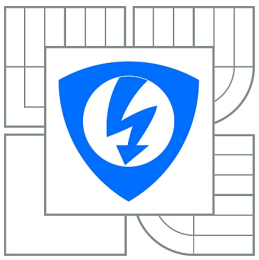
AUTOR PRÁCE — Bc. RADEK FUJDIAK

AUTHOR

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

SUPERVISOR

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student: Bc. Radek Fujdiak ***ID:*** 106440
Ročník: 2 ***Akademický rok:*** 2012/2013

NÁZEV TÉMATU:

Kryptografický protokol s veřejným klíčem

POKYNY PRO VYPRACOVÁNÍ:

Popište algoritmus Diffie-Hellman založený na kryptografii nad eliptickými křivkami (ECDH). Implementujte základní algoritmus Diffie-Hellman do Ultra-Low-Power Microcontrolleru (MSP430). Navrhněte postup implementace algoritmu ECDH do tohoto mikrokontroléru. Na základě toho návrhu implementujte do MSP430 jednotlivé metody nutné pro tento algoritmus (zápis velkých čísel, základní operace s velkými čísly, modulární násobení, generátor náhodných čísel atd.).

DOPORUČENÁ LITERATURA:

[1] Burda.K.: Bezpečnost informačních systémů. Skripta FEKT VUT v Brně, 2005.
[2] Menezes, A. J., van Oorschot, P. C., Vanstone, S. A.: Handbook of Applied Cryptography, CRC Press, 1997, ISBN 0-8493-8523-7.

Termín zadání: 11.2.2013 ***Termín odevzdání:*** 29.5.2013

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

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

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 diplomové 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)

ABSTRAKT

Diplomová práce je úvodem do kryptologie. Práce řeší popis kryptosystémů a dále výběr ideálního kryptosystému pro nízkovýkonový mikrokontroler. Dále je obśahnut návod na instalaci vývojového programu pro Code Composer Studio, několik demonstačních programů, základní ukázka implementace vybraných kryptosystémů s malými čísly a návrh implementace vybraného kryptosystému s velkými čísly.

KLÍČOVÁ SLOVA

Kryptologie, Kryptografie, Symetrický kryptosystém, Advanced Encryption Standard, Asymetrický kryptosystém, Diffie-Hellman, Eliptická křivka, Diffie-Hellman a eliptické křivky, MSP430, náhodný generátor čísel

ABSTRACT

The Master thesis is an introduction to cryptology. The Thesis describe cryptosystems and selects one ideal cypher for low-power microcontroler. In thesis provides manual for instal development program COde Composer Studio, basic implementation of selected cryptosystem with small numbers and suggestion for implementation selected cyptosystem with big numbers.

KEYWORDS

Cryptology, Cryptography, Symmetric cypher, Advanced Encryption Standard, Asymmetric cypher, Diffie-Hellman, Elliptic curve, Elliptic curve Diffie-Hellman, MSP430, random number generator

FUJDIAK, Radek *Kryptografický protokol s veřejným klíčem*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2013. 60 s. Vedoucí práce byl Ing. Petr Mlýnek

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Kryptografický protokol s veřejným klíčem“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

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

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Petru Mlýnkovi, PhD. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci. Dále také Ing. Ondřeji Rašo za pomoc při vypracování práce a také samozřejmě své rodině za podporu během celého studia.

Brno ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

V dnešním světě informací je čím dál více nutné dbát bezpečnosti dat a komunikace, tím se zabýva kryptologie. Nutnost šifrovat je tedy na denním pořádku a setkáváme se s ním v běžném životě (při internetovém bankovnictví, komunikaci na internetu, přenosu dat a jiných běžných lidských činnostech). Pokud by neexistovalo šifrování či metody zabezpečení komunikace a dat obecně, bylo by velice jednoduché zneužívat cizí účty, číst cizí elektronickou poštu či vydávat se na internetu pod cizí identitou. Pro zajištění ochrany nám slouží krpytosystémy.

Advanced Encryption Standard (AES) či Algorithm (AEA) jsou dnes jedny z nejvyužívanějších šifrovacích symetrických blokových kryptosystémů. AES nahradil dnes již zastaralý standard Data Encryption Standard (DES). V roce 2001 Národní institut standardů a technologie (NIST) schválil AES z 15 navrhovaných jako nejvhodnější. O rok později 25. května 2002, začal být tento standard používán i jako federální standard v USA.

Jak již bylo řečeno AES je šifrovací standard. Je zařazen do symetrických blokových kryptosystémů. Symetrický znamená, že využívá pouze jeden klíč k šifrování i dešifrování ($K_E = K_D$). Blokový znamená, že se v tomto kryptosystému pracuje s bloky pevně stanovéné délký (v bitech) narozdíl od proudových kryptosystémů, které využívají kombinaci s pseudonáhodným proudem bitů.

Klíč, který je použit v AES k šifrování a dešifrování je nutné nějakým způsobem vytvořit a následně jej distribuovat a to do jednotlivých zařízeních, které spolu budou spolupracovat. Klíč může být stanoven například náhodným generátorem čísel a následně při výrobě staticky nastaven v zařízeních. Tato varianta je však vzhledem k pozdější nutnosti klíč vyměnit vhodná pouze tam, kde je možné předpokládat velmi malé množství společně komunikujících zařízeních. Důvody pro výměnu klíče jsou různé ať už zastarání klíče (vzhledem k neustálému nárostu výpočetních kapacit je nutné neustále zvyšovat velikost klíče) či vyzrazení klíče. Nicméně při větším počtu účastníků je výměna klíče problematická, proto je u spíše používán ke generování klíče nějaký asymetrický kryptosystém, který následně může zařídit i jeho případnou distribuci po nezabezpečeném kanálu.

Tato práce se zabývá rozborem dnes používaných kryptosystémů, jejich rozdělení a popisem. Dále je vniknuto do problematiky nízkovýkonových mikrokontrolerů, které používají tyto kryptosystémy (převážně tedy kryptosystém AES). Následně se práce zabývá problematikou klíčů a jejich distribuce u kryptosystému AES a nízkovýkonových mikrokontrolerů, tedy návrhem řešení pro generování a distribuce symetrického klíče pro nízkovýkonový mikrokontroler (konkrétně MSP430).

1 KRYPTOLOGIE, KRYPTOGRAFIE A KRYPTOANALÝZA

Kryptografie je věda zabývající se utajováním informací ve zprávách či smyslu zprávy. Utajování probíhá za pomocí matematických (logických) operací změnou zprávy do podoby, která je čitelná pouze se speciální znalostí (například dešifrovací klíč, používaný k deširování zašifrovaných zpráv). Opakem kryptografie je pak kryptoanalýza, tato věda se zabývá problematikou z jiného pohledu. Řeší získávání informací, které jsou za normálních podmínek nepřístupné (informace v zašifrovaném textu apod.). Za pomocí kryptoanalýzy také ověřujeme bezpečnost kryptografických metod. Obecně tedy můžeme říci, že kryptoanalýza se zabývá překonáváním bezpečnosti algoritmů z kryptografie. Obě tyto vědy jsou velice důležité a vzájemně se doplňují. Pokud mluvíme v širším smyslu a chceme zahrnout pojmy kryptoanalýzy i kryptografie, mluvíme již o kryptologii.

Šifra je algoritmus obsahující matematické či logické operace, které dokáží šifrovat prostý text. Jednu z nejznámějších šifer můžeme vidět níže

nezašifrovaný text	S	T	A	S	T	N	E	A	V	E	S	E	L	E
klíč	H	E	S	L	O	H	E	S	L	O	H	E	S	L
zašifrovaný text	A	Y	T	E	I	V	J	T	H	T	A	J	E	Q

Tab. 1.1: Vigenérova šifra s tajným klíčem HESLO

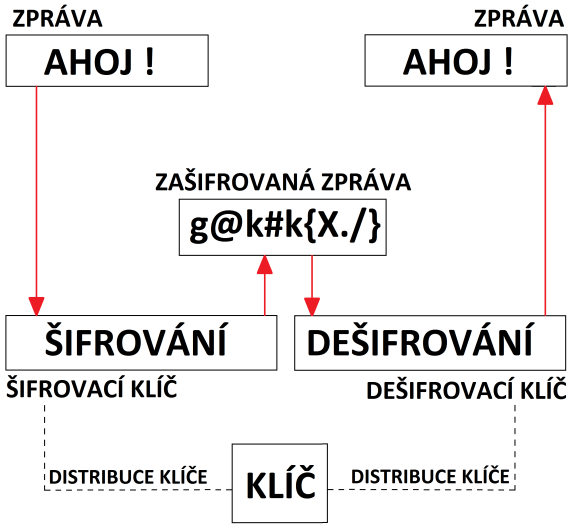
Šifrování je pak proces, při kterém se pomocí dohodnutých algoritmů změní zpráva tak, aby byla nečitelná pro třetí osobu, dešifrování je pak proces opačný, kdy ze zašifrované zprávy pomocí specifických algoritmů (které záleží na použité šifře) vytvoříme opět čitelnou zprávu (Obr. 1.1).

Obr. 1.1: Šifrovací a dešifrovací proces

My se budeme podrobněji zabývat pouze kryptografií, konkrétně dnes používanými algoritmy. Kryptografii dělíme na asymetrické a symetrické kryptosystémy (či někdy můžeme dělení vidět přímo jako symetrickou a asymetrickou kryptografii). Kryptosystém je obecnější soubor matematických nebo logických algoritmů, které již nemají za úkol pouze šifrovat a dešifrovat, ale například i vytvářet samotné klíče pro šifrování či dešifrování, podepisování aj. [1]

1.1 Symetrická kryptografie

Symetrická kryptografie (Obr. 1.2) využívá pro šifrování i dešifrování stejného klíče ($K_E = K_D$) a je tedy nutné zajistit bezpečnou distribuci klíčů, aby nedošlo k narušení bezpečnosti třetí osobou.



Obr. 1.2: Šifrování a dešifrování v symetrických kryptosystémech

K zajištění bezpečené distrubuce klíčů je potřeba předávat šifrovací klíč bezpečným kanálem a to před zahájením komunikace. Z pravidla to bývá velký problém. Další nevýhodou je nutnost zajistit velký počet klíčů při komunikaci ve velkých firmách, kdy je třeba zajistit komunikaci mezi všemi zaměstnanci takovým způsobem, aby mohl každý s každým komunikovat bezpečně, tedy tak, aby komunikace nebyla narušena třetí osobou. To například při společnosti o 10000 zaměstnancích znamená zajistit 5 miliónů klíčů, z čehož jasně plyne nevyužitelnost těchto metod například v internetu, kde počet uživatelů sahá k miliardám.

Velkou výhodou symetrických kryptosystému je jejich rychlost a malá náročnost na velikost klíče, tedy i malá náročnost na výkon. Pro 128 bitový symetrický kryptosystém AES-128 je pro udržení stejné bezpečnostní úrovně potřeba minimální

velikosti 3072 bitů velkého veřejného klíče pro asymetrický kryptosystém RSA [2]. Využití symetrických kryptosystémů se nalézá například při šifrování dat, kdy není třeba jakéhokoliv přenosu či komunikace (šifrování dat v počítači).

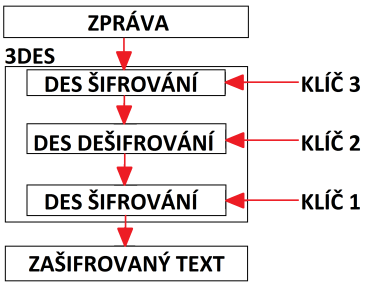
Symetrická kryptografie se dělí na proudové kryptosystémy a blokové kryptosystémy. V tomto textu se budeme zabývat pouze základními blokovými kryptosystémy DES, 3DES a AES.

1.1.1 Data Encryption Standard

Data Encryption Standard (DES) je jeden ze základních symetrických kryptosystémů. Byl prolomen a je dnes považován za nebezpečný. V upravené formě Triple Data Encryption se však stále používá.

Triple Data Encryption Standard

Triple Data Encryption Standard (3DES) je upravenou variantou kryptosystému DES. Algoritmus 3DES a jeho vztah k DES můžeme vidět na obrázku (Obr. 1.3)



Obr. 1.3: Šifrování a dešifrování v symetrických kryptosystémech

Jak můžeme z obrázku a názvu vyvodit jedná se o ztrojení kryptosystému DES. A můžeme jej popsat vztahem 1.1

$$\begin{aligned} C &= E(K_3, D(K_2, E(K_1, Z))), \\ Z &= D(K_1, E(K_2, D(K_3, C))), \end{aligned} \tag{1.1}$$

kde Z je nešifrovaná zpráva, C šifrovaná zpráva, K_1 až K_3 jsou klíče 3DES, E() je funkce pro šifrování a D() je funkce pro dešifrování.

Tento způsob má tři varianty, které jsou závislé na použitých klíčích.

- Plný 3DES, všechny klíče jsou nezávislé ($K_1 \neq K_2 \neq K_3$)
- Tzv. 2DES, první a druhý klíč jsou na sobě nezávislý, ale první a třetí jsou stejné ($K_1 = K_3 \neq K_2$)
- Jednoduchý DES všechny klíče jsou na sobě závislé ($K_1 = K_2 = K_3$)

První varianta je nejbezpečnější (je také nejčastěji používána), klíč má efektivní délku 168b (3x 56b). Druhá varianta má efektivní délku klíče již jen 112b (2x 56b). Třetí varianta je jen ekvivalent klasického DES použitého třikrát za sebou, dvě předchozí operace se vzájemně ruší a tak se jedná tedy o čistý klasický DES s efektivní délkou klíče 56b (varianta je tedy samozřejmě považována za nebezpečnou).

Právě 3DES je dnes používána pro elektronický platební průmysl [9] nebo data, která jsou chráněna heslem [10] [11] [12]. 3DES byl zaveden z důvodu, aby nebylo nutné přecházet na zcela nový algoritmus. Nicméně 3DES kryptosystém je oproti nově navrženým algoritmům (např. AES) mnohem pomalejší, a proto se od jeho používání pomalu ustupuje.

1.1.2 Advanced Encryption Standard

Advenced Encryption Standard (AES) je kryptosystémem nahrazující dříve užívaný kryptosystém DES [13]. AES využívá pro šifrování a dešifrování stejný klíč pro data s pevně stanovenou délkou bloku (je tedy symetrickým blokovým kryptosystémem). Základní princip algoritmu AES můžeme shrnout do těchto bodů

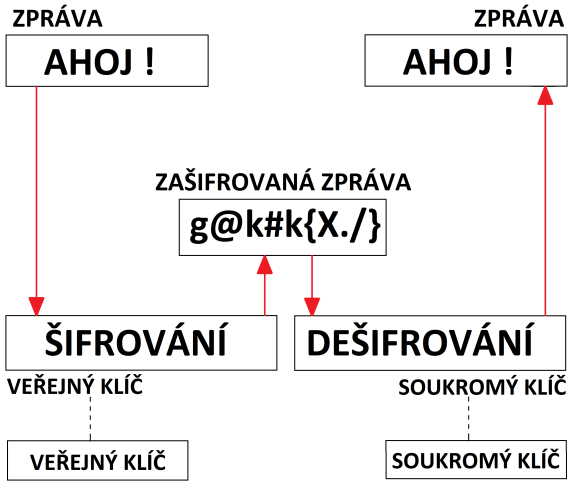
- Expanze klíče (Rijndaelův seznam klíčů)
- Inicializační část (přidání podklíče, XOR nad všemi bity)
- Iterace (záměna bytů, prohození řádků, kombinování sloupců a přidání podklíče)
- Závěrečná část (záměna bytů, prohození řádků, přidání podklíče)

Jak můžeme vidět jednotlivé body jsou složeny z několika částí/operací. Přesný matematický popisu modelu AES není však předmětem této práce a nebudeme se jím do podrobností zabývat.

AES se dnes používá například pro bezdrátové Wi-Fi sítě (WPA2), avšak vysoká rychlost AES a nízke nároky na RAM paměť určují obecně AES pro širokou škálu hardwaru, například 8-bitové čipové karty, nízkovýkonové mikrokontrolery a spousty dalších odvětví, kde se pracuje s nízkým výkonem.

1.2 Asymetrická kryptografie

Asymetrická kryptografie (Obr. 1.4) využívá pro šifrování i dešifrování různé klíče ($K_E \neq K_D$). Tyto kryptosystémy nejčastěji využívají dva klíče, soukromý klíč a veřejný klíč.



Obr. 1.4: Asymetrická šifra

Veřejný klíč je veřejně dostupný, nebývá utajován a naopak je zveřejňován například na internetu. Slouží například při ověřování podpisu či dešifrování zpráv. Soukromý klíč je naopak utajován (tvůrcem kryptosystému) a jeho vyzrazení znamená ohrožení bezpečnosti celého kryptosystému. V Případě vyzrazení soukromého klíče musí dojít k jeho výměně (nové vygenerování klíče). Soukromý klíč se využivá například pro podepisování či šifrování zprav.

Asymetrické kryptosystémy jsou založeny na nemožnosti narušit bezpečnost kryptosystému v reálném čase a přečíst tak zašifrovanou zprávu bez znalosti soukromého klíče. Pro tuto skutečnost jsou využívány složité matematické problémy (faktorizace, problém diskrétního logaritmu aj.).

Díky existenci dvou druhů klíčů (soukromý a veřejný klíč) odpadá u těchto metod nutnost distribuce klíče, jako tomu bylo u symetrických kryptosystémů. Na klíče však vzniká jiný nárok. Je zjevné, že klíče mají vnitřní matematickou souvislost. Je nutné, aby nebylo možné ze znalosti veřejného klíče odvodit klíč soukromý (ani za pomoci jiných prvků jako např. zašifrované zprávy). Dá se tedy říci, že asymetrická kryptografie je založena na jednocestných funkcích. Funkce u kterých je velice jednoduché vytvořit výstup, avšak z výstupu pak je velice obtížné (až nemožné) určit vstup (bez znalosti utajovaných parametrů). Příkladem může být jednoduché násobení, kdy jsme schopni vynásobit libovolná velká čísla, avšak následná faktorizace je

velice obtížná (tohoto jevu využívá jedna z metod asymetrické kryptografie - RSA).

Hlavní výhodou asymetrických kryptosystémů je tedy to (hlavně oproti symetrickým kryptosystémům), že odpadá nutnost přenosu klíče a tak se i zvyšuje bezpečnost kryptosystémů, díky tomuto odpadá i problém s velkým množstvím klíčů. U asymetrických kryptosystémů je potřeba mnohem menší množství klíčů. Nevýhodou pak, oproti symetrickým šifrám, jsou mnohonásobně pomalejší algoritmy a mnohonásobně větší velikost klíče (viz. kapitola o symetrických kryptosystémech). Využití asymetrických kryptosystémů je například při podepisování, šifrování dat, komunikace apod.

Vzhledem k výhodám a nevýhodám jednotlivých kryptosystémů se obvykle asymetrické a symetrické kryptosystémy kombinují. Tím se využívá výhod obou systémů. Například veřejný klíč (asymetrického kryptosystému) může být použit pro zašifrování tajného klíče (symetrického kryptosystému) a pak již není nutné zajišťovat pro přenos klíče speciální bezpečný kanál, ale je možné jej poslat i kanálem ovládaným útočníkem.

Kombinací symetrických a asymetrických kryptosystémů vzniká tzv. hybridní kryptosystém. Dnes používán například při certifikaci. Hybridními kryptosystémy se budeme zabývat v kapitole o eliptických křivkách, kde se v upravené formě používají. Nicméně mezi asymetrickými a hybridními kryptosystémy je v praxi zanedbatelný rozdíl, a proto hybridní kryptosystémy zařazujeme do asymetrických kryptosystémů (nestavíme je tedy mimo, jako samostatnou kategorii).

1.2.1 Merkle-Hellman Knapsack

Merkle-Hellman Knapsack (MHK) je jeden z prvních asymetrických kryptosystémů. MHK je jednosměrný (například oproti RSA), nedá se proto použít pro autentizaci. Veřejný klíč je tedy pouze pro šifrování a soukromý klíč pouze pro dešifrování. Pro generování klíčů pro n-bitovou zprávu použijeme tzv. superincreasing, tedy metodu, kdy vytvoříme množinu přirozených čísel, kdy každé následující je větší než součet všech předchozích

$$w = (w_1, w_2, ..., w_n), \tag{1.2}$$

z této množiny pak vybereme náhodné celé číslo q dle

$$q > \sum_{i=1} n w_i, \tag{1.3}$$

a dále náhodné celé číslo r dle

$$gcd(r, q) = 1, \tag{1.4}$$

kde gcd() je funkce určující největší společný dělitel dvou čísel (v našem případě r a q), dále ještě spočteme

$$\begin{aligned} \beta &= (\beta_1, \beta_2, ..., \beta_3), \\ \beta_i &= rw_i \bmod q. \end{aligned} \tag{1.5}$$

Z toho veřejný klíč je β a soukromý klíč je (w, q, r). Tyto klíče jsou pak použity pro šifrovací a dešifrovací funkce. MHK používá pro šifrování či dešifrování tzv. knapsack problem neboli problém batohu. Tento problém je popsán tak, že máme sadu čísel A a jedno číslo b, hledáme pak takovou podmnožinu A, která je sumarizací b.

Tento algoritmus byl prolomen a není dnes již považován za bezpečný. [14]

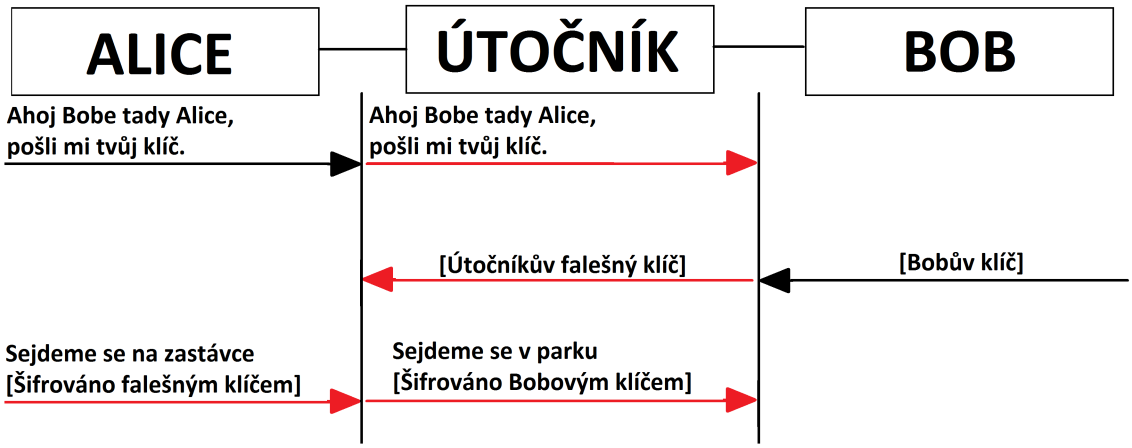
1.2.2 Diffie-Hellman

Diffie-Hellman (D-H) využívá problému diskrétního logaritmu, tento problém je využíván i dalšími kryptosystémy jako je například DSA či ElGamal, kterými se budeme zabývat níže. Tento kryptosystém umožňuje komunikovat přes nezabezpečený kanál, který může ovládat potencionální útočník a to aniž by se komunikující strany přede znaly (vytvoří se symetrický klíč, který se pak dále používá pro šifrování zbylé komunikace). Klíč je zkonstruován oběmi účastníky komunikace současně, tudíž není nutné klíč posílat, díky tomu potenciální útočník nezachytí při komunikaci šifrovací klíč, protože se nepřenáší, vzniká zde však jiný problém a to Man in Middle, protože D-H neumožňuje autentizaci účastníků (ověření identity účastníků komunikace).

Obr. 1.5: Man in Middle

Man in Middle (Obr. 1.5) neboli muž uprostřed, je jeden z nejznámějších problémů současné kryptografie. Je to snaha útočníka odposlouchávat, přesměrovávat, zablokovat či jinak poškodit nebo ovlivnit komunikaci mezi dvěma účastníky tím, že se stane aktivním prostředníkem komunikace.

Budeme se řídit obrázkem 1.6 a ukážeme si na něm příklad komunikace s úspěšným útokem.



Obr. 1.6: Man in Middle

1. Alice posílá zprávu útočníkovi s žádostí o klíč.
2. Útočník přijme a přepošle žádost Bobovi.
3. Bob posílá svůj klíč K_{Bob}, ten je ale přijat Útočníkem.
4. Útočník po přijetí klíče vysílá Alici svůj klíč $K_{Útočník}$.
5. Alice přijme falešný klíč a zašifruje svou tajnou zprávu za pomocí něj a tuto zprávu odešle.
6. Útočník zprávu přijme, rozšifruje svým klíčem $K_{Útočník}$ a pošle svou falešnou zprávu zašifrovanou klíčem K_{Bob}.

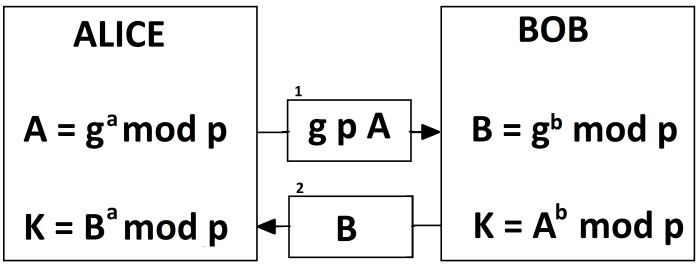
Nikdo během komunikace nepozná, že útočník nahradil zprávu, přijmul tajný klíč a přečetl tajnou zprávu. Systém (když nebereme v potaz kombinaci s jinými systémy) se používá pouze tam, kde útočník nemůže aktivně zasáhnout do komunikace, právě z důvodu, že není umožněna autentizace.

D-H, jak bylo řečeno, využívá problému diskrétního logaritmu. Vysvětleme si tedy nejprve diskrétní logaritmus. Mějme přirozená čísla (m, q, k, Y) pro které platí

$$Y \equiv q^k \bmod m. \tag{1.6}$$

Potom každé číslo k odpovídající uvedené rovnici nazveme diskrétním logaritmem o základu q z Y vzhledem k modulu m [8]. Pro asymetrickou kryptografii je hlavní vlastnostní pro diskrétní logaritmus, že zatímco Y můžeme velice snadno spočíst ze znalostí (k, m, q), tak spočíst diskrétní logaritmus k je velice obtížné.

Na této skutečnosti se pak zakládá algoritmus D-H, který můžeme vidět na obrázku (Obr. 1.7)



Obr. 1.7: Ukázka šifry Diffie-Hellman

Z Obrázku (Obr. 1.7) je patrný průběh komunikace mezi dvěmi stanicemi (Alice a Bob). Čísla (g, p) jsou veřejná. Jako první je na stanici Alice zvoleno náhodně tajné číslo a a vypočítáno A, které následně obdrží druhá stanice (Bob), která již vypočítala své B stejným způsobem. Nyní v jednom kroku proběhne zaslání B na stanici Alice. Jako poslední se pak vypočte tajný klíč K na obou stanicích.

Nejčastěji se Diffie-Hellman samostatně používá pro klíčovou výměnu, dále pak v technologii SSH2 (OpenSSH) se využívá algoritmus Diffieho-Hellmana pro výpočet společných klíčů pro libovolnou množinu uzlů a to právě z důvodu, že mezi těmito uzly nemusí (díky vlastnostem algoritmu Diffie-Hellman) existovat předem vytvořený bezpečný kanál, takto získané klíče se používají pro symetrické šifry (například AES,3DES), pomocí kterých je pak šifrována další komunikace.

1.2.3 Rivest,Shamir a Adleman kryptosystém

Rivest,Shamir a Adleman kryptosystém neboli RSA je první asymetrická šifra, kterou lze použít jak pro elektronický podpis, certifikaci, tak i pro šifrování samotné. U elektronického podpisu nejsou zpracovávána data jako taková, ale používá se jen pro jejich hašovací hodnotu (hašování je proces, kde ze vstupních dat vytvoříme pomocí matematického algoritmu velmi malý reprezentant hašovaných daných dat). Samotné šifrování se pak využívá při šifrování symetrických klíčů. Jak můžeme vidět RSA se využívá pouze pro data malých objemů, je to z toho důvodu, že RSA je velice pomalé (má velmi nízkou rychlost šifrování u velkých objemů dat)[3].

Oproti D-H využívá RSA problematiku faktorizace čísel používaných v rámci algoritmu. Tvorba klíčového páru pak probíhá následovně, jako první zvolíme dvě velká prvočísla

$$p, q > 10^{115}, \tag{1.7}$$

následně vypočteme modula n a r

$$\begin{aligned} n &= pq, \\ r &= (p-1)(q-1), \end{aligned} \tag{1.8}$$

poté následuje volba veřejného klíče VK, které je nesoudělné s r a výpočet soukromého klíče SK

$$(VK \bullet SK) \bmod r = 1. \tag{1.9}$$

Následně zveřejním VK a n, ostatní parametry zůstávají tajné. V tomto případě kryptosystém RSA stojí na nemožnosti faktorizovat n. Šifrování a dešifrování pak můžeme zjednodušeně popsat jako C a Z, kde C je šifrovaný text a Z (pro platí, že Z < n) jsou šifrovaná data či zpráva.

$$\begin{aligned} C &= Z^{VK} \bmod n, \\ Z &= C^{SK} \bmod n. \end{aligned} \tag{1.10}$$

Dnes je RSA používáno, avšak je nutné mít dostatečně velký klíč (za bezpečné je dnes považován klíč o velikost 2048 bitů), aby nedošlo k narušení bezpečnosti. Společnosti, které RSA používají, jsou například IBM, Novell a Apple, tyto společnosti jej využívají přímo ve svých produktech (IBM Remote Supervisor Adapter II). Následně firmy jako Citcorp, Raytheon nebo třeba Boeing jej používají na internetu (například pro certifikaci). Pak firmy jako například Xanadu využívají RSA pro zabezpečení komunikace v počítačových sítích.

1.2.4 Digital Signature Algorithm

Volně přeloženo znamená Digital Signature Algorithm (DSA) algoritmus digitálního podpisu, jak již z názvu vyplívá algoritmus DSA byl vyvinut hlavně pro digitální podpisy (tedy například podepisování na internetu), je založen jak již bylo řečeno na problému výpočtu diskrétního algoritmu (a je velice podobný algoritmu ElGamal). Standartizován je pak ve standardu Digital Signature Standard (DSS).

Algoritmus vytváření klíču má dvě fáze. V první se vyberou sdílené parametry.

1. Provede se výběr kryptografické hašovací funkce, ta zajišťuje proces hašování. V Původním DSA to byla striktně SHA-1, dnes je již povolena i její nástupce SHA-2 (popis funkcí SHA není předmětem této práce, proto se jím zabývat blíže nebudem).
2. Dále se rozhodne o parametrech (L, N), které určují délku klíče. (Původně L omezena násobky 64 v rozsahu 512 až 1024 včetně), dnes je doporučováno 3072 bitů pro klíče, které se budou využívat do roku 2030 a to při užití adekvátního N. [15] N je pak vybíráno dle norem a to L = 3070 a N = 256. [16] Obecné délka N musí být alespoň taková, jako délka výstupu použité hašovací funkce.
3. Dále se vybírá N-bitové prvočíslo q.
4. Výběr L-bitového prvočísla p, tak, že (p-1) je násobek q
5. Jako poslední vybereme g jehož multiplikativní řád modulo p je práve q.

Všechny výše zmíněné prvky jsou veřejné a následně se vytvoří klíč

1. Nejprve náhodně vybereme x v rozsahu $0 < x < q$
2. Pak se spočítá $y = g^x \bmod p$

Veřejný klíč je pak dán čtveřicí (p, q, g, y) a soukromý klíč je x. Následně tyto klíče můžeme využít pro podepisování a ověřování podpisu. DSA se tedy používá při digitálních podpisech a podobných činnostech, mimo jiné jej využivá OpenSSL, OpenSSH či GnuPG (dnes velice rozšířené a velice využívané open-source technologie pro šifrování).

1.2.5 ElGamal

ElGamal je jeden z dalších asymetrických kryptosystémů. Dělit jej můžeme na ElGamal Encryption System (EGES) a ElGamal signature scheme (EGSS). EGES se používá pro šifrování (a je velice podobná D-H algoritmu, využívá také diskrétní logaritmus) a EGSS je varianta pro digitální podpisy (z EGSS pak vychází DSA algoritmus, který je velice podobný). Jeho největší nevýhodou je však to, že velikost šifrovaných dat je dvakrát větší než data nešifrovaná a to bývá jeden z důvodů, proč není tak masově využívána jako například RSA.

Generování klíčů probíhá tak, že z cyklické grupy G o řádu q s generatorem g volíme náhodně $x \in \{1, ..., q-1\}$, následně se pak vypočítává $h = g^x$. (G, q, g, h) se vydává jako veřejný klíč a x jako soukromý klíč, který je držen v tajnosti. Pomocí těchto klíčů následně probíhá šifrování i dešifrování.

ElGamal šifra má využití v digitálních podpisech a i při šifrování, ikdyž není tak masově využívána jako ostatní algoritmy. ElGamal šifru využívá například GNU Project ve svém GNU Privacy Guard, což je volně šiřitelná alternativa k PGP Suite,

která se používá pro šifrování a dešifrování dat v počítači a pro autentizaci při přenosu dat.

1.2.6 Kryptografie s Lucasovými funkcemi

Je to jeden z nejnovějších algoritmů (není to samostatný kryptosystém), který se pokouší o nahrazení algoritmů uvnitř nynějších kryptosystémů jako RSA, Diffie-Hellman, snaží se využít všech výhod těchto kryptosystémů a přidat do nich vlastní způsob, kterým by se měla zvyšovat bezpečnost. Jak bylo řečeno, algoritmus Lucasových Funkcí (LUC) je asymetrický algoritmus, založený na poznatcích z RSA, ElGamal a Diffie-Hellman kryptosystémů, avšak oproti nim se u tohoto algoritmu používají Lucasovi funkce (místo umocňování se zde používá iterace). Lucasovi funkce se snaží o totéž jako eliptické křivky, avšak zde je místo eliptických křivek použita LUC. Některé studie ukazují, že Lucasovi funkce nemají tak velké výhody v bezpečnosti, jak tvrdí vývojáři [4]. Z tohoto důvodu se jimi nebudeme dále a blíže zabývat.

1.2.7 Kryptografie eliptických křivek

Eliptické křivky, tedy stejně jako LUC, samy o sobě nejsou kryptosystémem, pouze nahrazují či doplňují vnitřní algoritmy již hotových kryptosystémů a to tak, že jsou u nich využity právě eliptické křivky, dochází tak k navýšení bezpečnosti, rychlosti či jiných potřebných parametrů, kvůli kterým eliptické křivky zavádíme. Jejich využití se nachází u asymetrických kryptosystémů, jejich hlavní zástupce si nyní ve stručnosti přiblížíme.

Diffie-Hellman s využitím eliptických křivek

Diffie-Hellman a eliptické křivky se označuje jako ECDH, tento kryptosystém je založen na principu Diffie-Hellman a přizpůsoben eliptickým křivkám. Jedná se o kryptosystém, který umožňuje přenos tajných dat po nešifrovaném kanále pro dvě strany, které o sobě nemají žádné informace a dříve spolu ještě nikdy nekomunikovaly. Této metody se využívá například pro přenos klíče či jeho pro vytvoření. Tento klíč pak bývá použit pro zašifrování následující komunikace mezi dvěma stranami, již za použití symetrické šifry.

Tato metoda byla shledána bezpečnou, neboť vyřešení problému diskrétních logaritmů u eliptických křivek je složitý problém, který není možné vyřešit v reálném čase. Dále také se při komunikaci nepřenáší nic jiného než veřejné klíče a žádná ze stran nedokáže z této informace či z tohoto klíče zjistit klíč soukromý, pokud by právě nevyřešila problém diskrétních logaritmů nad eliptickými křivkami. Veřejné

klíče bývají neměnné (schvalované pomocí certifikátů například) či dočasné, kdy jejich platnost je omezena na nějaké určité období, kdy poté je nutné vygenerovat nový veřejný klíč.

Digital signature algorithm s využitím eliptických křivek

Digital signature algorithm s eliptickými křivkami (ECDSA) je varianta DSA, kde jak název napovídá, je využito eliptických křivek. Algoritmus se opět používá stejně jako DSA pro podepisování (signaturu). Opět je zde zmenšení bitové velikosti klíčů, ECDSA považuje za bezpečnou velikost 160 bitů, zatímco pro stejnou úroveň bezpečí je nutné u DSA použít 1024-bitový klíč. Ač ECDSA využívá všech výhod eliptických křivek a je variantou DSA, tak 29. března 2011 byl proveden úspěšný útok, chyba byla nalezena v OpenSSL v1.0.0e (který používá ECDSA), kde úspěšný útok znamenal odcizení privátního klíče ze serveru. Byla provedena revize a problém byl vyřešen, takže šifra ECDSA nebyla shledána jako nevyhovující [5].

Qu-Vanstone s využitím eliptických křivek

Elliptic Curve Qu-Vanstone šifra (ECQV) je speciální šifra, která je určena pro implicitní certifikaci či autentizaci. Používá se zde certifikační autority CA, která dohodne mezi dvěma stranami veřejné klíče, zajistí kopie pro obě strany, ověří identity. Implicitní certifikace se od digitálních liší tím, že certifikáty jsou menší (448-bitů) a díky implicitnímu ověřování je ušetřeno výpočtů, čili méně náročnější. Tato šifra je poměrně nová, navržena na certifikaci, zatím však téměř nevyužívána, i v novém Suite A/B zatím nebyla zavedena.

Menezes-Qu-Vanstone s využitím eliptických křivek

Menezes-Qu-Vanstone s použitím eliptických křivek (ECMQV) používá opět například oproti DH eliptické křivky. ECMQV šifry však má bezpečnostní problémy a neměly by být používány, vhodnější je použít práve DH s eliptickými křivkami neboli ECDH.

Integrační šifra s využitím eliptických křivek

Integrační šifra (IES) je hybridní šifra, která má dvě varianty a to DLIES, kde se používá diskrétního logaritmu a ECIES, kde se používá eliptických křivek, my se budeme zabývat druhou variantou a to variantou ECIES. Tato varianta byla vytvořena Victorem Shoupem v roce 2001. ECIES je také hybridní šifrou, která vychází z šifry Diffie-Hellman a je jednou z nejrozsáhlejších šifer v rámci eliptických křivek. Velká výhoda eliptických křivek je možnost mít mnohem menší klíč při stejné

bezpečnosti, u ECIES je možnost mít 160-bitový klíč stejně bezpečný jako například 1024-bitový RSA klíč., jak je vidět je to veliká úspora, máme tak menší omezení paměti a také větší šifrovací rychlost šifrovacího algoritmu.

1.3 Sada Suite A a Suite B

Suite A a Suite B jsou sady kryptografických algoritmů, takže to nejsou přímo kryptografické šifry, avšak obsahují náhrady za již dosluhující algoritmy. V Roce 2005 byly tyto sady vydány Americkou Národní bezpečnostní agenturou (NSA).

1.3.1 Suite A

Suite A je sada kryptografických algoritmů Accordion, Baton, Medley, Shillelagh a Walburn. Tyto algoritmy jsou přísně utajované a slouží pro šifrování či k použití tam, kde jsou ty nejcitlivější informace (armáda, vláda aj.). Tento balíček, konkrétně tedy samotné šifry, jsou tedy přísně utajovány.

1.3.2 Suite B

Suite B pak je balíček nahrazující algoritmy jako RSA, TrippleDES, MD5, SHA. Využívá se u neutajovaných informacích, důvěrných informací v rámci například státní zprávy a pro komerční účely.

Pro šifrování používá algoritmus Advanced Encryption Standart (AES) s délkou klíče 128/256 bitů. Pro přenos se využívá varianty AES Galois Counter Mode (AES GCM). Pro digitální podpisy se využívá pokročilého ECDSA, tedy algoritmu DSA v kterém jsou implementovány eliptické křivky, zde je použito 256/384 bitů. Pro změnu klíčů se pak používá opět eliptických křivek a to v rámci algoritmu Diffie-Hellman čili algoritmu ECDH, zde se používá také 256/384 bitů. Dále pak Suite B obsahuje například ještě standardy pro hašování [6].

2 PRAKTICKÁ ČÁST DIPLOMOVÉ PRÁCE

V teoretické části této práce jsme si popsali většinu známých a dnes používaných kryptosystémů z asymetrické i symetrické kryptografie. Pozastavíme se u kryptosystému AES, který se využívá právě například i v nízkovýkonových mikrokontrolerech [17] [18] [19] [20]. Jak bylo řečeno již v teoretickém úvodu, AES je symetrický kryptosystém a využívá tedy stejné klíče pro procesy šifrování i dešifrování a je nutné u něj zajistit distribuci klíčů do zařízeních, které spolu budou v budoucnu komunikovat. V nízkovýkonových mikrokontrolerech je používán systém staticky nastavených klíčů a to z důvodu, že je pro ně využití klasických asymetrických metod výkonově nemožné, jednoduše řečeno asymetrické kryptosystémy jsou příliš náročné pro omezený výkon těchto mikrokontrolerů. U této vyrianty vznikájí problémy pokud je nutné klíče vyměnit a tyto problémy se exponenciálně zvětšují s počtem účastníků při komunikaci. V praktické části se tímto problémem budeme zabývat.

Jako první je popsán vybraný nízkovýkonový mikrokontroler, dále je popsáno vývojové prostředí k němu a stručně naznačeno ovladání tohoto mikrokontroleru. Poté je vybrán asymetrický kryptosystém, který je blíže popsán a je řešena problematika jeho implementace do našeho mikrokontroleru.

2.1 Popis mikrokontroleru MSP430

MSP430 jsou nízkovýkonové mikrokontrolery, které jsou vyvíjeny firmou TexasInstrument. Pro naše účely byl poskytnut mikrokontroler řady MSP430x5xxx (konkrétně MSP430F5438A).

Tato řada mikrokontrolerů je schopna pracovat až do 25 MHz, mají 256 kB flash paměti a až 18kB RAM. Mají nízkou spotřebu a obsahují inovativní správu napájení modulu pro optimální spotřebu energie. Další výhodou je již integrované USB. Základní specifika jsou

Energetické parametry:

- 0,1 μA pro RAM
- 2,5 μA pro režim hodiny (reálný čas)
- 165 μA pro aktivní MIPS

Ostatní parametry mikrokontroleru:

- Obsahuje funkci rychlého probuzení (z pohotovostního režimu za méně než 5 μs.
- Flash až 256 kB
- RAM až 18 kB
- ADC 10/12-bit SAR
- GPIO 29,31,47,48,63,67,74 nebo 87 pinů
  a spousty dalších parametrů (časovače, hodiny aj.), které jsou určeny přímým výběrem mikrokontroleru, nastavením vstupních parametrů apod.

Obr. 2.1: MSP-FET430UIF USB

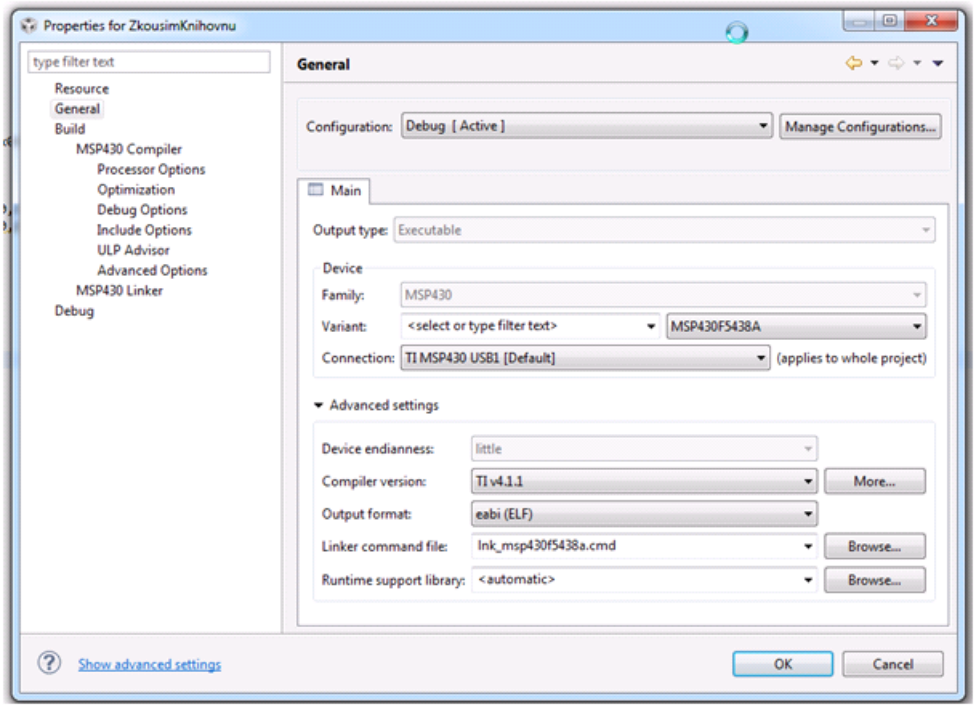
K popisovanému mikrokontroleru patří ještě MSP-FET430UIF (Obr. 2.1), jedná se o nízko výkonové zařízení, nepotřebuje žádné další externí napájení. Do počítače je toto zařízení připojeno pomocí USB. Jedná se o zařízení, díky kterému jsme schopni velice jednoduše a rychle ovládat mikrokontroler (například nahrávání softwaru, vymazání paměti).

2.2 Code Composer Studio

Code Composer Studio (CCS) je vývojový software pro práci s produkty od firmy Texas Instruments, tedy i pro náš přípravek MSP430. Instalace CCS není složitá, ale je třeba postupovat netradičně. Přiložený disk s knihovnami a ovladači neobsahuje vše, co CCS potřebuje, byli jsme tedy nuceni přistoupit na online instalaci. Po stáhnutí potřebného instalačního balíčku (CCS verze 5.2.1.00018), je třeba zvolit stažení veškerých knihoven a ovladačů, poté už jen počkat na instalaci a přejít k nastavení.

Nastaveni CCS není nikterak těžké, ale je třeba postupovat přesně dle postupu. Klikneme na záložku Project → Properties, dále pak nastavíme vše dle obrázku (Obr. 2.2).

Po nastavení CSS nebrání již nic ve vytvoření nového projektu a k případné kompilaci programu do mikrokontroleru.



Obr. 2.2: Nastaveni Code Composer Studia

2.3 Ovládání MSP430

Na praktických ukázkách je vysvětleno, jakým způsobem manipulovat s prvky vývojového kitu MSP430. Ke spuštění programu je třeba mít nainstalovanou správnou verzi programu CSS a nastavit jej dle návodu. Dále je třeba mít připojený mikrokontroler MSP430 k počítači pomocí MSP-FET430UIF.

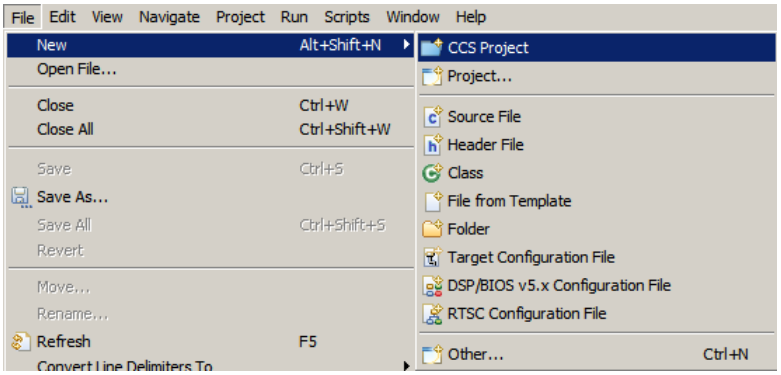
Vytvoření projektu

Zde si ukážeme, jak správně vytvořit projekt a základní main.c soubor. Nejprve vybereme File → New CCS Project (Obr. 2.3).

Následně se nám otevře okno (viz. příloha A). V Project Name vybereme jméno céleho projektu. Output nám určuje co budeme zakládat a my vybereme Executable (další možnosti jsou například library pro knihovnu ap.).

Následuje výběr zařízení pro který kód bude vytvářen. Family určuje tzv. rodinu, Variant určuje typ (první záložka je obecné rozdělení např. MSPx5xxx a druhá již

konkrétní typ např. MSPf5438A), connection nám poté určuje do jakého vstupu máme zařízení zapojené (je dobré zapojovat zařízení vždy do stejného vstupu).



Obr. 2.3: Vytvoření CCS Projektu

V posledních částech Advanced settings nastavujeme např. typ kódování a jiná pokročilá nastavení a v Project templates and examples nastavujeme zda chceme prázdný projekt (empty project), základní main (empty project with main.c) popř. jiné typy (SYS/BIOS apod.).

Po vyplnění a nastavení všech parametrů projektu můžeme stisknout tlačítko finish a projekt se nám vytvoří, dále pak už můžeme programovat pomocí jazyka C. V Následujícím textu si již ukážeme samotné programování pro MSP430f5438A.

Program pro ovládání diody

V této části si popíšeme program uvedený v (Příloze B). Jedná se o jednoduchý program, který řídí diodu D1 na poskytnutém přípravku. Nyní přejdeme k popisu kódu.

```
1 #include <msp430f5438.h>
```

Ukázka kódu 2.1: Knihovna

Jedná se o inicializaci knihovny pro mikrokontroler MSP430, konkrétně typu F5438 (popř. F5438A). Knihovna obsahuje inicializaci pinů, paměti pro daný mikrokontroler.

```
4 WDTCTL = WDTPW + WDTHOLD;
```

Ukázka kódu 2.2: WatchDog Timer

Tato část slouží pro vypnutí/zastavení tzv. WatchDog Timer, je to hlídač, který resetuje paměť/program. Slouží jako ochrana mikrokontroleru, pro nás však značně nevhodná [7].

```
5 P1DIR |= 0x41;
6 P1OUT = 1;
```

Ukázka kódu 2.3: Piny

Zde probíhá nastavení pinu diody.

```
8 while(1)
```

Ukázka kódu 2.4: Nekonečná smyčka

Nekonečné zacyklení programu, které je třeba pro neustálý běh programu.

```
11 P1OUT ^= 0x41;
```

Ukázka kódu 2.5: Piny

Zde probíhá binární negace hodnoty pinu diody.

```
13 i = 5000;
14 do (i--);
15 while (i != 0);
```

Ukázka kódu 2.6: Nastaveni

Nastavení zpoždění, tedy prodlevy mezi negací pinu diody.

2.4 Výběr kryptosystému pro nízkovýkonový mikrokontroler

Existuje velice velký výběr kryptosystému a jejich variant, některé mají široké využití, některé díky algoritmům a podstatě jejich vývoje, jsou vhodné jen například pro certifikaci. Pro mikrokontroler MSP430 bude vhodné vybrat kryptosystém, který nebude mít velké nároky na výpočetní kapacitu, tedy i velikost klíče a bude pomocí něj možné uskutečnit distribuci klíče do zařízeních, které budou následně využívat kryptosystém AES.

Symetrické	ECC	Asymetrické
80	163	1024
128	283	3072
192	409	7680
256	571	15360

Tab. 2.1: Porovnání velikosti klíče v bitech u různých kryptosystémů

Asymetrické kryptosystémy by byly pro mikrokontroler MSP430 příliš náročné. Z výše uvedené tabulky [25] můžeme vidět náročnost na velikost klíče při zachování

stejné úrovně bezpečnostni. Jsou zde srovnány symetrické kryptosystémy, asymetrické kryptosystémy a asymetrické kryptosystémy využívající eliptické křivky.

Pokud tedy uvažujeme pouze asymetrické systémy, je dobré se ještě podívat na výpočetní rychlost jednotlivých metod. RSA je znatelně pomalejší při použití RSA-3072 je výpočetní rychlost 184 ms a pro ECC-283 (tedy ECC stejné úrovně bezpečnosti) je výpočetní rychlost 29 ms. [2] Z toho tedy plyne, že výběr kryptosystémů bude mezi eliptickými křivkami.

Vzhledem k tomu, že eliptické křivky se snaží jen upravit vnitřní algoritmy jednotlivých kryptosystémů, máme na výběr stále z celé škály asymetrických kryptosystémů, bude tedy vhodné zabývat se těmi, které doporučuje National Security Agency (NSA) v mezinárodních standardech pro bezpečnost v balíčku Suite B.

Balíček Suite B doporučuje používání asymetrických kryptosystémů ECDSA a ECDH. Kryptosystém ECDSA je pouze upravený DSA, tedy kryptosystém určený pro podepisování. ECDH oproti němu je upravený D-H, který je určen pro přenos klíčů, bezpečnou distribuci klíčů. Z tohoto důvodu se budeme zabývat a volíme kryptosystém ECDH jako nejvhodnější.

2.4.1 Bližší rozbor D-H a ECDH

Kryptosystémy D-H a ECDH jsme si obecné vysvětlili již v předešlém textu. Nyní si tyto algoritmy popíšeme konkrétněji hlavně z matematického pohledu.

Diffie-Hellman

Problematiku D-H jsme si již vysvětlili, nyní si vysvětlíme D-H algoritmus pro výměnu klíčů. Dle tabulky

ALICE			BOB			Veřejnost
Kalkulace	Tajné		Kalkulace	Tajné		Veřejné
	a			b		p, g
$A \equiv g^a \bmod p$		$\rightarrow$			$\rightarrow$	A
			$B \equiv g^b \bmod p$		$\rightarrow$	B
$s \equiv B^a \bmod p$	$\rightarrow$ s	$=$	$s \equiv A^b \bmod p$	$\rightarrow$ s		

Tab. 2.2: Protokol Diffie-Hellman

Jak bylo řečeno, tak kryptosystém D-H je založen na složitosti výpočtu diskrétního logaritmu.

Obecné se dá algoritmus popsat následujícím způsobem (pro strany A, B)

- A zveřejňuje vybrané velké prvočíslo p a generátor g pro grupu $G = (Z_p^*; \cdot)$
- A volí SK_A (a) a spočítá VK jako $VK_A = g^a \bmod p$. VK_A je zveřejněn.
- B volí SK_B (b) a spočítá svůj VK jako $VK_B = g^b \bmod p$. VK_B je zveřejněn.
- A vypočítá $s \equiv VK_B^a \bmod p$
- B vypočítá $s' \equiv VK_A^b \bmod p$
- Čísla s, s' jsou totožná díky skutečnosti
  $s' = VK_A^b \bmod p = (g^a)^b \bmod p = (g^b)^a \bmod p = VK_B^a \bmod p = s$

Nyní si pro ukázku sprovozníme jednoduchý kód D-H s malými čísly. D-H algoritmus pro MSP430 je v příloze C. Nyní si vysvětlíme jen nejdůležitější části.

Funkce `prime()` je připravena pro dynamické vkládání vstupních parametrů, slouží pro kontrolu zda zvolené parametry jsou prvočísly. Druhá funkce `mod()` slouží pro zjednodušení matematických operací (konkrétně tedy pro D-H operace typu $x^y \bmod z$). Dále následuje již hlavní kód programu.

```
1   srand(time(NULL));
```

Ukázka kódu 2.7: Seed

Nastavení prvku náhodnosti pro funkci `srand()`. Dále volíme staticky prvočísla `p` a `q`. a náhodně parametry `a` a `b` viz kód níže.

```
1   p = 3;
2   g = 5;
3   a = rand() % 50;
4   b = rand() % 50;
```

Ukázka kódu 2.8: Nastavení základních parametrů

Jako poslední pak následuje výpočet parametrů A, B a klíčů K_A a K_B.

Diffie-Hellman s využitím eliptických křivek

Nejprve si přiblížíme eliptickou křivku. Eliptická křivka je hladká spojitá křivka, na které v nekonečnu leží bod O. Rovnice křivky je

$$y^2 + 2xy = ax^3 + bx^2 + cx + d, \tag{2.1}$$

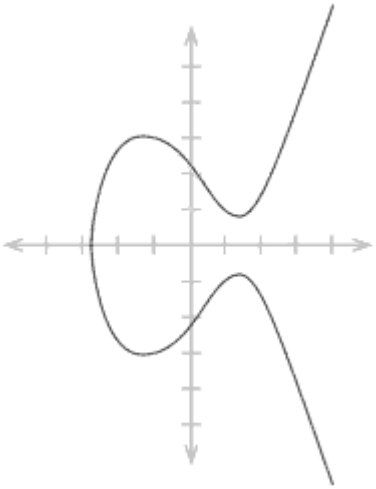
tato rovnice 2.1 se pak dále upravuje na Weierstrassův tvar

$$y^2 = x^3 + ax + b \tag{2.2}$$

a musí platit, že

$$0 \neq x^3 + ax + b, \tag{2.3}$$

pokud rovnice neplatí, křivka pak již není nesingulární (má ostrý bod) a nejedná se tedy o eliptickou křivku. Ukázku eliptické křivky můžeme vidět na obrázku 2.4.



Obr. 2.4: Eliptická křivka

Nyní podobně jako u D-H si na příkladu ukážeme vytváření klíče. Jako první je nutné dohodnout parametry (p, a, b, G, n, h). Číslo p volíme jako prvočíslo, (a, b) jsou konstanty z rovnice eliptické křivky (je tedy třeba dodržet rovnici eliptické křivky a dobře tyto parametry zvolit), bod G je bod na eliptické křivce, pak n je jeho řád a h je kofaktor, který udává podíl počtu prvků grupy bodů na eliptické křivce (grupa se dá jednoduše popsat jako množina o specifických vlastnostech). Po zvolení všech potřebných parametrů si každá strana volí své klíče. Tento klíč se sestavuje ze dvou částí, ze soukromého klíče d, ten je náhodně vygenerován jako celé číslo z intervalu [1; n-1] a veřejného klíče Q, kde platí

$$Q = dG. \tag{2.4}$$

První strana, nazveme si ji pro příklad třeba Alice bude mít (d_A, Q_A) a druhá strana, nazveme si ji pro příklad třeba Bob bude mít (d_B, Q_B). Poté proběhne výměna veřejných klíčů (Q_A, Q_B) a poté nalézá každá strana bod Z, kde pro tyto body platí, že

$$Z_A = d_A Q_B, Z_B = d_B Q_A \tag{2.5}$$

pro tyto rovnice platí

$$Z_A = Z_B \tag{2.6}$$

pro důkaz si do těchto rovnic dosadíme vzah 2.4

$$Z_A = d_A Q_B = d_A(d_B G) = d_A d_B G = d_B(d_A G) = d_B Q_A = Z_B \tag{2.7}$$

a tímto jsme ověřili, že vztah 2.6 skutečně platí a prakticky si šifru ECDH. Nyní si ukážeme příklad výpočtu ECDH. Mějme dvě strany Alici a Boba a rovnici eliptické křivky (viz. vztah 2.8)

$$y^2 = x^3 + 2x + 1. \tag{2.8}$$

Potom body na eliptické křivce jsou P_1(0,1), P_2(1,3), P_3(3,3), P_4(3,2), P_5(1,2), P_6(0,4) a tak dále. Pokud Alice zvolí tajný klíč $P_{TA} = 2$, pak veřejný klíč (P_{VA}) vypočítá dle vztahu 2.9

$$P_{VA} = P_{TA} P_1 = 2P_1 = P_1 + P_1 = P_2 = (1, 3) \tag{2.9}$$

Následně Bob zvolí svůj tajný klíč $P_{TB} = 3$ a spočte svůj veřejný klíč (P_{VB}) pomocí vztahu

$$P_{VB} = P_{TB} P_1 = 3P_1 = P_1 + P_1 + P_1 = P_3 = (3, 3) \tag{2.10}$$

Následně se spočte globální tajný klíč, tedy

$$P_{AB} = P_{TB} P_{VA} = P_{TA} P_{VB} = 3P_2 = 2P_3 = P_6 = (0, 4) \tag{2.11}$$

A tím je proces výpočtu hotov. Ukážeme jak implementovat ECDH pro malá čísla do MSP430. Kompletní kód je v příloze D. Přistoupíme k jednotlivým částem kódu.

```
1        a = 1;
2        b = 1;
```

Ukázka kódu 2.9: Parametry křivky

Proměnné (`a`, **b**) určují parametry eliptické křivky. Poté následuje cyklus naplňující pole bodů z eliptické křivky viz. kód níže.

```
1    for(i=0;i<100;i++) {
2            ECC[i] = i^3 + a*i + b;
3          }
```

Ukázka kódu 2.10: Pole bodů na eliptické křivce

Z tohoto pole následně volíme jeden bod, který dále používáme pro výpočet tedy

```
1       X = rand(100);
2       BODX = ECC[X]
3       TKA = rand() % 50;
4   ...
5       GKB = TKB * X;
```

Ukázka kódu 2.11: Výpočty klíčů

Proměnné (`TKA`, `TKB`) jsou náhodně zvolené tajné klíče a (`VKA`, `VKB`) jsou výpočty veřejných klíčů. Následně jsou vypočteny globální klíče (`GKA`, `GKB`), porovnány v podmínce `if`. Proměnné (`BODX`, `BODY`) určují výsledný globální klíč jako GK(BODX, BODY).

2.5 Implementace ECDH

Implementace ECDH skýtá mnohá úskalí. Je nutné se vypořádat s velkými čísly, které samotný mikrokontroler plně nepodporuje (datové typy jsou nedostatečně velké). Z tohoto důvodu je třeba vytvořit vlastní reprezentaci velkých čísel a k nim poté vytvořit funkce matematických operací, které se v ECDH používají. Dále pak kryptosystém ECDH k volbě svých parametrů potřebuje generátor náhodných čísel a následně samozřejmě samotná algoritmizace a implementace ECDH.

2.5.1 Problematika velkých čísel

Jelikož tedy klasické datové typy `int`, `long` aj. jsou nedostatečné svou velikostí, je nutné vytvořit datovou strukturu, která bude velká čísla reprezentovat. Tedy to jak čísla budeme ukládat do paměti, to jakým způsobem s nimi budeme pracovat a jak k nim budeme zpětně přistupovat.

Možností jak postupovat je mnoho, ověřeným způsobem je přistupovat k velkým číslům pomocí vlastní struktury s ukazatelem na místo v paměti (podobný postup se využívá například v OpenSSL, gcrypt, gmp) [21].

Použitá struktura vypadá takto

```
struct bignum_st
{
        BN_ULONG *d;
        int top;
        int dmax;
        int neg;
        int flags;
        }
```

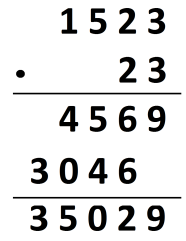
Ukázka kódu 2.12: Struktura pro velká čísla

Význam jednotlivých proměnných je následující, `BN_ULONG *d` je ukazatel pole, `top` je proměnná sloužící k ukládání do pole (pozice poslední uložené hodnoty v poly inkremenována o jedna), `dmax` je velikost pole (maximalní), `neg` pokud je nastaven, tak se jedná o záporné číslo a `flags` slouží k dalším nastavením. Deklarace velkých čísel (v příkladu 16-bitové číslo 0x1111) můžeme vidět v kódu níže.

```
struct bignum_st
{  #include "bn_lib.h"
            ...
    BN_ULONG a[]={0x1111,0x0000,0x0000,0x0000};
            ...
    BIGNUM bn_a;
     ...
    bn_a.d=a;
    bn_a.dmax=4;
    bn_a.top=1;
    bn_a.neg=0;
          }
```

Ukázka kódu 2.13: Deklarace velkých čísel

Díky použití vlastní interpretace velkých čísel je následně nemožné použít základní matematické operátory +, -, *, / a %. Pro matematické operace s velkými čísly je nutné vytvořit i vlastní matematické funkce. Opět je vhodné inspirovat se v již hotových návrzích, kdy existuje několik základních metod, školská metoda používá známých algoritmů, které se používají při matematických úkonech pod sebou (Obr. 2.5 nebo opět vycházet z otevřených algoritmů, který využívá například OpenSLL [21].



Obr. 2.5: Ukázka školské metody - násobení

V našem případě díky programátorským přednostem budeme vycházet z již navržených a popsaných algoritmů OpenSSL. Pro naše podmínky potřebujeme zajistit klasické sčítání, odčítání, násobení, dělení, modulo (umocňování vzhledem pouze k druhé mocnině v algoritmech ECDH jsme schopni zařídit pomocí algoritmů pro násobení). [22]

Použité funkce (resp. jejich hlavičky) s popisem můžeme vidět na kódu níže

```
1  int BN_add(BIGNUM *r, const BIGNUM *a, const BIGNUM *b);
2  % funkce (a+b)
3  int BN_sub(BIGNUM *r, const BIGNUM *a, const BIGNUM *b);
4  % funkce (a-b)
5  int BN_mul(BIGNUM *r, BIGNUM *a, BIGNUM *b, BN_CTX *ctx);
6  % funkce (a*b)
7  int BN_div(BIGNUM *dv, BIGNUM *rem, const BIGNUM *a,
8   const BIGNUM *d, BN_CTX *ctx); //funkce (a/b)
9  int BN_mod(BIGNUM *rem, const BIGNUM *a,
10  const BIGNUM *m, BN_CTX *ctx); //funkce (a mod b)
11 int BN_mod_add(BIGNUM *ret, BIGNUM *a, BIGNUM *b,
12  const BIGNUM *m, BN_CTX *ctx); //funkce ((a+b) mod m)
13 int BN_mod_sub(BIGNUM *ret, BIGNUM *a, BIGNUM *b,
14  const BIGNUM *m, BN_CTX *ctx); //funkce ((a-b) mod m)
15 int BN_mod_mul(BIGNUM *ret, BIGNUM *a, BIGNUM *b,
16  const BIGNUM *m, BN_CTX *ctx); //funkce ((a*b) mod m)
17 int BN_mod_div(BIGNUM *ret, BIGNUM *a, BIGNUM *b,
18  const BIGNUM *m, BN_CTX *ctx); //funkce ((a/b) mod m)
```

Ukázka kódu 2.14: Hlavičky matematických funkcí

Funkce použité v této práci jsou z projektu „Knihovna pro práci s modulární aritmetikou s velkými čísly na mikrokontrolérech řady MSP430“, který využívá funkce OpenSSL výše uvedené. [23] Popis jednolivých funkcí je v kódu a jejich použití vyplývá z jejich deklarace.

2.5.2 Generátor náhodných čísel

V rámci kryptosystému ECDH je nutné vytvořit náhodný generátor čísel (RNG). Je více variant, kterými lze RNG vytvořit. RNG se dělí na Hardwarové náhodné generátory a Softwarové náhodné generátory.

Hardwarové generátory budou přídavné moduly, které se připojují k mikrokontroleru. Tyto metody jsou například měření špičkových napětí v rozvodné síti, měření šumu, teplotní čidla ap.

Softwarové generátory nebo také generátory pseudonáhodných čísel (PNG) jsou generátory (algoritmy) vytvářející dlouhé řetězce čísel mající zdánlivě dobré náhodné rozdělení, ale později se tyto sekvence opakují a kvalita rozdělení se snižuje. Bývají tedy méně účinné než hardwarové generátory, ale naopak bývají rychlejší. Náhodnost čísel generovaných PNG bývá většinou dána výběrem metody a také seedem (počáteční řetězec bitů/čísel)

Použitý generátor náhodných čísel

V našem případě je vhodné využít integrovaných modulů přímo v MSP430, pomocí kterých jsme schopni navodit náhodné jevy a tak vytvořit náhodný generátor (hodiny ACLK a SMCLK integrované přímo v kitu). Analýza náhodnosti jevu byla provedena institucí Federal Information Processing Standards (FIPS) a uznána jako funkční pro RND [24] a o tuto skutečnost se budeme opírat při vývoji RNG. Použité prvky si nyní popíšeme a vysvětlíme oč se jedná, ukážeme si základní práci s hodinami (jejich nastavení, spuštění aj. viz. text níže) a na závěr vysvětlíme a popíšeme vytvoření náhodného generátoru.

Fyzické zapojení hodin microcontroleru (MCLK, SMCLK, ACLK) můžeme vidět na obrázku (Obr. 2.6). MCLK jsou master hodiny, které používá procesor (a několik dalších periferních zařízeních), tyto hodiny používají digitálně řízený oscilátor (DCO). Frekvence DCO je ovládána pomocí sady bitů v modulu registrů na třech úrovních k dispozici je několik frekvencí 1, 8, 12 a 16 MHz.

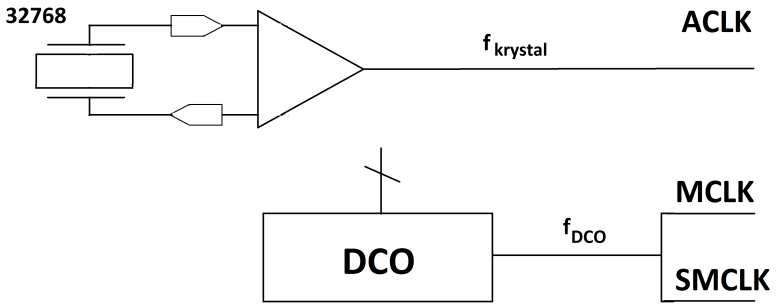
Nastavení DCO probíhá pomocí nastavení registrů (viz. kód níže)

```
1 CSCTL1 = CALBC1_1MHZ
2 DCOCTL = CALDCO_1MHZ
```

Ukázka kódu 2.15: Ukázka nastavení DCO na 1MHz

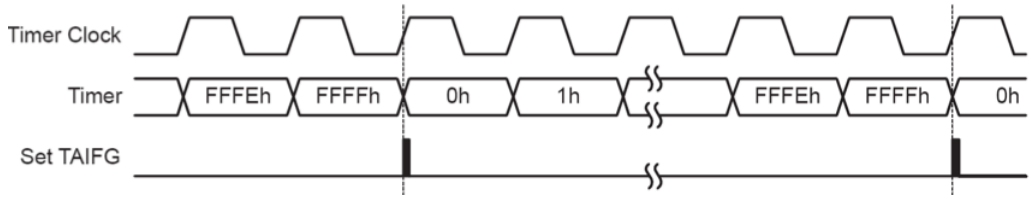
První řádek slouží k nastavení frekvence a druhý pak k nastavení kroku a modulace. Z Obrázku 2.6 je patrné, že hodiny SMCLK a MCLK používají obě DCO pro nastavení. ACLK pak využívá pevné frekvence krystalu (cca 12MHz).

Pro RND je (kromě hodin) ještě třeba užít čítačů (časovačů). Čítače fungují tak, že jsou poháněny hodinovým signálem a na každém taktu snižují nebo zvyšují



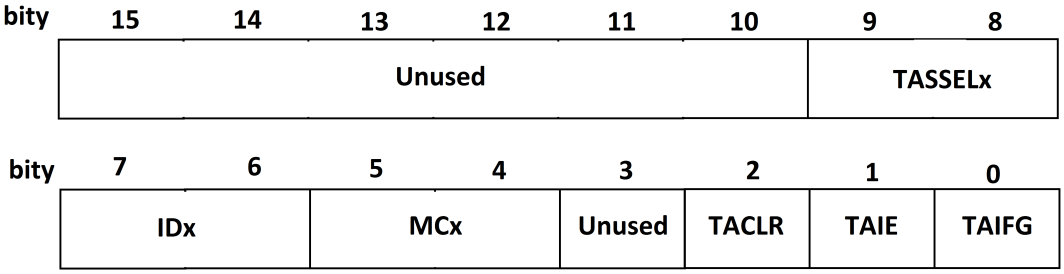
Obr. 2.6: Vnitřní zapojení hodin MSP430

svou hodnotu a po dosažení určité hodnoty lze časovač přerušit. Časovač má tři základni prvky. Prvním je vstup (obvykle hodiny), dále čítač a přerušení (v MSP430 se používá pro označení přerušení TAIFG). Vysvětlení fungování čítače můžeme vidět na obrázku 2.7, hodiny o frekvenci FFFFh Hz a limitu FFFFh.



Obr. 2.7: Ukázka funkce čítače a přerušení

Konkrétní čítače v MSP430 jsou TIMER_A a TIMER_B. Pro naše účely nám plně dostačuje jeden časovač, tedy TIMER_A (jejich nastavení je obdobné, používají pouze jiné registry). Registr pro Timer_A můžeme vidět na obrázku 2.8.



Obr. 2.8: Registr pro TIMER_A

Unused (bity 10 až 15 a bit 3) jsou nepoužívané bity. TASSELx (bity 8 až 9) slouží pro volbu zdroje pro Timer_A (00 TACLK, 01 ACLK, 10 SMCLK a 11 INCLK). IDx (bity 6 až 7) slouží pro výstup. MCx (bity 4 až 5) slouží pro nastavení Timer_A (00 stop-mode, 01 up-mode, 10 continuous-mode a 11 up/down mode). TACLR (bit 2) slouží pro vynulování pro Timer_A. TAIE (bit 1) a TAIFG (bit 0) slouží pro povolení/zakázání přerušení. Použití můžeme vidět na jednoduchém kódu v Příloze E.

My si nyní popíšeme důležité části programu.

```
1        TA0CCR0 = 1000000;
2        TA0CCTL0 = 0x10;
3        TA0CTL = TASSEL_2 + MC_1;
```

Ukázka kódu 2.16: Nastavení čítače

`TA0CCR0` nastavuje limit čítače, `TA0CCTL0` je zapnutí přerušení (tedy bit 4 = 1) a `TA0CTL` nastavuje parametry čítače. `TASSEL_x` je nastavení frekvence a hodin (v tomto případě `TASSEL_2` je SMCLK s 1MHz, ještě například můžeme použít `TASSEL_1` pro ACLK s 12KHz apod.) a MC_1 nastavuje čítání nahoru. Dále pak následuje globální nastavení

```
1        eint();
2        LPM0;
```

Ukázka kódu 2.17: Globální nastavení

Tedy globální nastavení přerušení zařizuje funkce `eint()` a nízkovýkonový mód pro mikrokontroler pak `LPM0`. Toto lze ještě nastavit jednodušeji před `_BIS_SR(LPM0_bits + GIE)`. Poslední část programu je pak funkce sloužící pro funkčnost programu, zařízeno je to pomocí blikání diody.

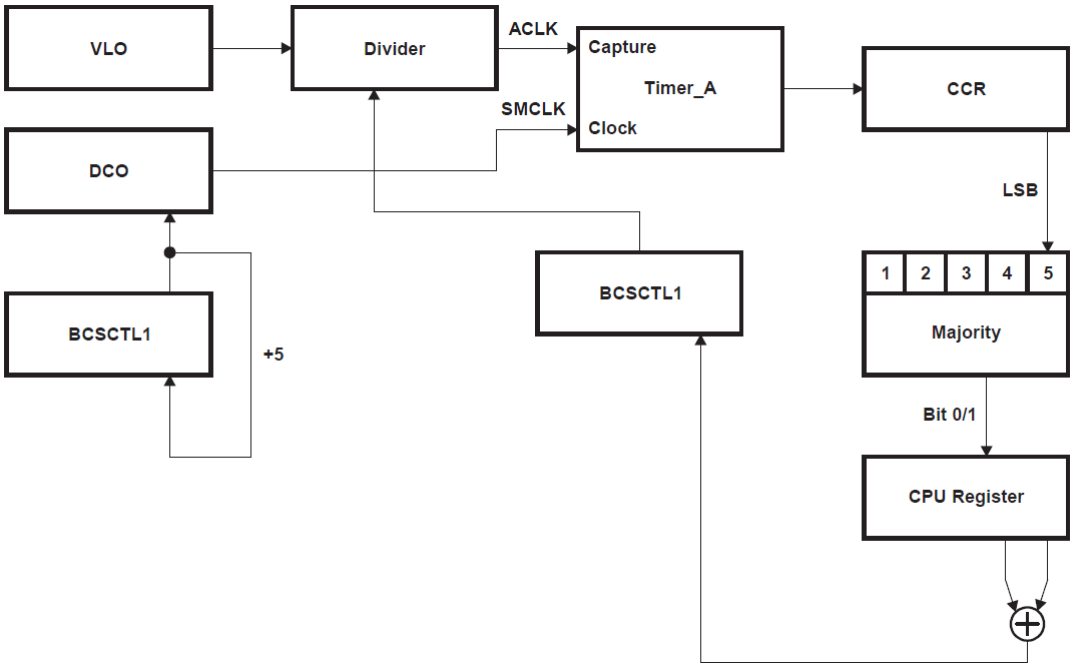
```
1 #pragma vector=TIMER0_A0_VECTOR
2    __interrupt void Timer0_A0 (void) {
```

Ukázka kódu 2.18: Ověřovací funkce

Ukázali jsme si práci s časovačem, čítačem a hodinami, v předešlém textu pak i jednoduchou práci s knihovnou. Ve všech ukázkách používáme k nastavení registry a globální proměnné. Tyto parametry (prvky) a jejich nastavení (či hodnoty), najdeme v knihovně MSP430f5438A.h, která se implementuje vždy na začátku programu. Tato knihovna tedy slouží pro základní nastavení pinů, registrů a globálních proměnných.

Nyní si již můžeme popsat vytvoření RNG pomocí těchto čítače a hodin. Základní nastavení a použité moduly můžeme pochopit z obrázku 2.9 Tedy TIMER_A bude používat SMCLK jako hodiny a ACLK pro přerušení. Výstupu tedy nejméně

významného bitu (LSB) se bude řídit politika přerušení. VLO je značení krystalu a DCO je tedy oscilátor pro SMCLK. BCSCTL1 je popsán výše. Pomocí LBS-bitů jsme pak schopni sestavovat jednotlivé x-bitové bloky náhodných bitů.



Obr. 2.9: Zapojení čítače TIMER_A

Pro zvýšení náhodnosti je třeba dodržet několik pravidel

- Pokaždém cyklu (cyklus je průběh algoritmu až do bodu vytvoření nového LSB) je LSB posunut a BCSCTL1 se přidává 5, tím způsobíme změnu rychlosti DCO vzhledem k VLO pokaždé smyčce (u BCSCTL1 je možné použít jakékoliv číslo, ale dle analýzy [24] bylo zjištěno, že číslo 5 dokazuje dostatečnou změnu vůčí VLO)
- Pokaždé když je LSB posunut jdou dva LSB z R12 registru pomocí XOR do tzv. DIVA bitů BCSCTL1. DIVA bity řídí dělič používaný VLO před tím než VLO vstupuje do TIMER_A. Což nám opét změní vztah DCO k VCO.

Při tomto postupu a dodržení předepsaných podmínek jsme schopni vytvořit náhodný jev tiků jednotlivých hodin vůči sobě.

Nyní přejdeme k samotnému kódu RNG. Ukázka kódu je v příloze F a my si nyní rozebereme funkci, která nám generuje náhodné bity.

```
1 unsigned int TACCTL0_old = TA0CCTL0;
2 unsigned int TACTL_old = TA0CTL;
```

Ukázka kódu 2.19: Uložení počátečního stavu

Tato část slouží pouze pro uložení počátečního stavu. Dále pak

```
TA0CCTL0 = CAP | CM_1 | CCIS_1;
TA0CTL = TASSEL_2 | MC_2;
```

Ukázka kódu 2.20: Nastavení čítače

`TA0CCTL0` nám nastavuje typ přerušení a nastavení zachytávacího módu (capture-mode) pro TIMER_A 0 (`CAP` má hodnotu 0x0100, `CM1` pak 0x8000 a `CCIS_1` je 0x4000). `TA0CTL` je pak nastavení vstupu TIMER_A 0, tedy `TASSEL_2` určuje SMCLK 1Mhz a `MC_2` pak mód čítání nahoru (continuous-mode up). Následuje další část programu a to Generování bitů.

```
for (i = 0; i < 16; i++) {
...
while (!(CCIFG & TA0CCTL0)); // Wait for interrupt
...
TA0CCTL0 = TACCTL0_old;
TA0CTL = TACTL_old;
...
return result;
}
```

Ukázka kódu 2.21: Generování bitů

Cyklus `for` je tedy již samotné generování bitů. Tento cyklus určuje, že konečné generované číslo bude mít 16-bitů. Cyklus `while` je pak samotné přerušení (viz. teoretický základ v předešlém textu). Poté je uložen LSB a TIMER_A je resetován do počátečního stavu. V proměnné `result` je pak uloženo náhodné 16-bitové číslo.

2.5.3 Fáze implementace ECDH

Konečná implementace je pak již jen o spojení vytvořených metod do jednoho funkčního celku. To je třeba provést dle potřeb uživatele (výrobce), kdy je třeba stanovit velikost generovaného klíče (počet period generování) aj. parametry, které se stanový při vývoji či výrobě. Je možné dle funkcí a kódů vytvořit funkční kryptosystém ECDH pro nízkovýkonový mikrokontroler MSP430. Nicméně díky náhodnému generátoru, metod pro aritmetiku s velkými čísly a otevřenosti vytvořených kódů je možné implementovat jakoukoliv symetrickou nebo asymetrickou metodu, ke které máme matematický popis.

Pro ukázku je uveden v příloze G příklad ECDH kryptosystému s velkými čísly. Popis kódu plyne z textu celé práce, samotný kód je předělaný algoritmus s malými čísly do podoby ECDH s velkými čísly. Jedná se o 192-bitový ECDH, parametry (a, b) pro eliptickou křivku jsou nastaveny staticky, stejně jako výběr prvního

bodu na eliptické křivce. Klíče jsou pak voleny náhodně pomocí generátoru `rand()` (v 12ti kolech, tedy 12 krát 16-bitů je 192-bitů). Výsledný klíč má pak podobu (bn_{BODX}, bn_{BODY}).

2.5.4 Popis přiloženého souboru a práce s ním

Jako poslední si vysvětlíme práci se souborem `blink.c`. Tento soubor obsahuje veškeré algoritmy a kódy, které jsou uvedeny zde v textu. Přehledně seřazené a rozdělené. Dále obsahuje i sekci TRASHCAN (koš). Tato sekce obsahuje kódy, které nejsou v práci popsány. Tyto kódy nejsou zprovozněny a převážně se jedna o kódy, které sloužili k vývoji programu.

Soubor není spustitelný ve windowsech a neobsahuje uživatelské rozhraní, je nutné mít k dispozici vývojové prostředí Code Composer Studio (verze na které byly kódy programovány je v5.2.1.00018) a mikrokontroler MSP430f5438A. Dále vzhledem k absenci uživatelského rozhraní je nutné pracovat s jednotlivými registry MSP430 (jejich kontrola probíhá pomocí CCS po překladu a spuštení programu).

3 ZÁVĚR

Byl vypracován rozbor kryptografie. Jsou popsány dnes používané asymetrické a symetrické kryptosystémy. Dále je nastíněna problematika nízkovýkonových mikrokontrolerů a kryptosystémů, které se u nich používají. Práce je převážně zaměřena na nejpoužívanější symetrický kryptosystém AES a mikrokontroler MSP430.

Po teoretickém úvodu do terminologie a problematiky je navrženo řešení problému distribuce klíču u AES za pomocí asymetrických šifer. Z analýzy a specifik jednotlivých asymetrických kryptosystémů je pak vybrán kryptosystém ECDH pro který je zpracována praktická část práce. Dále je pak vypracován popis MSP430 společně s jeho ovládáním a ovládáním vývojového prostředí CCS, k tomu jsou použity jednoduché ukázky kódů.

V poslední části práce je vypracován návrh implementace ECDH do MSP430, popsána a prakticky vysvětlena je práce s velkými čísly, dále pak je popsán, vybrán a vytvořen náhodný generátor a nakonec je vysvětlen postup implementace vytvořených kódu a funkcí. Z praktické části vyplynulo, že vytvořené metody/kódy nemusí sloužit pouze pro implementaci ECDH, ale i k jiným účelům či implementaci jiných kryptosystémů, dle požadavků uživatele. Pro ukázku je nakonec vytvořen i kód pro vybraný kryptosystém ECDH se 192-bitovým klíčem.

LITERATURA

[1] BURDA, K. *Bezpečnost informačních systémů* [Skripta FEKT VUT v Brně]. 2005, poslední aktualizace 1. 11. 2005 [cit. 29. 05. 2013].

[2] VANSTONE, S. *ECC Holds Key to Next-Gen Cryptography* [online]. 2004 [cit. 29. 05. 2013].

Dostupné z URL: <http://www.design-reuse.com/articles/7409/ecc-holds-key-to-next-gen-cryptography.html>.

[3] FRÁŇA, J., MIKULECKÝ, P. a SKALSKÝ, M. *Popis postupu šifrování pomocí algoritmu RSA* [online]. 2003, poslední aktualizace 20. 03. 2003 [cit. 29. 05. 2013].

Dostupné z URL: <http://kryptologie.uhk.cz/6.htm>.

[4] BLEICHENBACHER, D., BOSMA, W. a LENSTRA, A. *Advances in Cryptology Crypto* [strany 386-396]. 1995 [cit. 29. 05. 2013].

[5] BRUMLEY, B.B. a TUVERY N. *Remote Timing Attacks are Still Practical* [Cryptology ePrint Archive: Report 2011/232]. 2011 [cit. 29. 05. 2013].

Dostupné z URL: <http://eprint.iacr.org/2011/232>.

[6] RYBKA, Š. *Skupina algoritmů NSA Suite-B Cryptography* [bakalářská práce]. 2011 [cit. 29. 05. 2013].

Dostupné z URL: <http://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=42057>.

[7] QUIRING, K. *MSP430 Timers In-Depth* [PDF prezentace]. 2006 [cit. 29. 05. 2013].

Dostupné z URL: <http://www.ti.com/lit/ml/slap113/slap113.pdf>.

[8] WEISSTEIN, E.W. *Discrete Logarithm* [WEB prezentace]. [cit. 29. 05. 2013].

Dostupné z URL: <http://mathworld.wolfram.com/DiscreteLogarithm.html>.

[9] EMVCO, LLC. *EMV 4.2 Specifikace Kniha 2 - Bezpečnost a správa klíčů, Verze 4.2* [online]. 2008 [cit. 29. 05. 2013].

Dostupné z URL: <http://www.emvco.com/specifications.aspx?id=155>.

[10] ESCAPA, D. *Encryption for Password Protected Sections* [online]. 2006 [cit. 29. 05. 2013].

Dostupné z URL: <http://blogs.msdn.com/b/descapa/archive/2006/11/09/encryption-for-password-protected-sections.aspx>.

[11] MICROSOFT, Co. *Encrypt e-mail messages* [online]. 2007 [cit. 29. 05. 2013].

Dostupné z URL: <http://office.microsoft.com/en-us/outlook-help/encrypt-e-mail-messages-HP001230536.aspx>.

[12] MICROSOSFT, Co. *Technical Reference for Cryptographic Controls Used in Configuration Manager* [online]. 2012, poslední aktualizace 01. 01. 2013 [cit. 29. 05. 2013].

Dostupné z URL: <http://technet.microsoft.com/en-us/library/hh427327.aspx>.

[13] Westlund, H.B. *NIST reports measurable success of Advanced Encryption Standard* [online]. 2002, [cit. 29. 05. 2013].

[14] SHAMIR, A. *Hiding information and signatures in trapdoor knapsacks* [Information Theory, IEEE Transactions on 24 (5): 525–530]. 1978, [cit. 29. 05. 2013].

[15] FIPS, Pub. *Digital Signature Standard (DSS)* [FIPS PUB 186-3]. 2009, [cit. 29. 05. 2013].

[16] NIST, Pub. *NIST Special Publication 800-57* [Part 1]. 2007, [cit. 29. 05. 2013].

[17] ALCOM, El. *Contact secure microcontrollers 8-/16-bit risc CPU with PKI* [Online]. 2013, [cit. 29. 05. 2013].

Dostupné z URL: <http://www.alcom.be/binarydata.aspx?type=doc/INSIDE_C.pdf>.

[18] ATMEL, Co. *32-bit AVR UC3: The world's most efficient 32-bit microcontroller* [Online]. 2013, [cit. 29. 05. 2013].

Dostupné z URL: <http://www.atmel.com/products/microcontrollers/avr/32-bitavruc3.aspx>.

[19] ATMEL, Co. *SAM4L: Redefining Low Power in Cortex-M4 Processor-based MCUs* [Online]. 2013, [cit. 29. 05. 2013].

Dostupné z URL: <http://www.atmel.com/products/microcontrollers/arm/sam4l.aspx>.

[20] INSIDE, Sec. *Secure micro products* [Online]. 2013, [cit. 29. 05. 2013].

Dostupné z URL: <http://www.insidesecure.com/eng/Products/Secure-Microcontrollers/Secure-Micro-Products>.

[21] LEITNER, F. *Bignum Arithmetic* [Online]. 2006, [cit. 29. 05. 2013].

Dostupné z URL: <http://www.insidesecure.com/eng/Products/Secure-Microcontrollers/Secure-Micro-Products>.

[22] OpenSSL, Pr. *Cryptography and SSL/TSL Toolkit* [Online]. 2013, [cit. 29. 05. 2013].

Dostupné z URL: <http://www.openssl.org/docs/crypto/bn.html>.

[23] RÁŠO, O., MLÝNEK, P., KOUTNÝ, M. a MIŠUREC, J. *Knihovna pro práci s modulární aritmetikou s velkými čísly na mikrokontrolérech řady MSP430* [Online]. 2013, [cit. 29. 05. 2013].

Dostupné z URL: <http://www.utko.feec.vutbr.cz/~raso/libMSP430_aritmetika.html>.

[24] WESTLUND, L. *Random Number Generation Using the MSP430* [Application Report SLAA335]. 2006, [cit. 29. 05. 2013].

[25] LAUTER, K. *The Advantages of Elliptic Curve Cryptography for Wireless Security* [online]. 2004, [cit. 29. 05. 2013].

Dostupné z URL: <http://research.microsoft.com/en-us/um/people/klauter/ieeefinal.pdf>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

K_E Šifrovací klíč

K_D Dešifrovací klíč

NIST Institut norem a standardů

NIST Dalsi z instituci pro standardy

DES Symetrický kryptosystém, dnes považován za nebezpečný

3DES Upravená varianta kryptosystému DES, dnes ještě stále používána

AEA Jeden z nejpoužívanějších symetrických kryptosystémů, varianty dle velikosti klíče např. AES-128 pro velikost klíče 128 bitů

AES Standard kryptosystému AES

MHK Asymetrický kryptosystém Merkle-Hellman Knapsack, dnes považován za nebezpečný

D-H Asymetrický kryptosystém Diffie-Hellman, využívá problematiky diskrétního logaritmu

RSA Asymetrický kryptosystém, využívá problematiky faktorizace

DSA Asymetrický kryptosystém digitálního podpisu, Digital Signature Algorithm

DSS Standard kryptosystému pro digitální podpis, Digital Signature Standard

LUC Asymetrický kryptosystém, jenž užívá Lucasových funkcí

EC Elliptic curve, označení pro eliptickou křivku

ECC Elliptic curve cryptography, užívá se pro eliprické křivky v krpytografii

IES Jeden z hybridních kryptosystémů, existuje ve dvou variantách DLIES pro diskrétní logaritmus a ECIES pro eliptické křivky

QV Qu-Vanstone nad eliptickou křivkou, slouží pro certifikaci či autentizaci

MQV Menezes-Qu-Vanstone nad eliptickou křivkou, tento systém má bezpečnostní vady

CCS Vývojové prostředí pro výrobky od Texas Instruments, Code Composer Studio

MIN Muž uprostřed, kryptografický problém při komunikaci dvou účastníků

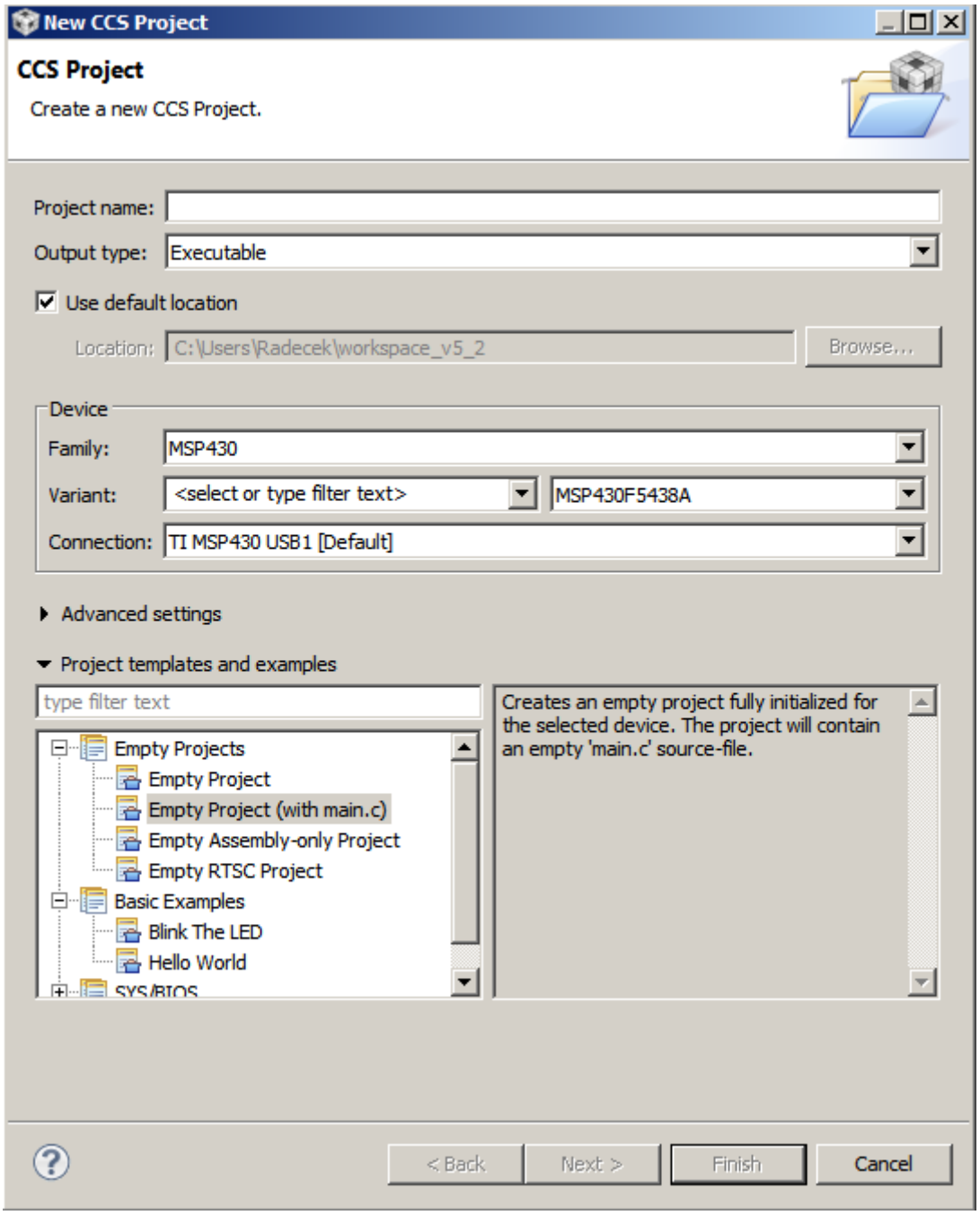
MIPS Výkonová jednotka dnešních počítačů, udává počet instrukcí za sekundu

RNG Generátor náhodných čísel

PNG Generátor pseudo-náhodných čísel

SEZNAM PŘÍLOH

A NASTAVENÍ CSS PROJEKTU



B OVLÁDÁNÍ DIODY

Ukázka kódu B.1: Výpis souboru main.c

```
1  #include <msp430f5438A.h>
2
3  void main(void) {
4
5          WDTCTL = WDTPW + WDTHOLD;
6          P1DIR |= 0x41;
7          P1OUT = 1;
8
9          while(1)
10         {
11                 volatile unsigned int i;
12                 P1OUT ^= 0x41;
13
14                 i = 5000;
15                 do (i--);
16                 while (i != 0);
17
18                 P1OUT ^= 0x41;
19         }
20 }
```

C D-H ALGORITMUS PRO MSP430

Ukázka kódu C.1: Výpis souboru main.c

```
1  #include <stdio.h>
2  #include <math.h>
3  #include <stdlib.h>
4  #include <time.h>
5
6  int prime(int num) {
7      int i = 0, j = 0;
8      for (i = 2; i < num/2; ++i)
9          if (num % i == 0)
10             return 0;
11     return 1;
12 }
13
14 int mod(int base, int expo, int num) {
15     int res = 1;
16     int i;
17     for (i = 1; i <= expo; ++i)
18         res = (res * base) % num;
19     return res;
20 }
21
22 int main() {
23     int p, g, a, b, i, j, A, B, KA, KB;
24         srand(time(NULL));
25         p = 3;
26         g = 5;
27         srand(time(NULL));
28         a = rand() % 50;
29         b = rand() % 50;
30         A = mod(g, a, p);
31         B = mod(g, b, p);
32         KA = mod(r2, a, p);
33         KB = mod(r1, b, p);
34     }
35 }
```

D ECDH PRO MSP430 S MALÝMI ČÍSLY

Ukázka kódu D.1: Výpis souboru main.c

```
1  #include <stdio.h>
2  #include <math.h>
3  #include <stdlib.h>
4  #include <time.h>
5
6  int main() {
7      int a, b, i, BODX, BODY, VKA, VKB, TKA, TKB, KAB, ECC[100], X;
8
9          srand(time(NULL));
10
11         a = 1;
12         b = 1;
13
14         for(i=0;i<100;i++) {
15          ECC[i] = i^3 + a*i + b;
16          }
17
18         X = rand(100);
19         BODX = ECC[X]
20
21         TKA = rand() % 50;
22         TKB = rand() % 50;
23
24         VKA = TKA * X;
25         VKB = TKB * X;
26
27         GKA = TKA * VKB;
28         GKB = TKB * VKA;
29
30         if (GKA == GKB) {
31           BODX = GKA;
32           BODY = BODX^3 + a*BODX + b;
33             }
34
35         else return 0;
```

```
36 |         return 0;
37 | }
```

E PRÁCE S ČÍTAČEM

Ukázka kódu E.1: Výpis souboru main.c

```
1  #include <msp430f5438A.h>
2
3  void main(void) {
4          WDTCTL = WDTPW + WDTHOLD;
5          P1DIR |= BIT0;
6          P1OUT &= ~BIT0;
7
8          TA0CCR0 = 1000000;
9          TA0CCTL0 = 0x10;
10         TA0CTL = TASSEL_2 + MC_1;
11
12         _BIS_SR(LPM0_bits + GIE);
13 }
14
15 #pragma vector=TIMER0_A0_VECTOR
16     __interrupt void Timer0_A0 (void) {
17
18         P1OUT ^= BIT0;
19 }
20
21 }
```

F NÁHODNÝ GENERÁTOR ČÍSEL

Ukázka kódu F.1: Výpis souboru main.c

```
1  #include <msp430f5438.h>
2  #include <stdio.h>
3  #include "rand.h"
4
5  unsigned int rand() {
6  int i, j;
7  unsigned int result = 0;
8
9  unsigned int TACCTL0_old = TA0CCTL0;
10 unsigned int TACTL_old = TA0CTL;
11
12 TA0CCTL0 = CAP | CM_1 | CCIS_1;
13 TA0CTL = TASSEL_2 | MC_2;
14
15 for (i = 0; i < 16; i++) {
16 unsigned int ones = 0;
17
18 for (j = 0; j < 5; j++) {
19 while (!(CCIFG & TA0CCTL0));
20
21 TA0CCTL1 &= ~CCIFG;
22
23 if (1 & TA0CCR0)
24 ones++;
25     }
26
27 result >>= 1;
28
29 if (ones >= 3)
30 result |= 0x8000;
31   }
32
33 TA0CCTL0 = TACCTL0_old;
34 TA0CTL = TACTL_old;
35 return result;
```

```
36 }
37
38 unsigned int prand(unsigned int state) {
39 return (M * state + I);
40 }
41
42 int main(void) {
43
44         int ii;
45         int a;
46
47         WDTCTL = WDTPW + WDTHOLD;
48         TA0CCR0 = 1000000;
49         TA0CCTL0 = 0x10;
50         TA0CTL = TASSEL_2 + MC_1;
51
52         _BIS_SR(LPM0_bits + GIE);
53
54         ii=0;
55         ii = rand();
56         for (a=0;a<1000;a++) {
57         }
58 }
59
60 }
```

G ECDH PRO MSP430 S VELKÝMI ČÍSLY

Ukázka kódu G.1: Výpis souboru main.c

```
1  #include <msp430f5438.h>
2  #include <stdio.h>
3  #include "rand.h"
4  #include "bn_lib.h"
5
6  int main() {
7      int ret, i;
8
9     BN_ULONG a[]    ={0x1111,0x0000,0x0000,0x0000,0x0000,0x0000,
10                       0x0000,0x0000,0x0000,0x0000,0x0000,0x1111,};
11     ...
12    // kód kvůli velikost zkrácen, zde jen deklarace proměnných
13    // celý kód je pak k dispozici na CD/DVD
14    BN_ULONG GK[]   ={0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
15                       0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,};
16
17    BIGNUM bn_a;
18    BIGNUM bn_b;
19
20    bn_a.d=a;
21    bn_a.dmax=12;
22    bn_a.top=12;
23    bn_a.neg=0;
24          ...
25    // kód kvůli velikost zkrácen, zde jen deklarace proměnných
26    // celý kód je pak k dispozici na CD/DVD
27    bn_bodx.d=GK;
28    bn_bodx.dmax=12;
29    bn_bodx.top=0;
30    bn_bodx.neg=0;
31
32    for(i=0;i<12;i++) {
33     bn_TKA[i] = rand();
34     bn_TKA.top++;
35    }
```

```
36
37      for(i=0;i<12;i++) {
38       bn_TKB[i] = rand();
39       bn_TKB.top++;
40      }
41
42      ret = BN_mul(&bn_VKA, &bn_TKA, &bn_BODX, ctx1);
43      ret = BN_mul(&bn_VKB, &bn_TKB, &bn_BODX, ctx1);
44
45      ret = BN_mul(&bn_GKA, &bn_TKA, &VKB, ctx1);
46      ret = BN_mul(&bn_GKB, &bn_TKB, &VKA, ctx1);
47
48      ret = 1;
49      ret = BN_cmp(&bn_GKA, &bn_GKB);
50      if (ret == 0) {
51            bn_BODX = bn_GKA;
52            ret = BN_mul(&bn_POM1, &bn_BODX, &bn_BODX, ctx1);
53            ret = BN_mul(&bn_POM1, &bn_POM1, &bn_BODX, ctx1);
54            ret = BN_mul(&bn_POM2, &bn_BODX, &bn_a, ctx1);
55            ret = BN_(&bn_BODY, &bn_BODY, &bn_POM1, ctx1);
56            ret = BN_(&bn_BODY, &bn_BODY, &bn_POM2, ctx1);
57            ret = BN_(&bn_BODY, &bn_BODY, &bn_b, ctx1);
58             }
59            else return 0;
60
61            if (GKA == GKB) {
62              BODX = GKA;
63              BODY = BODX^3 + a*BODX + b;
64                }
65
66            else return 0;
67            return 0;
68 }
```