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

ZABEZPEČENÍ KOMUNIKACE VESTAVĚNÉHO SYSTÉMU

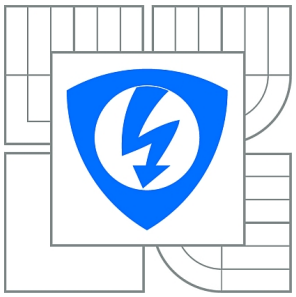
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE LUBOŠ RÁCEK
AUTHOR

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

ZABEZPEČENÍ KOMUNIKACE VESTAVĚNÉHO SYSTÉMU

SECURE COMMUNICATION IN EMBEDDED SYSTEMS

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

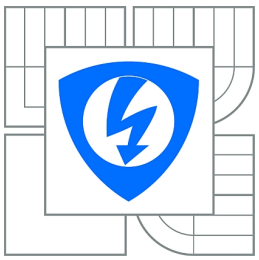
AUTOR PRÁCE — LUBOŠ RÁCEK

AUTHOR

VEDOUCÍ PRÁCE — Ing. VLADIMÍR ČERVENKA

SUPERVISOR

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student: Luboš Rácek ***ID:*** 148148
Ročník: 3 ***Akademický rok:*** 2013/2014

NÁZEV TÉMATU:

Zabezpečení komunikace vestavěného systému

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je podrobně analyzovat energetickou náročnost asymetrické kryptografie na výpočetně omezených zařízeních, společně s návrhem a implementací zabezpečení komunikace s pomocí hardwarových akcelerátorů s ohledem na nízkou spotřebu energie. Vytvořený systém bude porovnán s energetickou náročností dostupných softwarových řešení.

DOPORUČENÁ LITERATURA:

[1] KLEIDERMACHER, D., KLEIDERMACHER, M. Embedded systems security: practical methods for safe and secure software and systems development. 1st ed. Amsterdam: Elsevier, 2012, 396 p. ISBN 01-238-6886-6.

[2] H. K. PATIL, Stephen A. Security for wireless sensor networks using identity-based cryptography. S.l.: Auerbach Publishers, Incorporated. ISBN 14-398-6901-4.

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

Vedoucí práce: Ing. Vladimír Červenka
Konzultanti bakalářské práce:

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

UPOZORNĚNÍ:

ABSTRAKT

Bakalářská práce se zaměřuje na návrh a analýzu protokolu pro zajištění zabezpečené komunikace s využitím hardwarového akcelerátoru. Protokol bude navržen s ohledem na nízkou spotřebu a malý výpočetní výkon použitého mikrokontroléru. Při návrhu bude zohledněna možnost využití asymetrické kryptografie.

KLÍČOVÁ SLOVA

omezená zařízení,asymetrická kryptografie, kryptografie elyptických křivek, akcelerátor, autentizační modely, mikrokontrolér

ABSTRACT

Bachelor's thesis focuses on a design and analysis of protocol to provide secure communication using a hardware accelerator. The protocol will be designed with low power consumption and small computing power of a microcontroller. Asymetric cryptography will be taken into account in the design.

KEYWORDS

limited devices, asymmetric cryptography, elliptic curve cryptography, accelerator, authentication models, microcontroller

RÁCEK, Luboš *Zabezpečení komunikace vestavěného systému*: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2014. 60 s. Vedoucí práce byl Ing. Vladimír Červenka

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Zabezpečení komunikace vestavěného systému“ 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 práce, panu Ing. Vladimíru Červenkovi za velmi užitečnou metodickou pomoc, konzultace, a cenné rady při zpracování bakalářské práce.

Brno ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Cílem práce je zajištění bezpečné komunikace dvou zařízení s omezeným zdrojem energie a výpočetním potenciálem. Zajištění bezpečné komunikace znamená ustanovení základních pravidel pro zahájení komunikace. Abychom zabránili nechtěné komunikaci s cizími zařízeními, je nutné provést ověření jeho identity, neboli autentizaci. Proces autentizace musí být na takové úrovni, aby nedošlo k jeho narušení, ani možnosti okopírování. Autentizace musí být jednoznačná a pro každý případ jedinečná. Je proto nutné k autentizaci zavést kryptografii. Kryptografie zajistí určitý stupeň bezpečnosti autentizace, odvozený od druhu použité kryptografické metody. Vyšší stupeň bezpečnosti v kryptografii znamená použití složitějších šifrovacích metod. V informatice se používají matematické algoritmy, díky kterým se dá nastavit stupeň obtížnosti výpočtu algoritmu podle dostupných výpočetních prostředků. V tomto případě jsme omezeni výpočetním výkonem použitého systému, je proto potřeba najít vhodné řešení, které by splnilo požadavky na úroveň zabezpečení. Hlavním požadavkem je energetická nenáročnost procesu autentizace kvůli omezenému energetickému zdroji. Druhým požadavkem je vysoká úroveň bezpečnosti, čehož dosáhneme složitým šifrovacím algoritmem na výpočetně omezeném zařízení. Nejvhodnějším kompromisem zmíněných požadavků a omezení je použití externího kryptografického prvku, hardwarového akcelerátoru, který je navržen pro výpočet právě takových algoritmů. V této práci se budu věnovat výběru kryptografické a autentizační metody, návrhu a implementaci vybraných metod do konečného systému složeného z omezených zařízení. Typickou aplikací tohoto zabezpečení jsou bezdrátové senzorové sítě – Wireless sensor network (WSN), které jsou napájeny většinou baterií a vyznačují se dlouhou životností na úkor nízké výpočetní schopnosti.

1 KRYPTOGRAFIE NA OMEZENÝCH ZAŘÍZENÍCH

V bezdrátových senzorových sítích existují mnohé nástrahy, které omezují bezpečný provoz systému. Bezdrátové senzorové sítě se skládají z hlavní stanice a mnoha nízkonákladových a bateriemi napájených zařízení. Hlavní stanice shromažďuje snímaná data z uzlů senzorů, které dále zpracovává, ale také senzorům vysílá pokyny. Zde je nutné dbát na ochranu přenášených dat, obzvláště v prostředí, kde je důležitá bezpečnost a soukromí shromážděných údajů.

Pro bezpečnou komunikaci ve WSN je zapotřebí efektivní kryptografický algoritmus vhodný právě pro takovéto prostředí. Ideální je volit efektivitu kryptografického algoritmu vzhledem k parametrům jakými jsou rychlost provádění operací, velikost v paměti a nízká spotřeba. Vzhledem k tomu, že každý kryptografický algoritmus má jiné výhody, je potřeba zvážit který algoritmus je vhodný pro vybraný systém.[2]. Nabízejí se nám dvě kryptografické metody, které si dále rozebereme a vybereme pro nás vhodné řešení.

1.1 Symetrická kryptografie

Symetrická kryptografie neboli také kryptografie sdíleného klíče, pracuje na principu šifrování a dešifrování zprávy stejným klíčem.

Zpráva se zašifruje tajným klíčem a odešle. Na výsledek se pak aplikuje dešifrovací algoritmus, pro který se použije týž tajný klíč. Dešifrováním se vyruší šifrování a tím získáme původní zprávu. Symetrická šifra má v jistém smyslu autorizační účinek. Z pohledu příjemce šifry mohl zprávu zašifrovat pouze známý odesílatel, který zprávu napsal, protože nikdo jiný než odesílatel a příjemce šifrovací klíč nezná.

Pro použití této metody je nezbytné, aby si komunikující strany nejprve předali tajný klíč, ale takovým způsobem, aby jej nemohl nikdo odposlechnout. Zde nastává určité riziko, jestliže nemáme k dispozici takový systém předání zprávy, který by zaručil maximální ochranu před rozšířením klíče nežádoucí straně.

Jiným rizikem symetrické kryptografie je možnost prolomení klíče „hrubou silou“. Kvůli zvyšujícímu se výkonu výpočetních systémů se často zvyšuje také hranice bezpečnosti mechanizmů zabezpečení a aktuální algoritmy je potřeba dále rozšiřovat. Zvýšení bezpečnosti se většinou provádí prodloužením délky klíče. To má ovšem také své hranice, proto je nezbytné vyvíjet další algoritmy, které používají složitější výpočty pro generování šifry. [1]

1.1.1 SHA-256

Nedílnou součástí v oblasti kryptografie jsou takzvané otisky zpráv (hash). Tyto otisky nám zaručují správnost konzistence původního řetězce při přebírání zprávy. Otisky mají vždy stejnou délku, ale mohou být vytvořeny z libovolně dlouhého vstupního řetězce. Principem je vytvořit takovou kombinaci hodnot, která by odpovídala pouze jednomu vstupnímu řetězci, ale takovým způsobem, aby se původní řetězec z výsledného otisku nedal získat zpět. Stejným způsobem lze taktéž porovnáním dvou takovýchto otisků ověřit totožnost zdroje bez nutnosti předání samotné zprávy.

Různá síla hashe zaručuje různou úroveň zabezpečení, kterou určuje neopakovatelnost výsledného otisku. Neboli delší otisk zprávy znamená menší pravděpodobnost stejného otisku různé zprávy. V našem případě budeme využívat hash dlouhý 256 bitů, který budeme využívat k získání symetrického klíče.

V kryptografii se hash využívá jako autentizační kód zprávy – Message Authentication Code (MAC), který se skládá z předem určené hodnoty, většinou se jedná o tajný klíč a náhodné číslo které se předává druhé straně. S těmito znalostmi se lze poté jednoduše domluvit na společném symetrickém komunikačním klíči.[1]

1.1.2 AES

Standard pokročilého šifrování – Advanced Encryption Standard (AES) je šifrovací metoda symetrické kryptografie, kterou vydal americký úřad pro standardizaci – (NIST). Je nástupcem předchozího standardu DES, jehož prolomení bylo podnětem pro uspořádání soutěže pro výběr nové šifrovací metody, kterou vyhrál algoritmus Rijndael od tvůrců V. Rijmena a J. Daemena.

Jedná se o blokovou šifru o délce 128 bitů s klíči o délce 128, 192 a 256 bitů. V současné době se jedná o velice rozšířenou metodu šifrování. Obzvláště v bezdrátových senzorových sítích má velké uplatnění. Velkou výhodou je skutečnost, že výsledná šifra je stejné délky jako šifrovaná zpráva, díky čemuž nedochází k navyšování přenášených dat a tudíž ke zvýšení spotřeby energie.[10]

1.2 Asymetrická kryptografie

Asymetrická kryptografie řeší nedostatek symetrické kryptografie, jakým je nutnost sdílení klíče. Jak jsme již zmínili, předání klíče druhé straně je ohroženo odposlechem a možností následného zneužití. Tento problém je u asymetrické šifry vyloučen použitím párů šifrovacích klíčů.

V následující kapitole uvedeme, že operace šifrování a dešifrování jsou v některých případech zaměnitelné, proto se u asymetrické kryptografie nemohou používat pojmy šifrovací a dešifrovací klíč, ale jsou zavedeny termíny klíč veřejný a soukromý. Základní vlastností této metody šifrování je skutečnost, že po relativně jednoduchém zašifrování dat veřejným klíčem, je stejným klíčem získání původních dat ze šifry velice obtížné. K dešifrování je zapotřebí využít klíč soukromý. [1]

1.2.1 RSA, DH, DSA

Mezi známé výrazy v oboru kryptografie veřejného klíče patří RSA a Diffie-Hellman (DH).

RSA je šifrovací algoritmus založený na složitosti operace faktorizace čísla n. To je dané vztahem n = pq, kde z čísla n se odvozuje veřejný šifrovací klíč, resp. z p, q klíč soukromý. Číslo n je společně s veřejným šifrovacím klíčem všeobecně známé. Hodnoty p, q společně s tajným klíčem jsou uchovány v tajnosti. V praxi se jedná o hodnoty čísel p a q velikosti přesahující 10^{153}.[5]

Diffie-Hellman je algoritmus používaný k výměně klíčů nezabezpečeným kanálem. Tento systém dovoluje dvěma komunikujícím zařízením bezpečné předání tajného klíče, neboli symetrického kryptografického klíče. Později tento klíč muže být použit pro šifrování tajných dat. [4]

Mezi standardy digitálních podpisů se řadí **algoritmus digitálního podpisu – Digital signature algorithm (DSA)**. Digitální podpis má podobný účel jako běžný podpis, s tím rozdílem, že zde zaručujeme kompletní autenticitu podepsané zprávy.

Využívá se asymetrické kryptografie v tom smyslu, že se hash zprávy zašifruje soukromým klíčem, ke kterému je vydána veřejná verze, pomocí níž lze takto šifrovaný hash dešifrovat a zkontrolovat porovnáním znovu vytvořeného hashe přijaté zprávy. Díky tomu, že víme čí je veřejný klíč, máme zajištěno, že původní zpráva je od jistého vydavatele tohoto veřejného klíče a od nikoho jiného.

V určitých případech se může ale stát, že není prokazatelný vydavatel tohoto podpisu, je proto nutné na podpis aplikovat certifikát, který se vydává s podpisem a je možné jej zkontrolovat u certifikační autority.[11]

1.2.2 Digitální certifikát

Kryptografie s veřejným klíčem umožňuje řešit problém distribuce klíčů veřejných i tajných. Jednou z možností jsou digitální certifikáty.

Jestliže chceme zavést šifrovací algoritmus veřejného klíče jako je RSA, účastník musí být schopen odeslat vlastní veřejný klíč jinému účastníkovi. Může se ale stát,

že nezúčastněná strana veřejný klíč zfalšuje a dostane přístup k tajným zprávám. Řešením je certifikace veřejného klíče. To znamená sloučení veřejného klíče s uživatelským identifikátorem vlastníka klíče a celým blokem podepsaným důvěryhodnou třetí stranou. Tato třetí strana může být certifikační orgán, ke kterému má komunita uživatelů důvěru.

Problémem distribuce tajných klíčů s veřejnými řeší algoritmus výměny klíčů Dffie-Hellmana. Algoritmus výměny klíčů může být použit pro sdílení tajného klíče mezi dvěma uživateli, nicméně to nezahrnuje autenticitu obou komunikujících stran.

Řešením je použití certifikátu veřejného klíče. Například jestliže chce strana A komunikovat se stranou B, strana A připraví zprávu, kterou zašifruje použitím symetrické kryptografie jednorázovým tajným klíčem. Dále je finální soukromý klíč zašifrován asymetrickou kryptografií veřejným klíčem strany B. Po zašifrování je tento klíč přiložen ke zprávě a společně jsou odeslány straně B. Jenom strana B je nyní schopna dešifrovat přiložený privátní klíč. Strana A má garanci platnosti veřejného klíče a tím i jistotu identity strany B, jestliže byl získán s platným certifikátem certifikační autority. [4]

1.2.3 Autenticita zpráv

Důležitou částí komunikace je ověřování zpráv. Je to velmi důležitá bezpečnostní služba, která nabízí pravost a nezměněnost zpráv a autenticitu jejich zdroje. Ověření může být provedeno šifrováním nebo také bez šifrování. V symetrické kryptografii, jestliže zpráva obsahuje kód detekce chyb a pořadové číslo, pak je příjemci garantována nezměněnost a správné pořadí zprávy. To je zajištěno tím, že pouze skutečný odesílatel je schopen zprávu zašifrovat takovým způsobem.

Ověřování zpráv bez šifrování zahrnuje přidání autentizačního štítku k odesílané zprávě. Tento štítek obsahuje takzvaný MAC nebo jednosměrnou hashovací funkci.

- V technice MAC jsou zahrnuty obě komunikující strany, ty sdílejí společný tajný klíč. Odesílatel vypočítá kód MAC pomocí tajného klíče, který přidá k odesílané zprávě. Příjemce provede stejný výpočet na přijaté zprávě. Pokud kód MAC obdržený ve zprávě odpovídá výpočtu příjemce, je zaručeno, že zpráva nebyla změněna. Existuje několik algoritmů pro výpočet kódu MAC.
- Hashovací funkce bere jako vstup zprávu proměnlivé velikosti a výstupem je souhrn o pevné délce, který může být přiložen k odesílané zprávě. Pro zaručení autenticitu zprávy, můžeme použít tři metody. Buďto šifrováním výsledného souhrnu symetrickou nebo asymetrickou kryptografií, případně lze provést výpočet ze zprávy a tajného čísla, které znají obě komunikující strany, které ale není ve zprávě obsaženo.

Kryptografie s veřejným klíčem může sloužit k ověření zprávy pomocí digitálního podpisu tak, že se výsledný souhrn zašifruje pomocí soukromého klíče. Příjemce zprávy může souhrn dešifrovat pomocí veřejného klíče odesílatele a potvrdit tak skutečnost, že zpráva byla odeslána známým odesílatelem.[4]

Jedním z omezení úrovně zabezpečení ve vestavěných systémech je potřeba dlouhých časů pro výpočet kryptografických operací. Kryptografie s veřejným klíčem reprezentuje nákladnější část zabezpečení systému, protože je všeobecně známo, že bez ohledu na délku klíče jsou asymetrické šifrovací algoritmy výpočetně mnohem náročnější než symetrické algoritmy.[1]

Jako příklad uvedeme dekódování klíče pomocí RSA algoritmu. S frekvencí 20 MHz 16. bitového vestavěného procesoru se RSA dešifrování tajného kódu o standartní délce 1024 bitového klíče potrvá 48 sekund. Takže s frekvencí procesoru 1000 MHz 32 bitové architektury bude stejná operace trvat méně než sekundu. Doba výpočtu delší jak jedna minuta může mít fatální následky u většiny aplikací běžících v reálném čase.[4]

1.3 Kryptografie eliptických křivek

V dnešní době se úroveň zabezpečení šifrovacími algoritmy zajišťuje jednou ze tří matematických operací. Jsou jimi výpočet diskrétního logaritmu, faktorizace celých čísel a diskrétní logaritmus na eliptických křivkách.

Jak již bylo řečeno, RSA kryptosystém je založen na obtížnosti faktorizace čísla. Kryptografie nad eliptickými křivkami – Eliptic curve cryptography (ECC) naproti tomu využívá obtížnosti diskrétního algoritmu na bodech eliptické křivky. V zásadě jde o problém, jak ke dvěma bodům G a Y náležejícím eliptické křivce Y = kG najít číslo k. Využívá ke svému účelu rovnici eliptické křivky (1.1), vyhovující množině bodů v rovině této křivky. V kryptografii využíváme pro výpočet diskrétní hodnoty.

$$y^2 = x^3 + ax + b \tag{1.1}$$

Obor kryptografie zabývající se eliptickými křivkami již dlouhou dobu přitahuje pozornost vědců zejména kvůli menším výpočetním nárokům a délce použitého klíče, které vyžaduje. Tyto vlastnosti jej činí velice lákavým pro nasazení právě v sítích, kde jsou možnosti jednotlivých zařízení omezené. Tato metoda v porovnání s předešlými dokáže pracovat s několikanásobně kratšími klíči na stejné úrovni bezpečnosti. Při srovnání s předchozím kryptosystémem RSA je doba nutná k prolomení několikanásobně nižší.

V předešlých letech byla považována za bezpečnou úroveň v systému RSA velikost klíče minimálně 1024 bitů a po roce 2010 je organizací RSA Security doporučována velikost minimálně 2048. V univerzálním měřítku vyjádření doby řešení

výpočetního problému nezávislém na specifické rychlosti zpracování instrukcí v procesoru má hodnotu 10^{20} miliónů instrukcí za sekundu. Systém ECC dále nabízí techniky pro ustanovení klíčů, šifrování a digitální podpisy.[5]

Pro své možnosti a vlastnosti se i přes bezpečnost a rozšíření RSA, kryptografie eliptických křivek pro použití veřejného klíče stále specifikuje. Důvodem je účinnost. Podle NIST je 224 bitový klíč používaný pro digitální podpisy ekvivalentní 2048 bitové délce klíče systému RSA. Například po roce 2030 NIST doporučuje klíč pro ECC o délce 256 bitů. ECC používá matematické funkce, které jsou mnohem složitější než u RSA, což vede ke složitější implementaci. To je také důvod, proč někteří stále dávají přednost systému RSA. Protokol digitálního podpisu s využitím eliptických křivek – The Elliptic Curve Digital Signature Algorithm (ECDSA) je samostatný algoritmus digitálního podpisu ve skupině kryptografických algoritmů NSA. [3]

1.3.1 ECDSA

ECDSA je rozšíření protokolu DSA o problematiku eliptických křivek. Funguje na podobném principu, kdy se vytvoří hash podepisované zprávy. Tento hash se následně podepíše a přiloží k odesílané zprávě. Obě komunikující strany musí být předem domluveny na parametrech, které budou při podpisu využívat, jako je typ použité křivky a hashovací funkce.[12]

1.3.2 ECDH

Protokol Diffieho-Hellmana s elyptickými křivkami – Elliptic curve Diffie–Hellman (ECDH) je rozšíření protokolu pro předání komunikačního klíče o eliptické křivky. Protokol je velmi náročný pro výpočet, proto se do zabezpečení bezdrátových senzorových sítí neaplikuje. Řešením by bylo aplikace protokolu v síti pomocí hardwarového akcelerátoru. [12]

2 AKCELEROVANÁ KRYPTOGRAFIE

Kryptografie je založena na složitosti určitých výpočtů. Efektivnost algoritmu závisí na rozdílu složitosti výpočtu vytvoření klíče nebo zašifrování zprávy a složitosti prolomení klíče případně šifry. V každém případě jde o výpočty náročné na energii a čas. Proto se vyhledává vhodné řešení nahrazující potřebný výkon mikrokontroléru ve formě externího hardwarového akcelerátoru. Výpočty v akcelerátorech probíhají na hardwarové úrovni, s minimální spotřebou a několikanásobně rychleji.

ECC třída algoritmu s veřejným klíčem může přinést významné zlepšení v době zpracování, což umožňuje použití real-time aplikací. Hlavním důvodem tohoto zvýšení rychlosti je velikost potřebného klíče, která je mnohem menší než které se používají u algoritmu RSA pro šifrování s podobnou výkonností.[4] Podle předchozí studie a výsledků energetické analýzy porovnávající výpočty různých délek klíčů pro různé situace algoritmů ECC a RSA, je vhodný i přes svoji výpočetní složitost algoritmus eliptických křivek kryptografie s veřejným klíčem.

Autentizačního procesu se dá dosáhnout softwarovou implementací operací na straně hostitele nebo použitím hardwarového akcelerátoru. Zabezpečovací proces není jednoduchý a je obtížné systém o tuto činnost softwarově doplnit, mnohem efektivnější řešení je proto s použitím externího akcelerátoru, který autentizaci zajistí. Především kvůli výpočetně a energeticky omezeným možnostem zařízení je nutné zabezpečení systému řešit externím prvkem, který tento systém doplňuje o výpočty potřebné k autentizaci zařízení použitých v systému.[5]

Pro tuto práci máme k dispozici akcelerátor ATECC108 od firmy Atmel, který bude využit pro autentizaci a distribuci klíčů v síti.

2.1 Seznámení s akcelerátorem ATECC108

Akcelerátor nabízí hned několik způsobů použití jako je například antiklonovací zařízení, zavedení ověřování softwaru proti pirátům, kontrola přístupu v síti a k počítači nebo autentizovanou případně šifrovanou síťovou komunikaci.

Využívá hardwarově akcelerovaný algoritmus pro výpočet veřejného klíče PKI, ECDSA a NIST. Podporuje standardy výpočtu na eliptických křivkách P256, B283 a K283. Podporuje SHA-256 hashovací algoritmus a možnost šifrovaného hashovacího algoritmu HMAC. Obsahuje paměť pro 16 klíčů, unikátní 72 bitové sériové číslo, vnitřní FIPS generátor náhodných čísel (RNG), 8,5 Kb EEPROM pro klíče, certifikáty a data, 512 OTP bitů pro fixovanou informaci nebo protokolování spotřeby, 1 MHz I2C sběrnici. Je napájen 2 – 5,5 V, přičemž spotřeba ve spánku je menší než 150 nA.

Akcelerátor dále obsahuje statickou paměť RAM (SRAM), používanou k ukládání vstupních příkazů, výsledků zadaných funkcí, případně pro uložení jejich mezivýpočtů. Veškerý obsah této paměti je vymazán, kdykoliv akcelerátor přejde do režimu spánku, nebo když je vypnuto napájení. Tato dočasná paměť je označena jménem TempKey. Jsou zde uloženy data jako vstup pro funkce MAC, HMAC, CheckMac, GenDig a DeriveKey. Paměť je dále používaná pro uložení klíče při šifrování nebo dešifrování dat, při čtení nebo zápisu dat. Data uložená v této paměti nemohou být nikdy přečtena ze zařízení, ale slouží pouze pro vnitřní použití. [6]

2.2 Aplikace akcelerátoru

Akcelerátor využijeme pro autentizaci zařízení pomocí jeho sériového čísla, díky tomu odpadá nutnost zadávat šifrovací klíč ručně. Zvolený akcelerátor plní také podmínku energetické náročnosti, díky jeho rychlosti a možnosti se vypnout po dokončení autentizace. K akcelerátoru je připravena knihovna, která obsahuje všechny dostupné funkce akcelerátoru:

- CheckMac funkce vypočítá hodnotu MAC vytvořenou jiným kryptografickým zařízením a porovná ji se vstupní hodnotou. Příkaz vrací logickou hodnotu označující rovnost nebo nerovnost obou hodnot. Před spuštěním tohoto příkazu mohou být případně spuštěny příkazy Nonce anebo GenDig, aby vytvořili klíč nebo kódové slovo uložené v TempKey, ze kterých se vstupní data pro výpočet hodnoty MAC skládají. Čas potřebný pro vykonání tohoto příkazu výrobce udává 12 až 38 ms.
- DeriveKey je funkce která kombinuje aktuální hodnotu klíče s hodnotou kódového slova (nonce) uloženou v TempKey za použití SHA-256 algoritmu a výsledek vloží do klíčového slotu. Před provedením tohoto příkazu musí být do TempKey vložena hodnota příkazem Nonce. V závislosti na nastavení módu příkazu DeriveKey musí být toto kódové slovo vytvořeno vnitřním generátorem náhodných čísel, nebo musí být předem určeno. Čas potřebný pro vykonání tohoto příkazu je 14 až 62 ms.
- DevRev příkaz vrací hodnotu o délce čtyř bajtů představující číslo revize zařízení. Navržený software by neměl využívat toto číslo k výpočtům, protože se může časem změnit. Potřebný čas pro přečtení hodnoty je 0,4 až 2 ms
- GenDig generuje kód pro ochranu dat z náhodného nebo vstupního slova a klíče. Používá algoritmus SHA-256 na kombinaci uložené hodnoty s obsahem TempKey, které musí být nastaveny před spuštěním tohoto příkazu. Uložené hodnoty mohou pocházet z jednoho z datových slotů, a to z části OTP paměti, případně ze slotu z konfigurační zóny zařízení. Výsledný kód je uložen v Temp-

Key a je použit jako součást zprávy používané příkazy MAC, CheckMac nebo HMAC, případně pro příkaz čtení nebo zápis může použít hodnotu pro autentizaci nebo utajení dat. Příkaz vyžaduje pro své provedení čas 11 až 43 ms

- GenKey slouží k vytvoření soukromého nebo veřejného klíče. K jejich vytvoření používá algoritmus eliptických křivek. Vytváři klíč o délce 256 nebo 283 bitů. Čas potřebný pro výpočet klíče je 40 až 96 ms.
- HMAC příkaz počítá podle algoritmu HMAC/SHA-256 hodnotu z klíče uloženého v paměti akcelerátoru, vstupních dat a z různých informací o akcelerátoru uložených v paměti. Obsažením sériového čísla akcelerátoru v použité zprávě pro výpočet algoritmu je zajištěna diverzifikace výsledku. Délka provádění příkazu je 27 až 69 ms
- funkce Lock umožňuje preventivní uzamčení proti modifikaci vybrané zóny akcelerátoru. Případný útočník nebude mít tedy možnost, bez znalosti klíče pro odemčení, nijak narušit bezpečnost uložených dat a ohrozit tím celý systém. Čas potřebný pro uzamčení zóny je 5 až 24 ms.
- MAC funkce vypočítá podle algoritmu SHA-256 hash ze vstupních dat kombinovaných s daty uloženými v paměti akcelerátoru. Této funkci musí předcházet příkaz Nonce, který vytvoří bitové slovo z generátoru náhodných čísel nebo ze vstupních dat, případně jejich kombinaci a výsledek uloží do TempKey. Podle potřeby lze do výsledného řetězce zakomponovat sériové číslo akcelerátoru. Výpočet funkce MAC zabere 12 až 35 ms.
- Nonce slouží k vytvoření kódového slova. Ke své funkci využívá generátor náhodných čísel, který zajistí jedinečnost každého vytvořeného kódu. Výsledek zapíše do TempKey. Umožňuje také vložení uživatelem zadaného slova do TempKey a to v původní podobě nebo jeho kombinací s hodnotou vytvořenou generátorem náhodných čísel. [9]

Tyto informace jsou převzaty z dokumentace předešlé verze akcelerátoru, který ještě nedisponoval výpočty asymetrické kryptografie. Popisy funkcí novějšího akcelerátoru jsou odvozeny z dostupné knihovny vytvořené výrobcem pro tento akcelerátor. Podrobné informace je možné nalézt v originální dokumentaci aktuálního akcelerátoru, ten je ovšem pod ochrannou licencí podmíněnou podpisem smlouvy o mlčenlivosti, tu schvaluje jeho výrobce - společnost Atmel.

2.3 Komunikace

Pro účel této práce je akcelerátor napájen 3,3 V. Je připojen pomocí sběrnice I2C k mikrokontroléru, který obsahuje program pro obsluhu akcelerátoru. Příklad komunikace naznačuje tabulka 2.1

Průběh komunikace probíhá následovně:

- Začátek komuniace je stejný pro všechna zařízení komunikující po sběrnici I2C, tedy odesláním start condition, neboli podmínku startu, kdy se postupně nastaví z logické 1 na 0 SDA a poté SCL.
- Následuje krok probuzení. Tento krok je individuální pro každé zařízení. V případě akcelerátoru je nutné pro další komunikaci nastavit SDA na dobu alespoň 60 μs na úroveň logické 0.
- Po probuzení následuje ukončení odesílání bajtu probuzení, podobně jako jeho začátek tentokrát podmínkou stop, kdy se nejprve nastaví SDA na log. 1 a poté SCL.
- Pro odeslání dalšího bajtu se po určité době opět odešle start condition.
- Následuje adresa slave zařízení vyjádřena sedmi bity. Další bit nese informaci o tom, zda se bude na zařízení zapisovat nebo z něj číst.
- Po přijetí devátého potvrzovacího bitu se přijímou resp. odešlou data.
- Ukončení bajtu potvrdí stop condition. Tímto způsobem probíhá komunikace na sběrnici I2C dokud se požadavek nezpracuje. Aby se zachovala co nejmenší energetická zátěž, je potřeba zařízení přepnout do režimu spánku.

2.3.1 Šifrovaná komunikace

Pro lepší zabezpečení systému čip disponuje možností zabezpečené komunikace, kdy se text po sběrnici posílá šifrovaně. To probíhá XORováním posílané zprávy hashem vytvořeným z kombinace předem vybraného klíče, který je uložen v paměti akcelerátoru a případně náhodného čísla, které se získá z akcelerátoru před samotným odesláním zprávy. Takto vytvořená šifra se odesílá společně s hashem její nešifrované podoby a dalšími předem určenými informacemi podle dokumentace.

Kontrola samotné správnosti šifrované zprávy se provádí v akcelerátoru porovnáním vlastního vytvořeného a přijatého hashe dešifrované zprávy. Zpráva se automaticky v akcelerátoru dešifruje a v paměti zůstává uložena v nešifrované podobě. Pro případné čtení z paměti akcelerátoru lze opět aplikovat šifrování v podobě XORování zprávy klíčem uloženým v jednom ze slotů v akcelerátoru. Takto získaná zpráve se dešifruje v systému stejným klíčem.

Klíče pro šifrovaný zápis a šifrované čtení mohou být různé podle toho, jak se nastaví paměťový slot akcelerátoru. Takto nastavený akcelerátor musí být pro použití jeho funkcí uzamčen a nelze tak poté měnit jeho konfiguraci. Šifrovaně lze pouze odesílat a přijímat informace uložené v paměti. Příkazy, potvrzení a všechna data, která v paměti akcelerátoru uložena nejsou, se posílají nešifrovaně, stejně tak i odpovědi s vygenerovanými náhodnými čísly. Náhodná čísla lze získat zabezpečeně pouze jejich kombinací s vybraným slotem v paměti akcelerátoru, kterou lze následně

Tab. 2.1: Příklad komunikace MCU s ATECC108 [9]

MCU	**Směr**	**ECC108**
Start	$\longrightarrow$	
Probudit	$\longrightarrow$	
Stop	$\longrightarrow$	
Start	$\longrightarrow$	
Adresa/čti	$\longrightarrow$	
	$\longleftarrow$	Data
Stop	$\longrightarrow$	
Start	$\longrightarrow$	
Adresa/zapiš	$\longrightarrow$	
Příkaz	$\longrightarrow$	
Data	$\longrightarrow$	
Stop	$\longrightarrow$	
Start	$\longrightarrow$	
Adresa/čti	$\longrightarrow$	
	$\longleftarrow$	Data
Stop	$\longrightarrow$	
Start	$\longrightarrow$	
Adresa/zapiš	$\longrightarrow$	
Uspat	$\longrightarrow$	
Stop	$\longrightarrow$	

bezpečně přečíst.

V případě odcizení akcelerátoru by se útočník mohl pokoušet zapsat vlastní hodnotu do paměti a opět ji přečíst. Získal by tím zašifrovanou podobu své zprávy a mohlo by dojít k odvození šifrovacího klíče. Proto akcelerátor disponuje dodatečnou autorizací.

2.3.2 Autorizace

Autorizace je funkce akcelerátoru, kterou lze přiřadit každému ze slotů v paměti akcelerátoru. Po přiřazení této funkce klíči v daném slotu je potřeba provést před každým použitím klíče autorizaci. Ta se provádí pomocí dalšího klíče, který je uložen v paměťovém slotu akcelerátoru. Před autorizací může být vyžadováno vytvoření náhodného čísla, ze kterého se bude autorizace skládat. Takto vytvořené a systému

odeslané náhodné číslo je potřeba zkombinovat s hodnotou autorizačního klíče a dalších hodnot podle dokumentace a následně vytvořit hash. Takto vytvořený hash se odešle spolu s žádostí o použití vybraného klíče, tu akcelerátor zpracuje a vyhodnotí. Po kladném vyhodnocení žádosti lze vybraný klíč použít například pro šifrovaný zápis, případně čtení. Útočník tak nemá možnost zjisti šifrovací komunikační klíč bez předchozí autorizace. Autorizaci lze použít i pro další funkce akcelerátoru, aby nemohlo dojít k případné změně hodnoty v paměťovém slotu akcelerátoru a tím ke znehodnocení celého uzlu.

2.4 Zajištění bezpečnosti

Pro maximální ochranu dat v akcelerátoru výrobce stanovil několik pravidel, se kterými je nutno počítat již při návrhu bezpečnostního protokolu v němž bude čip osazen. Datová paměť akcelerátoru je rozdělena na 16 slotů, které jsou samostatně konfigurovatelné a uzamykatelné proti přečtení nebo přepisu.

Dále je zde paměť SRAM, označovaná výrobcem jako TempKey. Tato dočasná paměť slouží k přípravě dat pro příjem nebo odeslání. Tato paměť je chráněna proti přečtení, slouží pouze jako mezipaměť mezi příkazy pro akcelerátor.

V akcelerátoru je další část paměti označena jako konfigurační zóna, ve které se nastavují mimo jiné parametry pro jednotlivé datové sloty. Tato paměť musí být před použitím slotů uzamčena, aby nemohlo dojít k případné změně konfigurace a vyčtení dat ze slotů. Uzamčení konfigurační zóny je nevratná operace, je proto velmi důležité provést návrh nastavení konfigurační zóny velice pečlivě a s uvážením, které sloty by mohly být použity do budoucna.

Zvláštní zabezpečení výrobce přiřadil slotům nastaveným pro asymetrické klíče, které se řídí specifickými pravidly a pro které má konfigurační zóna jiný význam. Privátní klíče asymetrické kryptografie není možné z čipu jakkoliv přečíst, neexistuje způsob jak se dozvědět jejich hodnotu. V některých případech je však tato vlastnost nežádoucí (např. v případě potřeby distribude klíče). Soukromé klíče lze pouze generovat, případně přímo zapisovat. Veřejné klíče je poté možno vytvářet z těchto soukromých a použít je pro podpis dat, případně certifikát. [16]

3 AUTENTIZAČNÍ MODELY

Hardwarový akcelerátor použitý v této práci nám nabízí níže popsané metody autentizace. Jsou rozděleny podle potřeb použitých zařízení. Protože v této práci k autentizaci používáme identifikační číslo akcelerátoru, bylo nutné vhodně zvolit takový model, kdy není potřeba zadávat šifrovací klíč. Dále je potřeba vybírat autentizační model podle úrovně bezpečnosti vybrané metody tak, aby byla nejlépe chráněna před případnými útoky.

3.1 Autentizace s pevnou výzvou

Autentizace s pevnou výzvou je model, který nevyžaduje klíč nebo výpočet současně na obou zařízeních. S tímto modelem hostitel odesílá stejnou výzvu pokaždé, kdy je potřebná autentizace a klient vždy reaguje stejnou odpovědí. Zajištěním použití stejné výzvy a odpovědi mohou mít obě strany předem vypočítané verze párů výzev a odpovědí.

3.2 Autentizace s náhodnou výzvou

Metoda náhodné výzvy vylepšuje předešlou verzi přidáním náhodně zvolené výzvy na každý požadavek. Tato funkce vylepšila systému ochranu před takzvanými replay-style útoky. Ty fungují na principu monitorování sběrnice, zapsání páru výzvy a odpovědi a poté použití této nahrávky k oklamání systému při validaci falešného zařízení.

3.3 Autentizace s unikátní výzvou

Tento model je rovněž zdokonalením metody s pevnou výzvou přidáním unikátní výzvy na každý požadavek. Tato autentizační funkce chrání systém před útoky replay-style. Je ale zapotřebí dalšího hardwarového prvku pro generování náhodných čísel, aby byla zajištěna maximální unikátnost výzvy.

3.4 Autentizace s různorodým klíčem

Tato metoda zahrnuje sériové číslo každého autentizačního akcelerátoru jako součást autentizačního výpočtu. Jedinečnost každého klíče, odvozeného sériovým číslem zařízení, umožňuje hostiteli identifikaci specifického příslušenství, pokoušející se o autentizaci. Tento postup umožňuje použití seznamu přístupů aktuálního systému.[8]

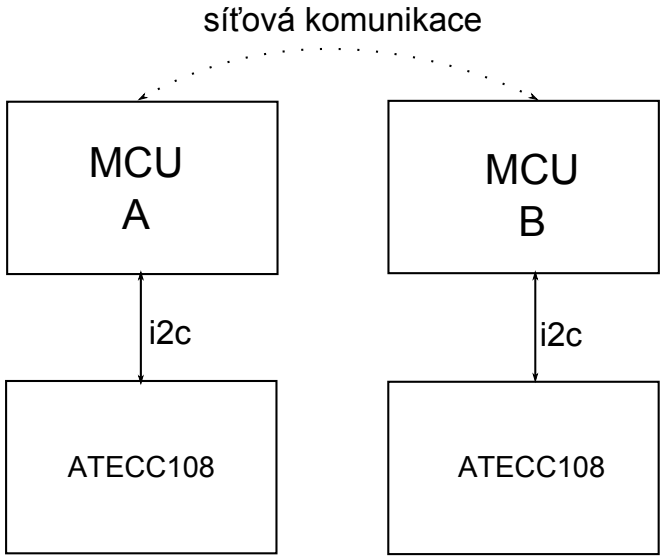
Pro účel bakalářské práce je využita autentizace s různorodým klíčem, kdy se k hlavnímu ROOT klíči přidá distribuované náhodné číslo, z jejichž kombinace se vytvoří hash, který bude sloužit jako komunikační klíč mezi dvěma uzly pro předání hlavního symetrického komunikačního klíče sítě.

4 APLIKACE VLASTNÍHO MODELU

Při návrhu vlastního modelu bylo nutné se podřídit dostupným funkcím vybraného akcelerátoru. Asymetrickou kryptografii bylo možné aplikovat pouze na vytvoření digitálního podpisu. Při vytváření dalších bezpečnostních protokolů, se vzhledem k omezením akcelerátoru, asymetrická kryptografie dále využívat nedala.

Navržený model je možné pro lepší bezpečnost aplikovat s akcelerátorem, kde se využije především jeho vlastnost nedobytné paměti pro klíče, dále pak jeho generátor náhodných čísel, který je vyvinut podle přísných kritérií společnosti NIST a případně jeho výpočetních schopností v případě výpočtu hashe, který se v čipu používá téměř při každé funkci.

Navržený model se bude implementovat do uzlu v síti, který se bude skládat mimo jiné z mikrokontroléru a akcelerátoru. Mikrokontroler bude obsahovat hlavní obslužný program a akcelerátor bude provádět potřebné kryptovací výpočty a bude sloužit jako paměť pro tajné klíče potřebné pro provoz sítě.4.1



Obr. 4.1: Schéma komunikace dvou uzlů s akcelerátory

4.1 Autentizace pomocí ECDSA

Pro aplikaci technologie eliptických křivek máme k dispozici pouze protokol ECDSA, který využijeme pro zvýšení úrovně zabezpečení v případě autentizace zpráv. Každá zpráva posílaná v síti bude elektronicky podepsána a tento podpis bude v přijímacím zařízení ověřen.

Pro tento model bude v každém akcelerátoru v síti uložen stejný soukromý klíč asymetrické kryptografie, ze kterého se budou vytvářet stejné veřejné klíče v každém uzlu na základě náhodného čísla tvořící základ výpočtu. Takto vytvořený klíč se aplikuje na elektronický podpis, který se přikládá ke zprávě. Díky tomuto soukromému klíči je zajištěna autenticita zdroje, který zprávy vytváří. Jakákoliv změna posílané zprávy by znamenala podvrh a zpráva by nebyla akceptována.

ECDSA taktéž zaručuje pravost vytvořeného klíče především díky technologii eliptických křivek, takže se nemůže stát, že by si útočník vytvořil vlastní podpis. Díky kombinaci s náhodným číslem dostáváme záruku neopakovatelnosti podpisu, jinak by mohlo dojít k použití dvou stejných podpisů na dvě různé zprávy a mohlo by tak dojít k vyzrazení klíče a zneužití útočníkem pro vytváření vlastních podpisů.

4.2 Distribuce klíče

Pro komunikaci mezi všemi uzly sítě je nutné ustanovení pravidel pro předání a akceptaci klíče pro každý samostatný uzel. Toto řízení předání klíče bude pro každý uzel stejné. Aby systém fungoval, musí se klíčům přiřadit určité parametry, podle kterých se bude rozhodovat zda klíč převzít, případně oznámit odesílateli, že je klíč neplatný.

Společný komunikační AES klíč sítě má určitou životnost. Ta je určena počtem bitů počítadla v šifrovacím systému. V tomto případě je pro počítadlo stanoveno 16 bitů, což je celkem 65 536 možností použití jednoho klíče. Toto číslo bude sloužit jako jeden parametr, který se bude během procesu distribuce klíče porovnávat.

Dalším parametrem je pořadové číslo aktuálního AES klíče. Pro tento parametr jsme vymezili velikost 32 bitů, což odpovídá celkem 4 294 967 296 klíčům. Tento počet je tak vysoký z toho důvodu, aby nedošlo při probuzení uzlu po dlouhém spánku k riziku opakovaného použití AES klíče, způsobené přetečením tohoto čísla. Větší číslo snižuje možnost takovéhoto přetečení.

Posledním parametrem klíče je takzvaná popularita. Toto číslo má délku 8 bitů, což je celkem 256 možností inkrementace popularity. Popularita klíče udává počet uzlů které již tento klíč vlastní. Počítáme tedy, že v síti bude maximálně 256 uzlů. Pro navýšení počtu uzlů je potřeba zvětšit toto číslo o další bity. Při každém potvrzeném předání klíče se toto číslo inkrementuje a odešle dalšímu uzlu již s novou hodnotou.

Samotný postup je takový, že se při rozhodování o přijetí klíče systém řídí podle priorit jednotlivých parametrů. Hlavním parametrem v tomto ohledu je pořadové číslo klíče, tedy číslo dlouhé 32 bitů. Nemělo by se nikdy stát, že by nově vytvořený klíč měl menší hodnotu než již jednou použitý. Tento hlavní parametr, rozhoduje

zda klíč přijímout nebo ne.

Může se stát, že se v síti rozhodnou dva uzly obnovit klíč najednou. V tomto případě by bylo pořadové číslo obou nových klíčů stejné, proto je zaveden další parametr, který udává počet uzlů, jenž tento nový klíč již vlastní, tzv. popularita. Klíč s vyšší popularitou má vždy přednost a bude nahrazen klíčem s nižší popularitou. Jak bylo řečeno, při každém předání se popularita inkrementuje. Při přepisování klíče s menší popularitou klíčem s větší popularitou se klíč přepisuje společně s jeho ostatními parametry, to znamená že nedochází k navyšování pořadového čísla klíče.

Poslední parametr, počet použití klíče, slouží pouze při probouzení zařízení ze spánku nebo při přidání nového uzlu, kdy se v síti již nějaký čas komunikuje. Takový uzel si podá žádost o klíč a ten přebere i s tímto parametrem.

4.2.1 Schéma protokolu distribuce klíče

Abychom alespoň částečně omezili autentizaci mezi uzly, což by znamenalo zvýšení spotřeby, je v každém uzlu rezervována paměť pro 3 komunikační AES klíče s jejich parametry. V uvedených diagramech jsou tyto klíče označeny jako aesKey0, aesKey1 a aesKey2.

AesKey1 je pozice pro aktuální komunikační klíč, který se právě v síti používá pro symetrické šifrování posílaných dat. Jestliže vyprší platnost klíče, tedy dojde k dosažení limitu použitelnosti pro aktuální klíč, přepíše se aktuální klíč v pozici aesKey1 na pozici aesKey0, a nově vytvořený klíč se uloží do pozice aesKey1. Původním aktuálním klíčem, nyní tedy klíčem v pozici aesKey0 se tento nový aktuální klíč bude šifrovat a rozesílat dále do sítě. Takto odeslaný klíč se svými parametry se odešle vedlejšímu uzlu, který jej uloží do své paměti na pozici aesKey2. Tato paměť slouží vždy pouze pro ukládání nových klíčů, které byly přijaty ze sítě. Je to proto, že klíč v pozici aesKey0 můžeme stále ještě potřebovat, protože nevíme zda je přijatý klíč akceptovatelný. Přijatý klíč tedy zkusíme dešifrovat aktuálním klíčem v pozici aesKey1.

Jestliže dešifrování proběhlo úspěšně, teprve nyní se klíč z pozice aesKey1 přemístí do pozice aesKey0, a klíč z pozice aesKey2 se přemístí do pozice aesKey1. Úspěšné dešifrování klíčem v pozici aesKey1 zaručuje skutečnost, že přijatý klíč má vždy pořadové číslo o jedna větší než původní aktuální klíč. Proto v tomto případě není nutné porovnávání těchto ani dalších hodnot. Takový klíč je vždy přijat a stejnou metodou s navýšenou popularitou odeslán dalšímu uzlu.

Jestliže ale dešifrování neproběhlo úspěšně, uzel se pokusí dešifrovat přijatý klíč klíčem v pozici aesKey0. Jestliže je dešifrování úspěšné, znamená to, že přijatý klíč má pořadové číslo stejné jako vlastní aktuální klíč v pozici aesKey1. V tomto případě je na řadě porovnání popularit obou klíčů.

Jestliže popularita přijatého klíče, tedy klíče v pozici aesKey2, je větší než popularita aktuálního klíče, znamená to, že přijatý klíč je v síti rozšířen více než aktuální a je potřeba jej přijmout a rozeslat dále do sítě. V tom případě se tedy pouze přemístí nově přijatý klíč z pozice aesKey2 do pozice aesKey1. Distribuce tohoto klíče opět probíhá šifrováním klíčem v pozici aesKey0 s navýšenou popularitou. Jestliže je ale popularita přijatého klíče v pozici aesKey2 menší než popularita aktuálního klíče uzlu v pozici aesKey1, proběhne opět šifrování aktuálního klíče v pozici aesKey1 klíčem v pozici aesKey0 s navýšenou popularitou a odešle se zpět uzlu, od kterého jsme přijali nový klíč, který ale nebyl akceptován.

Všechny další případy, které končí neúspěšným dešifrováním klíči v obou pozicích, znamenají, že přijatý klíč je od uzlu, který se právě probudil a promeškal aktualizaci komunikačního AES klíče, případně je to právě tento uzel, který se probudil a přijal zprávu o aktualizaci AES klíče. V tomto případě se provede obnova a distribuce klíče s autentizací.

Nyní je potřeba kontrolovat aktuálnost klíče pomocí pořadového čísla. Takto distribuovaný klíč s autentizací je přijat a uložen do pozice aesKey2 a porovná se pořadové číslo. Jestliže tento přijatý klíč má menší pořadové číslo, jedná se o starý klíč. Aktuální klíč v pozici aesKey1 se tedy zašifruje tímto přijatým klíčem v pozici aesKey2 a odešle se zpět uzlu, který nám tento starý klíč odeslal. Ten jej automaticky dle předchozích pravidel příjme a starý klíč se zahodí. Jestliže ale tento přijatý klíč má větší pořadové číslo, klíč se akceptuje a uloží se do aktuální pozice aesKey1.

Podobně se postupuje při dešifrování přijatých dat, tedy podle toho jakým klíčem lze zprávu dešifrovat se určí, zda tomuto odesílajícímu uzlu poslat aktuální klíč. Jestliže se tedy zprávu podaří dešifrovat starším klíčem v pozici aesKey0, zašifruje se aktuální klíč v pozici aesKey1 klíčem v pozici aesKey0 a odešle se zpět tomuto uzlu. Jestliže ale zpráva nebyla správě dešifrována žádným z klíčů, proběhne opět autentozovaná distribuce aktuálního klíče.

4.3 Schéma protokolu autentizace

Autentizační protokol pomocí ECDSA je vyobrazen v diagramech 4.2, 4.3 a 4.4. Autentizace je důležitá především pro zamezení cizím subjektům zasahovat do běhu a bezpečnosti senzorové sítě. Je to důležitý krok, na kterém závisí ochrana dat.

Jestliže by byla autentizace na nízké úrovni, mohlo by dojít k odcizení a zneužití symetrického komunikačního klíče. Případný útočník by mohl přečíst posílaná data nebo odesílat do sítě vlastní falšovaná data.

V tomto případě využijeme digitální podpis pro ověření žadatele o klíč. Ověření podpisu zajistí soukromý klíč, který je ve všech uzlech v síti stejný. Podmínkou je,

aby tento klíč nemohl nikdo zjistit, je tedy nutné tento klíč zapsat jednorázově do paměti každého uzlu již při jeho výrobě. Klíč se v uzlech po celou dobu životnosti sítě nemění, je proto vhodné využít technologii eliptických křivek, zejména proto, že je tato technologie na nejvyšší úrovni zabezpečení a s délkou klíče 256 bitů by prolomení trvalo déle než životnost celé sítě.

Hlavní předání klíče proběhne pomocí dvou klíčů. Pro vyšší stupeň zabezpečení, tedy znemožněním vyčtení klíče z paměti, je vhodný akcelerátor, který tuto vlastnost nabízí. Je ale potřeba aplikace dvou klíčů, kdy jedním klíčem, určeným pro komunikaci mezi mikrokontrolérem a akcelerátorem, se zpráva zašifruje. V akcelerátoru se zpráva podepíše pomocí ECDSA algoritmu a z akcelerátoru se přečte zašifrovaná klíčem, který je uložen pouze v akcelerátorech. Klíč v akcelerátorech je ve všech stejný a neměnný. Neměnný proto, že nedokážeme zaručit absolutní synchronizaci jednotlivých uzlů. Jinak by teoreticky mohlo dojít ke změně klíče ve chvíly, kdy by byl jeden z uzelů neaktivní, nevěděl by o této změně a aktuální klíč by nemohl obdržet.

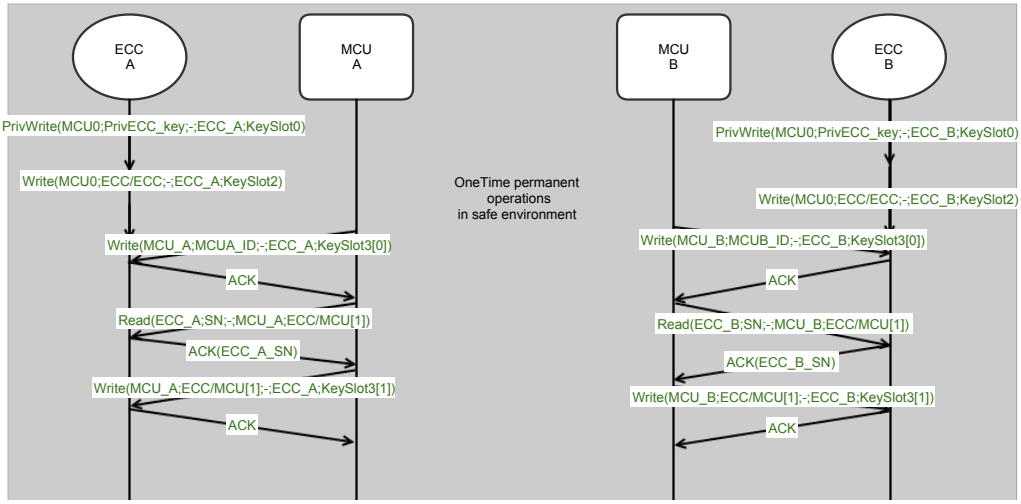
Ovšem díky tomuto neměnnému klíči, který ani mikrokontroléry neznají, se takto zašifrovaná zpráva na jednom uzlu z akcelerátoru přečte a v druhém uzlu se úplně stejná zpráva do akcelerátoru zapíše a v tomto uzlu se z akcelerátoru opět přečte klíčem pro komunikaci mezi akcelerátorem a mikrokontrolérem. Tyto komunikační klíče pro akcelerátor a mikrokontrolér jsou v každém uzlu jiné, jsou složené z kombinace sériového čísla akcelerátoru a unikátního čísla mikrokontroléru. Tím se zabrání případnému zneužití akcelerátoru po jeho odletování z uzlu. Je tak zajištěno, že právě jeden vybraný mikrokontrolér může s právě jedním vybraným akcelerátorem komunikovat.

4.3.1 Sekvenční diagram

Sekvenční diagram vyobrazuje postup protokolu od jeho počátku do ukončení s konkrétním účelem. Zobrazuje postup sekvence zpráv a akcí v jednotlivých uzlech v síti.

Před uvedením sítě do provozu je nutné jednotlivé uzly naplnit tajnými daty v prostředí, kde zaručeně nemůže dojít k jejich odposlechu. Načtení dat musí proběhnout v zabezpečeném prostředí. Toto načtení proběhne pouze jednou za celou dobu provozu jednotlivých uzlů. V našem případě se nastaví tajné klíče pro komunikaci jednotlivých uzlů v síti, tedy soukromý klíč asymetrické kryptografie pro ověření uzlu, tajný distribuční klíč pro autentizované předání klíče a komunikační klíč pro mikrokontrolér a akcelerátor. Tento klíč, jak je uvedeno výše je složen z kombinace identifikačních údajů mikrokontroléru a akcelerátoru. Tento postup je vyobrazen na sekvenčním diagramu na obr. 4.2.

Nejprve dojde k zapsání hodnoty soukromého klíče asymetrické kryptografie do

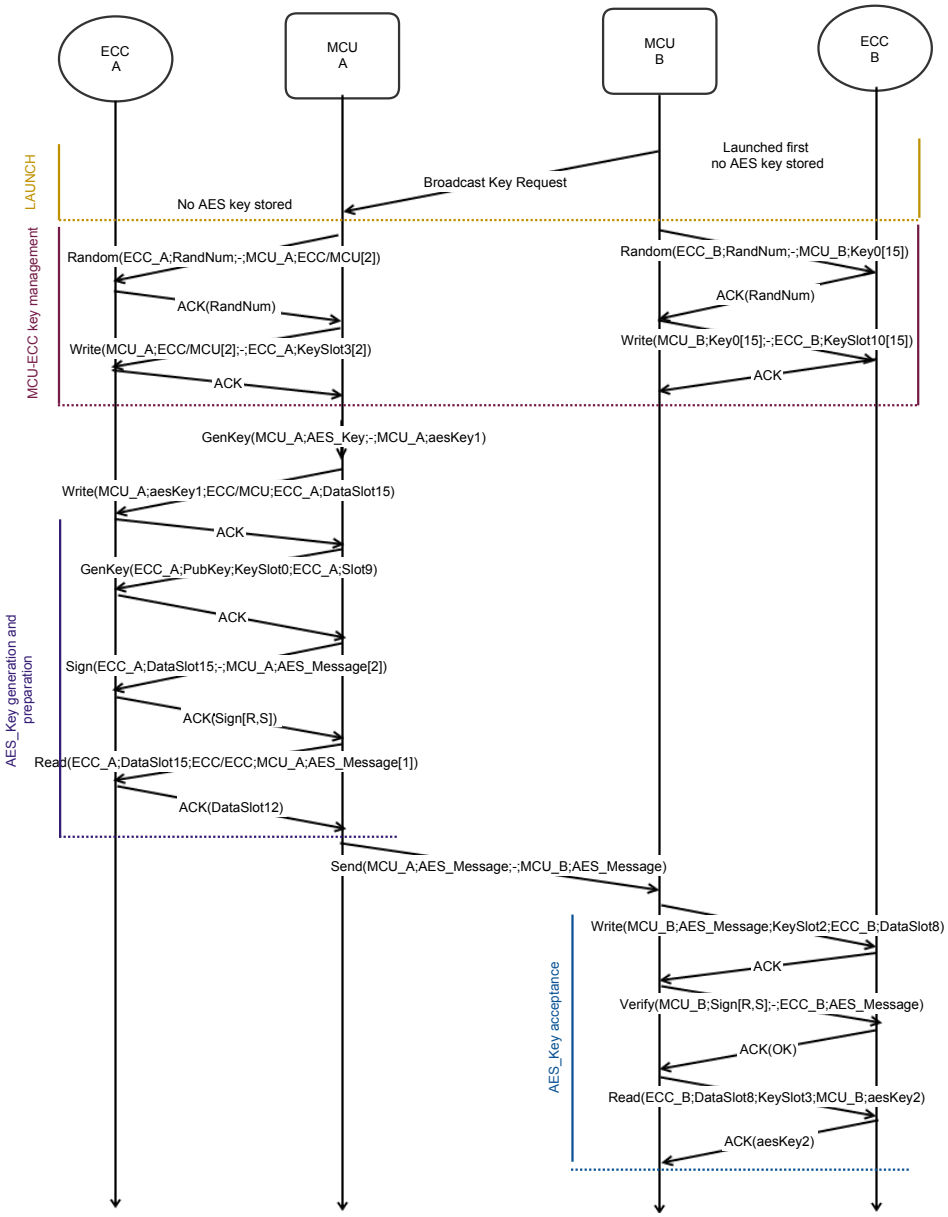


Obr. 4.2: Program pro bezpečné prostředí ECC modelu

akcelerátoru pomocí externího systému, kterým se takto provede zápis do všech akcelerátorů v síti. V diagramu je označen zápisem PrivECC_key. Stejně tak proběhne další krok a to zapsání tajného distribučního autentizačního klíče, označen jako sharedSHA. Další komunikace probíhá již mezi dvojicí akcelerátoru a mikrokontroléru která bude nasazena v síti. Proběhne zápis unikátního čísla mikrokontroléru do části paměti akcelerátoru. Toto je v diagramu označeno jako MCU_MAC_X. V dalším kroku se získá sériové číslo z akcelerátoru, označeno jako ECC_X_SN. Toto číslo se uloží do flash paměti mikrokontroléru a zpátky se zapíše do paměťového slotu pro klíče v akcelerátoru k hodnotě unikátního čísla mikrokontroléru. Takto se předchystá komunikační klíč akcelerátoru a mikrokontroléru. Ke stejnému uspořádání dojde i v paměti mikrokontroléru. Po dokončení zápisu do paměti akcelerátoru se tato datová paměť uzamkne a znemožní se tak nechtěnému zásahu do paměti akcelerátoru. V tuto chvíli je uzel připraven pro nasazení v terénu.

Další postup je vyobrazen diagramem na obr. 4.3. Zde je vykreslena situace ihned po nasazení uzlu do sítě. Tedy situace, kdy ještě uzly nejsou domluveny na společném symetrickém komunikačním klíči. Začátek situace nastává spuštěním jednoho uzlu. Je téměř nemožné aby se dva uzly spustily v přesně stejný čas, proto schéma ukazuje případ, kdy se jeden uzel probudí dříve. V tomto případě to je uzel B, proto jako první zjistí že nemá uložen komunikační AES klíč a odvysílá broadcastem žádost o klíč. V tuto chvíli se probudí druhý uzel, jenž zachytí toto vysílání. Protože ale také nevlastní klíč, musí si jej vytvořit.

Před samotným vytvořením klíče je nutné se domluvit na komunikačním klíči mezi akcelerátorem a mikrokontrolérem. Proto mikrokontrolér požádá akcelerátor o vygenerování náhodného čísla, které si připíše do paměti k již předchystanému



Obr. 4.3: Spuštění programu a ustanovení komunikačního klíče

klíči z ID akcelerátoru a mikrokontroléru. Stejně tak zapíše toto náhodné číslo do paměti akcelerátoru. Nyní je připraven komunikační klíč pro přenos dat pro rozhraní I2C. Stejným způsobem se vytvoří komunikační klíč i v druhém uzlu. Dalším krokem je vytvoření a zpracování pro odeslání AES klíče. Vytvořený klíč se zapíše do paměti akcelerátoru zašifrovaný jejich komunikačním klíčem. Následuje vytvoření veřejné verze asymetrického klíče a podpis zprávy. Z akcelerátoru dostaneme výsledný podpis a pomocí distribučního autentizačního klíče (sharedSHA) přečteme zašifrovanou zprávu. Takto získaná data se zkompletují a společně s podpisem se

odešlou uzlu který o klíč požádal. Příjemce, uzel B, odešle data do akcelerátoru, tím se uloží dešifrovaný klíč do jeho paměti a zkontroluje se podpis. Jestliže je vše v pořádku, uloží se klíč do paměti mikrokontroléru. Analogicky probíhá autentizace již zaběhlých uzlů v síti po ztrátě aktuálního klíče.

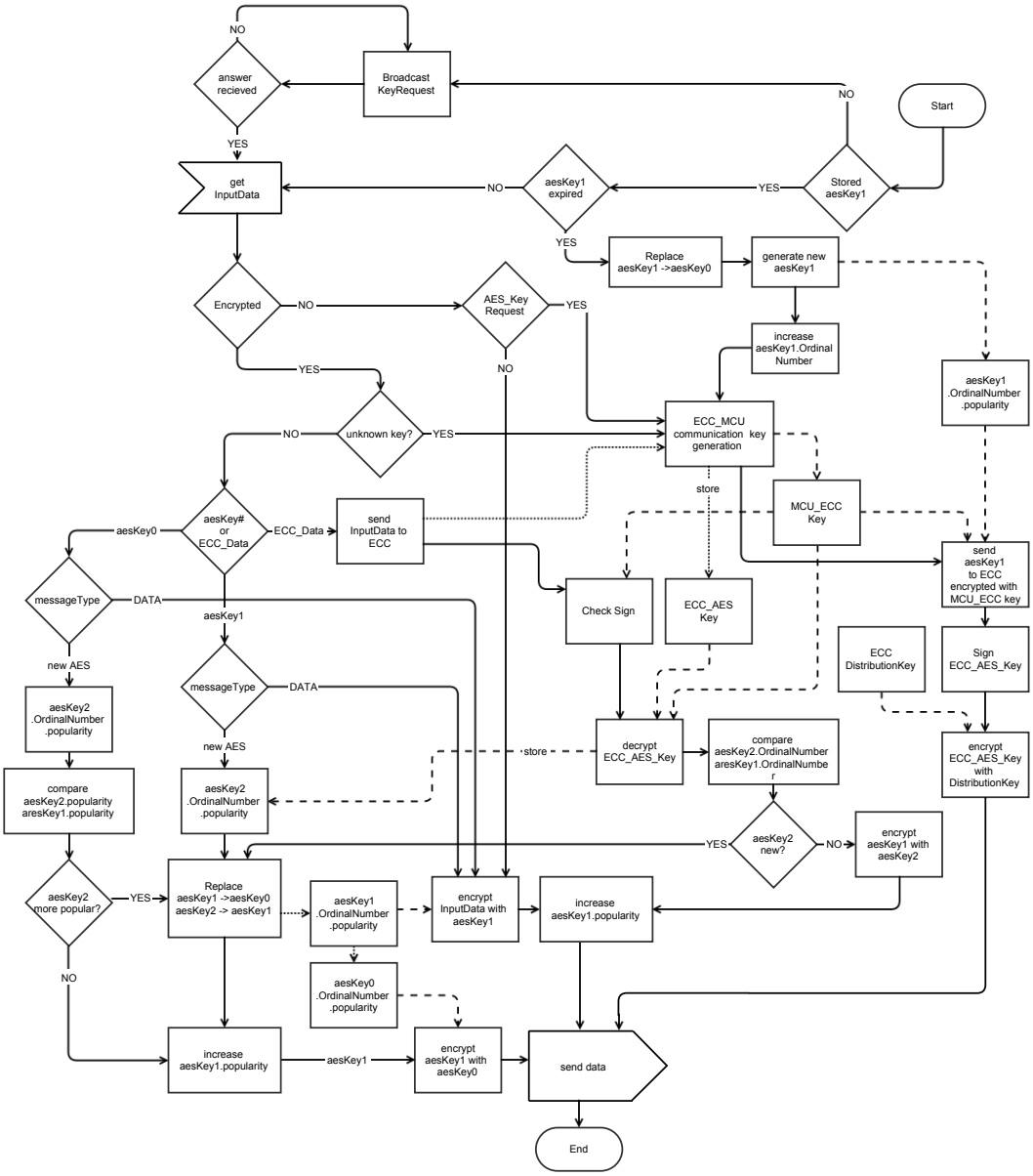
4.3.2 Vývojový diagram

Vývojový diagram ukazuje průběh zpracování jednotlivých zpráv. Jak je vyobrazeno v diagramu na obr. 4.4, program začína v poli Start a končí políčkem End. Jako první po startu programu se zkontroluje, zda je v uzlu uložen komunikační symetrický klíč v pozici aesKey1. Jestliže je uzel bez klíče, požádá o klíč jiné uzly. Podle přijaté zprávy klíč přijme, případně vytvoří vlastní a ten odešle, jak bylo ukázáno na obr. 4.3. Po každém spuštění programu se také kontroluje, zda nebylo dosaženo limitu použitelnosti aktuálního komunikačního AES klíče. V případě vypršení limitu se vytvoří nový klíč stejným principem jako je na obr. 4.3. Dále je zde celkem 10 druhů zpráv, které mohou být vstupem programu. Dle těchto dat se rozhoduje o dalším postupu.

Jako první se kontroluje zda jsou vstupní data šifrovaná. Pokud šifrována nejsou, může se jednat o žádost o klíč, případně jde o data vlastního uzlu, která se zašifrují aktuálním AES klíčem v pozici aesKey1 a odešlou se. Při každém šifrování či dešifrování se inkrementuje hodnota počtu použití tohoto klíče. Jestliže jde o žádost o klíč, postupuje se opět podle obr. 4.3. Jestliže jsou již ale vstupní data šifrována, postupuje se dále podle toho, kterým klíčem lze data dešifrovat.

Jednou možností je, že se jedná o autentizované předání klíče. V tomto případě se data zpracují akcelerátorem a porovná se pořadové číslo klíče. Jestliže přijatý klíč má vyšší pořadové číslo, nahradí se aktuální AES klíč v pozici aesKey0 tímto novým klíčem, který je uložen v pozici aesKey2. Jestliže je pořadové číslo přijatého klíče menší než aktuální, tímto přijatým klíčem se zašifruje aktuální klíč z pozice aesKey1 a odešle se zpět uzlu, který zprávu odeslal. Toto odeslání je ovšem podmíněno, že zpráva s novým klíčem byla úspěšně zašifrována společným distribučním klíčem a došlo k úspěšnému ověření podpisu. Tím se zabrání nechtěnému rozšíření klíče mimo vlastní síť.

Dalším typem zprávy můžou být data, případně nový AES klíč šifrovaný klíčem v pozici aesKey1. V případě, že jde o nový klíč se automaticky tento klíč uloží do pozice aesKey1 poté, co se původní aktuální klíč v pozici aesKey1 přemístil do pozice aesKey0. Navýší se popularitia přijatého klíče, zašifruje se nyní klíčem v pozici aesKey0 a takto šifrovaná zpráva se odešle dalšímu uzlu. Jestliže se jedná o data a né klíč, data se dešifrují, zpětně se zašifrují klíčem v pozici aesKey1, navýší se jeho počet použití a data se odešlou dalšímu uzlu.



Obr. 4.4: Diagram programu pro model s ECDSA

Poslední možností dešifrování přijaté zprávy může být aplikací klíče v pozici aesKey0. Opět se zkontroluje, zda přijatá zpráva obsahuje nový klíč, případně data. Jestliže jde o data, je potřeba předchozímu uzlu sdělit, že má neaktuální symetrický AES klíč. Předání nového klíče proběhne zašifrováním aktuálního klíče klíčem v pozici aesKey0 a přijatá data se dále zašifrují klíčem v pozici aesKey1 a odešlou se dalším uzlům v síti. Jestliže dešifrovaná zpráva obsahuje nový klíč, jedná se o klíč, který má stejné pořadové číslo, jako má aktuální klíč v tomto uzlu, a je

potřeba porovnat popularitu klíčů. Jestliže je popularita přijatého klíče vyšší, dojde k nahrazení aktuálního klíče nově přijatým klíčem v pozici aesKey2 a tento klíč se zašifruje původně aktuálním klíčem v pozici aesKey0.

Posledním možným typem zprávy je šifrovaná zpráva klíčem v pozici aesKey0, která obsahuje komunikační AES klíč, který má ale nižší popularitu než aktuální komunikační AES klíč v tomto uzlu v pozici aesKey1. Je tedy nutné uzlu, který zprávu odeslal oznámit neaktuálnost klíče a odeslat aktuální klíč sítě. Proběhne tedy šifrování aktuálního klíče v pozici aesKey1 klíčem v pozici aesKey0 a odešle se zpět tomuto uzlu.

5 REALIZACE NÁVRHU

Pro realizaci bylo navrženo celkové schéma protokolu, pomocí kterého se sestavil výsledný program pro mikrokontrolér. Pro vyobrazení situace byly použity message-sequence chart a flow diagram. Ke každému modelu byly pro lepší názornost vytvořeny oba diagramy. Ty byly vytvořeny pomocí webové aplikace Gliffy podporující následný export do souboru SVG, který byl dále zpracován programem Inkscape a vektorově exportován do PDF. Cílem práce bylo realizovat zabezpečení systému s nízkou energetickou náročností, k tomu byl pro tuto práci vybrán kit s mikrokontrolérem od společnosti Silicon Labs, jenž byl pro nízkoenergetické aplikace vyvinut. Mikrokontroléry disponují vlastním měřícím systémem energetické spotřeby, ke které je vydán software Energy Aware Profiler, ten snímá a vyobrazuje do grafu aktuální spotřebu energie. Práce je implementována na EFM32 Wonder Gecko starter kit. Tento kit obsahuje procesor ARM Cortex-M4F s maximálním taktem až 48 MHz. K dispozici máme taktéž 256 kB paměť Flash pro program a data a 32 kB paměti RAM. Akcelerátor se ke kitu připojuje pomocí rozšiřujícího portu, ve kterém je obsažen port rozhraní I2C. Pomocí diagramů byl následně vytvořen program v jazyce C ve vývojovém prostředí od výrobce kitu Simplicity Studio.

5.1 Nedostatky modelu ECDSA

Vybraný akcelerátor nabízí omezené možnosti zabezpečení s technologií eliptických křivek. ECDSA je jediná fukce s eliptickými křivkami, kterou akcelerátor nabízí. V tomto návrhu však nemá velké opodstatnění. Původně bylo v plánu využít akcelerátor pro asymetrické šifrování, ale jak se později ukázalo, vybraný akcelerátor tuto funkci nenabízí. Proto byl návrh s elektronickým podpisem představen jako možné i když ne příliš efektivní řešení. Podobnou úroveň bezpečnosti lze dosáhnout jiným protokolem a to za pomocí vytváření hashových zpráv. Proto se dále nebudeme v práci zabývat modelem s ECDSA, ale návrhem nové autentizační metody pomocí hashovacího algoritmu SHA-256. Systém distribuce klíče v síti zůstává stejný jako v předchozím případě. Změní se pouze metoda autentizovaného předání klíče mezi uzly. Abychom mohli porovnat, zda je akcelerátor v takovém případě přínosem, je potřeba navrhnout stejný model pro využití akcelerátoru a pro softwarovou implementaci hashovacích funkcí v mikrokontroléru.

5.2 Algoritmus SHA-256

Pro zachování nízké energetické náročnosti je důležité nejdříve vybrat vhodnou softwarovou interpretaci algoritmu SHA-256. Pro tuto práci byly vybrány 3 knihovny, které budou porovnány a na základě výsledků se vybere nejvhodnější. Pro tento účel byly vybrány knihovny PolarSSL, Open SSL a knihovna Brad Conte. Výběr nejvhodnější knihovny bude proveden změřením a porovnáním doby trvání výpočtu, počtu cyklů při výpočtu hashe z konstantní zprávy a spotřebou při výpočtu. Měření proběhne na různých kmitočtech procesoru mikrokontroléru. Délka zprávy bude odpovídat délce, se kterou počítá akcelerátor, protože bude na závěr vybraná knihovna porovnána s akcelerovaným výpočtem.

5.2.1 PolarSSL

Knihovna PolarSSL je původně vyvinuta z knihovny XySSL, vytvořené ve Francii v roce 2006 pod licencí GPL a BSD. Knihovna je cílena pro nízkoenergetická omezená zařízení. Knihovna obsahuje implementaci protokolů TLS a SSL a k nim potřebné kryptografické algoritmy. Poslední verze knihovny je 1.3.7, vydaná 3. května 2014.[13] Pro tuto práci byl z knihovny vybrán pouze algoritmus pro výpočet SHA-256 hashe. Výsledky měření algoritmu z této knihovny ukazuje tabulka 5.3.

Tab. 5.1: Tabulka měření výpočtu hashe užitím knihovny PolarSSL

frekvence CPU	**délka výpočtu**	**počet cyklů**	**změřená spotřeba**
$1\,MHz$	$7{,}36\,ms$	9 220	$23{,}58\,\mu J$
$7\,MHz$	$1{,}12\,ms$	9 220	$12{,}83\,\mu J$
$11\,MHz$	$0{,}64\,ms$	9 220	$10{,}36\,\mu J$
$14\,MHz$	$0{,}48\,ms$	9 220	$9{,}57\,\mu J$
$21\,MHz$	$0{,}48\,ms$	9 220	$11{,}89\,\mu J$
$28\,MHz$	$0{,}32\,ms$	9 220	$10{,}03\,\mu J$

5.2.2 OpenSSL

Projekt OpenSSL je kolaborativní pokus o vytvoření robustního toolkitu pro plnohodnotné nasazení v komerční sféře. Knihovna implementuje protokol SSL verze 2 a 3 a TLS verze 1 s plnými celosvětově vyvíjenými kryptografickými algoritmy. Dobrovolníci z celého světa knihovnu vyvíjejí pro internetovou komunikaci včetně dokumentace.[14] Na vývoji knihovny se podílí taktéž společnost Apple, která již

knihovnu používá ve svých produktech. Z této knihovy byly taktéž vybrány pouze algoritmy pro výpočet SHA-256 hashe a výsledek měření použití této knihovy ukazuje tabulka 5.1.

Tab. 5.2: Tabulka měření výpočtu hashe užitím knihovny OpenSSL

frekvence CPU	**délka výpočtu**	**počet cyklů**	**změřená spotřeba**
$1\,MHz$	$10{,}88\,ms$	13 280	$35{,}05\,\mu J$
$7\,MHz$	$1{,}6\,ms$	13 280	$18{,}89\,\mu J$
$11\,MHz$	$0{,}96\,ms$	13 280	$17{,}58\,\mu J$
$14\,MHz$	$0{,}8\,ms$	13 280	$17{,}48\,\mu J$
$21\,MHz$	$0{,}48\,ms$	13 280	$13{,}79\,\mu J$
$28\,MHz$	$0{,}48\,ms$	13 280	$12{,}27\,\mu J$

5.2.3 Brad Conte

Poslední knihovna je vytvořena člověkem jménem Brad Conte. Aktuálně je zaměstnanec ve společnosti Google. Knihovnu vyvinul ve svém volném čase během studií kde se mimo jiné věnoval a dále věnuje svým koníčkům jako Linux, kryptografie, programování C/C++, Python a diskrétní matematika. [15] Výsledky měření knihovny jsou zaznamenány v tabulce 5.2.

Tab. 5.3: Tabulka měření výpočtu hashe užitím knihovny Brad Conte

frekvence CPU	**délka výpočtu**	**počet cyklů**	**změřená spotřeba**
$1\,MHz$	$6{,}88\,ms$	8 466	$20{,}43\,\mu J$
$7\,MHz$	$0{,}96\,ms$	8 466	$9{,}7\,\mu J$
$11\,MHz$	$0{,}64\,ms$	8 466	$9{,}14\,\mu J$
$14\,MHz$	$0{,}48\,ms$	8 466	$8{,}53\,\mu J$
$21\,MHz$	$0{,}32\,ms$	8 466	$7{,}15\,\mu J$
$28\,MHz$	$0{,}16\,ms$	8 466	$3{,}73\,\mu J$

5.2.4 Vyhodnocení měření

Ze změřených výsledků je patrné, že nejefektivnější knihovnou je verze Brad Conte. Knihovna dosáhla nejlepších výsledků ve všech směrech. Samotná knihovna je v po-

rovnání s předchozími nejméně obsáhlá, celkem ve dvou souberech, z toho je jeden hlavičkový pro definice. Zdrojový kód tohoto algoritmu zabere přitom pouhých 140 řádků v souboru sha256.c. Takto sestavený algoritmus zabere v procesoru při výpočtu hashe ze zprávy dlouhé 88 bajtů celkem 8 466 hodinových cyklů, což přibližně odpovídá délce celého výpočtu. Při přepočtu na čas jsou zde v porovnání s naměřeným časem odchylky zejména proto, že hodinové impulzy nastavené frekvence procesoru mají určitou odchylku a délka periody se pohybuje v určité toleranci.

5.3 Verze protokolu s SHA-256

Tento protokol bude změněn z předchozího návrhu s ECDSA pouze v autentizovaném předání klíče jak již bylo řečeno výše. Samotná distribuce symetrického klíče v síti se řídí dle systému uvedeném v kapitole 4.2. Principem je sestavení šifrované relace mezi dvěma uzly na základě primárního klíče ROOTkey. Předpokladem je znalost ROOTkey mezi sousedními uzly, které si předávají symetrický komunikační klíč sítě. Tento ROOTkey může být sestaven pomocí identifikačních údajů uzlů nebo pomocí jednotného klíče, který bude uložen v každém uzlu v celé síti. Pro sestavení relace si tyto dva uzly předají náhodné číslo, které se skombinuje s ROOTkey, a z takto vytvořeného klíče se vytvoří hash, který bude sloužit jako relační klíč. Po sestavení relace se XORováním zašifruje symetrický klíč sítě a odešle žádajícímu uzlu, který stejným způsobem symetrický klíč dešifruje.

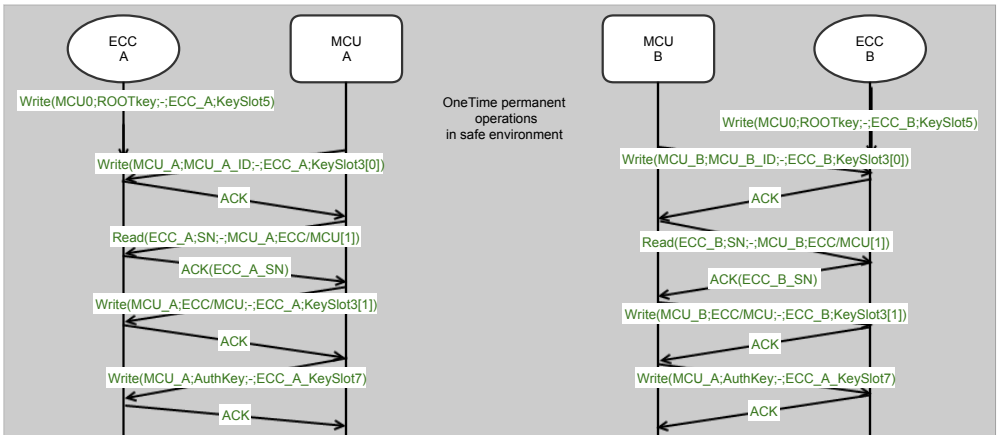
5.4 Schéma protokolu

Pro lepší představu a jako základ při tvorbě programu byly sestaveny diagramy pro tento protokol. K sestrojení byla opět použita webová aplikace Gliffy, ve kterém byly vytvořeny 2 diagramy – sekvenční a vývojový.

5.4.1 Sekvenční diagram

Před nasazením uzlu do sítě je opět potřeba zapsat do paměti mikrokontroléru, případně akcelerátoru potřebné klíče k šifrování, autentizaci a nyní i autorizaci dat. Tento zápis je potřeba vykonat v bezpečném prostředí, kde nehrozí riziko odposlechu. V tomto případě, jak je uvedeno na obr. 5.1 se zapisuje hlavní ROOTkey, který slouží k vytváření relačního klíče. Dále je potřeba si vyměnit mezi akcelerátorem a mikrokontrolérem vlastní ID, které budou sloužit jako klíč pro komunikaci mezi nimi. Posledním krokem je zapsání autorizačního klíče do akcelerátoru.

Po načtení klíčů do paměti je potřeba paměť akcelerátoru i mikrokontroléru uzamknout. Poté je uzel připraven pro připojení do sítě. Následuje spuštění uzlu

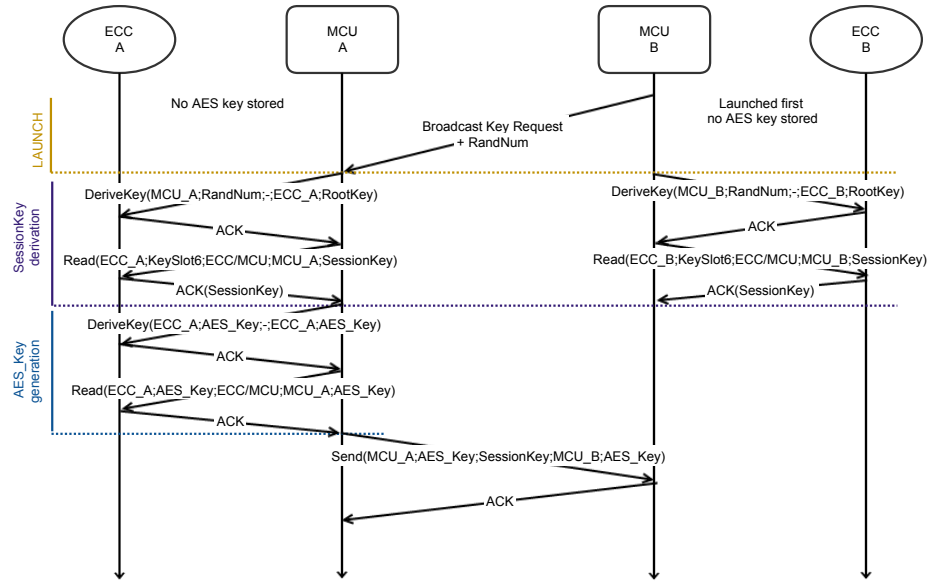


Obr. 5.1: Program pro bezpečné prostředí SHA-256 modelu

jak je na obr.5.2. V tomto demonstračním schématu je vyobrazena simulace spuštění dvou uzlů po sobě bez přítomnosti komunikačního symetrického klíče v pozici aesKey1. První probuzený uzel po zjištění absence AES klíče vygeneruje náhodné číslo, které pošle ve zprávě žádosti o klíč. Druhý probuzený uzel zprávu zachytí, protože ale také nemá uložený AES klíč, nějaký si vytvoří. Předtím si nejprve oba uzly z vytvořeného a přijatého náhodného klíče sestaví pomocí ROOTkey relační klíč výpočtem hashe jejich kombinace. V tomto okamžiku se vytvoří AES klíč funkcí DeriveKey, která vytvoří hash z náhodného čísla a aktuální hodnoty uložené v pozici pro vytváření nového AES klíče v akcelerátoru a uloží se opět na stejné místo v paměti akcelerátoru, odkud se šifrovaným čtením po úspěšné autorizaci z akcelerátoru přečte a uloží do mikrokontroléru na pozici aesKey1. Nyní je nový AES klíč připraven k zašifrování XORováním relačním klíčem. Po šifrování se zpráva odešle druhému uzlu. Protože tento uzel nemá žádný uložený klíč, nemusí nic porovnávat a dešifrovaný klíč si uloží do pozice aesKey1. Rozšíření tohoto klíče po síti proběhne jakmile se připojí další uzly, pouze s tím rozdílem, že se AES klíč již nevytváří, ale zašifruje relačním klíčem, který bude vytvořen z náhodného čísla odeslaného od nově připojeného uzlu.

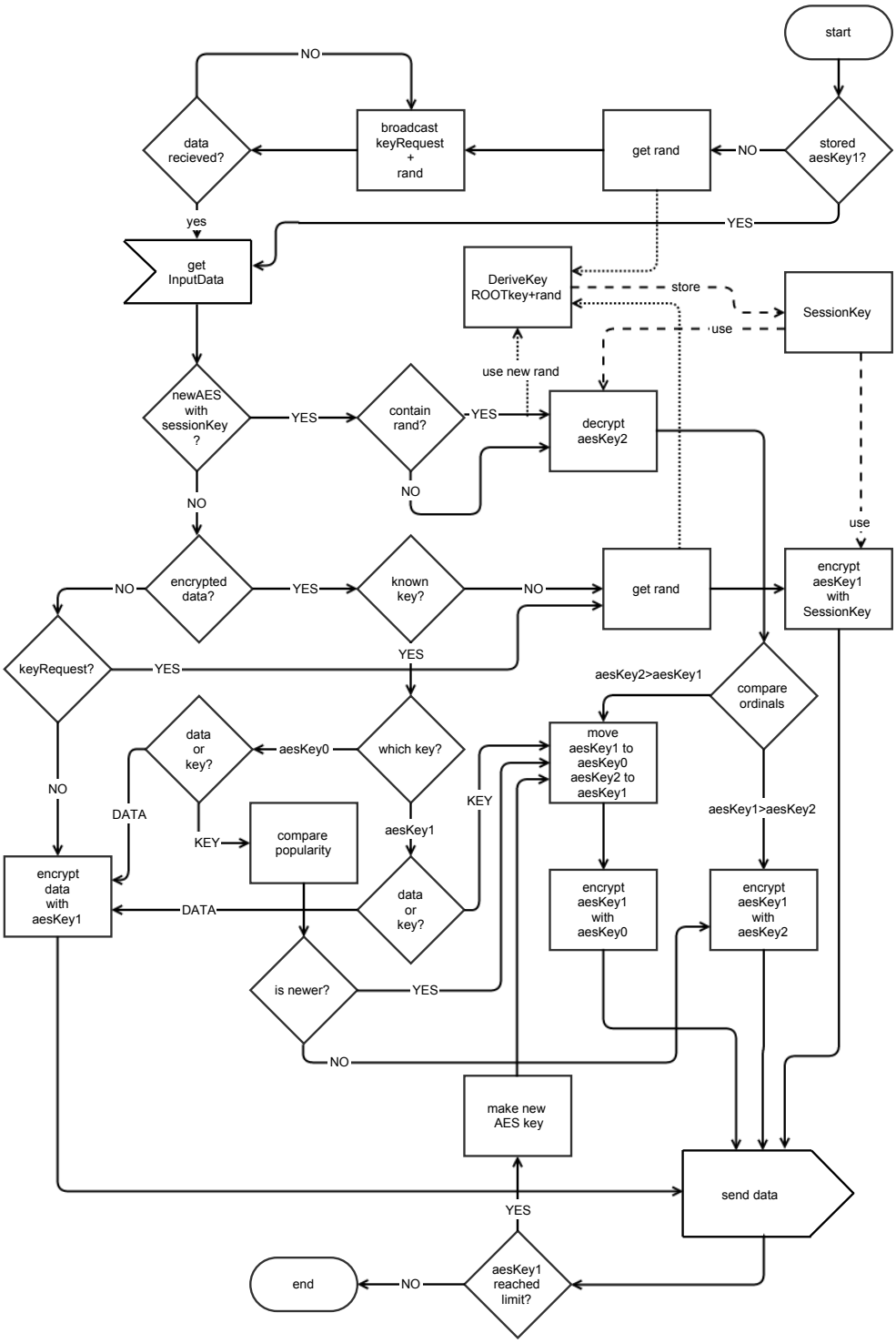
5.4.2 Vývojový diagram

Vývojový diagram, jak ukazuje obr. 5.3, je sestaven z původního návrhu distribuce klíčů a nového protokolu autentizovaného předání klíče. Diagram začíná políčkem start, kdy si nejprve zkontroluje zda již má uložený klíč AES v pozici aesKey1. Situace kdy jej uložený nemá je pouze při uvedení uzlu do provozu. Jestliže tedy tento klíč chybí, vytvoří se náhodné číslo, které se odešle uzlu spolu s žádostí o klíč a počká na odpověď. Mezitím si sestaví svůj relační klíč z tohoto náhodného a ROOTkey



Obr. 5.2: Program SHA-256 modelu

pomocí SHA-256. Další krok je stejný pro případ že již pozice aesKey1 je zaplněna. Po přijetí dat se nejprve zkontroluje, zda nejde o odpověď, případně o iniciaci změny AES klíče zašifrované pomocí relačního klíče. Jestliže to není odpověď na moji žádost, zpráva bude obsahovat i náhodné číslo pro sestavení nového relačního klíče. V obou případech se přijatý klíč dešifruje pomocí aktuálního relačního klíče. Následně se porovná aktuálnost klíče pomocí pořadového čísla. Jestliže je přijatý klíč novější, přemístí se aktuální klíč do pozice aesKey0 a nový klíč z pozice aesKey2 do aktuální pozice aesKey1.Dalším krokem je zašifrování tohoto aktuálního klíče klíčem v pozici aesKey0, případně jestli je přijatý klíč starší než aktuální, zašifruje se aktuální klíč tímto nově přijatým klíčem a v obou případech se data následně odešlou a program se ukončí. Zbývá ještě dalších 8 možných typů zpráv v případě, že se nejedná o zprávu šifrovanou relačním klíčem. Přijatá zpráva může být nešifrovaná žádost o přeposlání symetrického klíče. V tomto případě zpráva obsahuje náhodné číslo, ze kterého se opět vytvoří pomocí ROOTkey relační klíč mezi dvěma uzly a tímto relačním klíčem se žádajícímu uzlu odešle aktuální symetrický klíč. Dalším typem zprávy můžou být nešifrovaná data vlastního uzlu od senzorů. V tomto případě se pouze data zašifrují aktuálním symetrickým klíčem a odešlou do sítě. Další typy zpráv budou již jen šifrované a dělí se podle typu klíče, kterým je zpráva zašifrována. Jestliže jsou ale přijatá data zašifrována klíčem, který uzel nevlastní, vytvoří se opět náhodné číslo pro sestavení relace, které se přiloží k žádostí o klíč. Přijatá zpráva může být dále zašifrována klíči v pozici buď aesKey1 nebo aesKey0. V obou případech se dále zjišťuje, zda zpráva obsahuje nový klíč nebo data. Jestliže



Obr. 5.3: Schéma programu SHA-256

je to nový klíč šifrovaný klíčem aesKey0, uloží se do pozice aesKey2 a porovnají se hodnoty popularity aktuálního klíče aesKey1 a nového klíče aesKey2. Jestliže je přijatý klíč více rozšířený, přemístí se do aesKey1 a odešle se zašifrovaný klíčem

aesKey0 dále do sítě. Jestliže je ale přijatý klíč s menší popularitou, tímto klíčem se zašifruje aktuální klíč aesKey1 a odešle se zpět uzlu, který nám zprávu odeslal. Zbývají již pouze dva možné typy zpráv. V obou případech je zpráva zašifrována aktuálním klíčem v pozici aesKey1. Rozdílem je pouze obsah zprávy, kdy může zpráva obsahovat data nebo nový klíč. V případě dat se zpráva dešifruje a opět zašifruje aktuálním klíčem v pozici aesKey1. Toto přešifrování je nutné aby zpráva posílaná v síti nebyla stejná. Toto zajistí počítadlo použití aktuálního AES klíče, které se přikládá ke zprávě, případně se používá spolu s dalšími daty jako iniciační vektor pro šifrování AES. Posledním typem zprávy je nový šifrovaný symetrický klíč, který je zašifrovaný posledním aktuálním klíčem, tedy klíčem v pozici aesKey1. Takto šifrovaný klíč znamená, že se vždy jedná o nový klíč, proto se automaticky uloží do pozice aesKey1 po přesunutí klíče z aesKey1 do pozice aesKey0. Na závěr programu před ukončením je nutné zkontrolovat dosažení limitu použitelnosti aktuálního AES klíče. Jestliže klíč vyprší, je nutná obnova. Vytvoří se tedy nový AES klíč s vyšším pořadovým číslem a po přesunutí původního AES klíče z pozice aesKey1 do aesKey0 se zapíše do pozice aesKey1. Dále se zašifruje klíčem aesKey0 a odešle do sítě.

5.5 Softwarová implementace

V softwarové verzi se zabýváme pouze implementací návrhu autentizace pomocí SHA-256 hashe bez akcelerovaných výpočtů. Program je interpretován jako knihovna protokolu, jejíž hlavní 2 funkce jsou preinicializační a inicializační algoritmy. Preinicializační funkce se volá pouze jednou a to při prvním spuštění programu. Slouží pro počáteční ustanovení symetrického AES klíče mezi uzly v síti. Hlavní část protokolu je implementována v inicializační funkci, kde je vyřešena správa klíčů. Řeší se zde obnova AES klíče po jeho vypršení a autentizované předání aktuálního symetrického AES klíče v pozici aesKey1.

Pro porovnání softwarové verze s akcelerátorem byl také sestaven program, kdy se v cyklu v určité periodě spouštěly vybrané knihovny s výpočtem SHA-256 hashe spolu s akcelerovanou verzí tohoto výpočtu. Měření proběhlo pomocí EnergyProfileru a výsledky byly exportovány do souboru csv, ze kterého se následně vytvořili grafické závislosti těchto výpočtů. Měření proběhlo celkem pro 6 frekvencí procesoru mikrokontroléru.

5.6 Rozhraní

Pro účel měření spotřeby a demonstrace protokolu bylo potřeba určit a nastavit komunikační rozhraní. V případě akcelerátoru se jedná o rozhraní i2c, jehož konfiguraci

si ukážeme níže. Pro demonstraci bylo vybráno rozhraní UART, kterým předvedeme komunikaci mikrokontroléru s terminálem v počítači.

Akcelerátor je ke kitu s mikrokontrolérem připojen přes rozšiřující port piny PC4 pro SDA a PC5 pro SCL.

Rozhraní UART je nastaveno na piny PD0 pro TX a PD1 pro RX. Rychlost přenosu je nastavena na 9600 baudů za vteřinu. Rozhraní je nastaveno bez parity, odesílá se osmibitové slovo s jedním stopbitem. Komunikace s počítačem bude probíhat přes aplikaci RealTerm. Nastavení UARTu je v souborech UART.c a UART.h.

5.7 Aplikace

V následujících dvou kapitolách si předvedeme základní prvky programu a demonstraci autentizovaného předání klíče.

5.7.1 Program pro měření spotřeby akcelerátoru

Měření musí probíhat bez aktivního debug módu, toho lze dosáhnout pouze stisknutím tlačítka reset na kitu. Protože nemůžeme použít debugger pro spouštění jednotlivých funkcí programu, byla vytvořena smyčka, která dokola spouští jednu funkci za druhou s odstupem času a s mezipřechody do nízkoenergetického módu. Pro tento účel bylo potřeba přerušení. V tomto případě byl využit nízkoenergetický časovač LETIMER, který dovoluje probouzet procesor z nízkoenergetického módu EM2.

Nízkoenergetické periferie byly vybrány především pro přesnější měření spotřeby a lepší orientaci v grafu EnergyProfileru. Jednotlivé softwarové knihovny se spouštějí v intervalu 500 ms. V případě akcelerovaného výpočtu je potřeba naplnit paměť TempKey v akcelerátoru hodnotou, ze které se bude počítat hash.

V tomto případě se využije funkce Nonce v PassThru módu, abychom docílili výpočtu hashe ze stejné zprávy. Zpráva bude dlouhá 88 bajtů. Při manipulaci s akcelerátorem se využívá taktéž nízkoenergetický mód EM2 s přerušením z LETIMERu. LETIMER je nastaven v režimu dekrementace z úrovně nastavené v programu, která se průběžně podle potřeby mění. Časovač má nastaven zdroj hodin LFRCO, který nám dovoluje nastavit nejkratší puls 0,31 μs. Pro probuzení akcelerátoru je potřeba nastavit pin SDA na log.0 po dobu alespoň 60 μs, k tomu použijeme tedy 2 pulzy časovače.

Po provedení funkce nonce se počká 250 ms pro přehlednější odečítání z grafu a odešle se příkaz pro vytvoření hashe (MAC). Po odeslání příkazu se procesor přepne do režimu EM2 a počká se na provedení výpočtu. Výpočet typicky trvá minimálně 5 ms, tato hodnota se však mění a pro různé frekvence procesoru se čas čekání nastavuje individuálně. Po provedení výpočtu v akcelerátoru se z něj výsledný

hash přečte a mikrokontrolér přejde do módu EM2. Po dalších 250 ms se vypne i2c a opět přejde do EM2.

5.7.2 Program protokolu s využitím hashe bez akcelerátoru

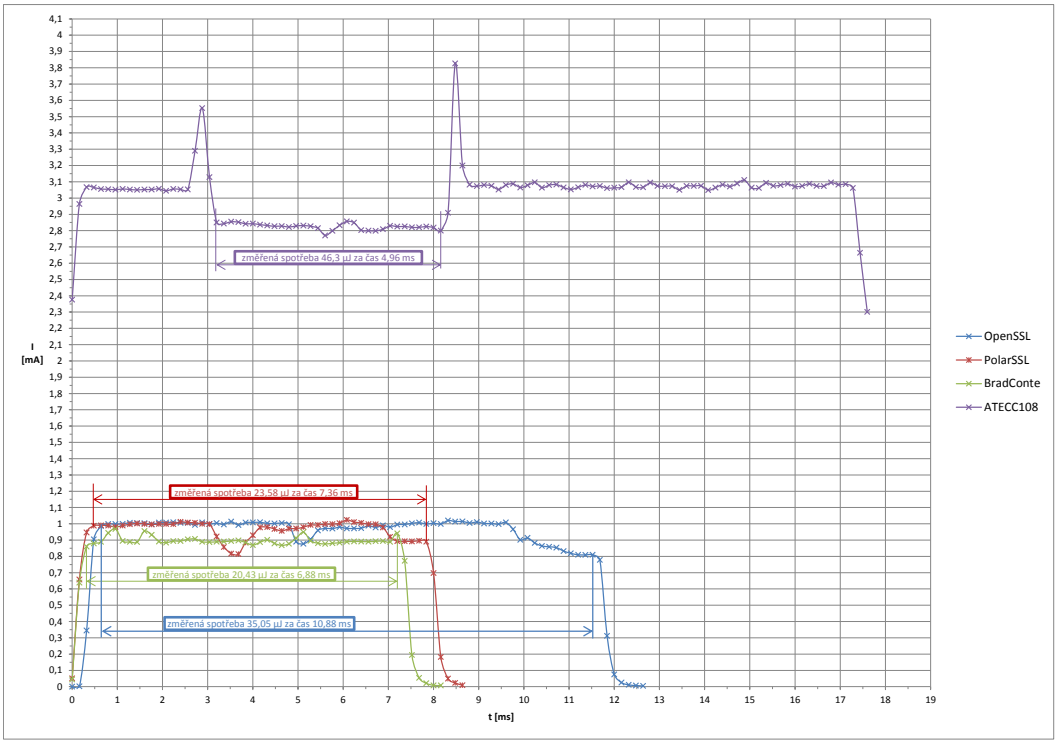
Program je rozepsán celkem v pěti částech. První částí je nastavení UARTu v souborech UART.c a UART.h jak bylo popsáno výše. Další část se týká hashovací funkce. Ta byla vybrána na základě měření a její obsah je v souborech sha256.c a sha256.h. Pro spolupráci EnergyProfileru s programem je zde nastavení SWO v souborech SetupSWO.c a SetupSWO.h. Obsah této funkce je převzat z programu EnergyProfileru. Předposlední částí je funkce pro symetrické šifrování, ta je předpřipravena v souborech AES.c a AES.h. Poslední částí je nastavení inicializací, to je v souborech crypto.c a crypto.h. Podrobný popis je připsán přímo v kódu. Běh programu je nastaven podle schématu programu na obr. 5.3 Program lze pomocí malých úprav přizpůsobit pro komunikaci dvou mikrokontrolérů mezi sebou pomocí rozhraní UART.

6 ENERGETICKÁ ANALÝZA

V této části bude popsána analýza měření algoritmů SHA-256. Popsán bude akcelerátor a jeho spotřeba v porovnání se softwarovými verzemi a to na různých frekvencích procesoru mikrokontroléru. Bude vyhodnocena energetická náročnost akcelerátoru a délky výpočtů hashe v závoslosti na různých kmitočtech.

6.1 Energetická náročnost akcelerátoru

Pro představu je zde graf spotřeby výpočtu 88 bajtů dlouhé zprávy na frekvenci procesoru 1 MHz.6.1 V grafu je zobrazena křivka spotřeby akcelerátoru společně se zadáním příkazu MAC (přibližně do 3. ms), dalších 5 ms je výpočet samotného hashe v akcelerátoru, zatímco je mikrokontrolér v nízkoenergetickém módu EM2 a čeká na přerušení nastavené pro dobu výpočtu udanou z dokumentace čipu a nastavenou na nejkratší možný čas nutný pro odeslání odpovědi z akcelerátoru. Dále je výsledný hash z akcelerátoru vyčten. Jde celkem o přečtení 259 bajtů rozhraním i2c. Délka celého výpočtu s komunikací je 17,28 ms a celkem bylo spotřebováno za tuto dobu 171,45 μJ. Délka samotného výpočtu hashe v akcelerátoru je 4,96 ms a za tuto dobu bylo spotřebováno 46,3 μJ



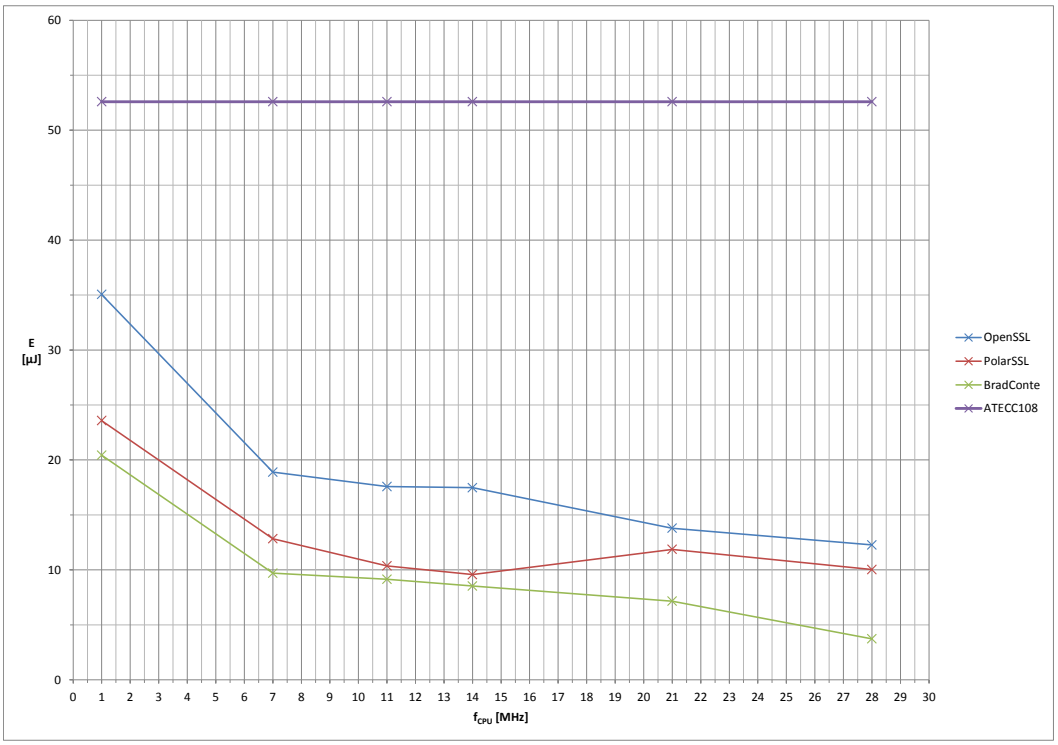
Obr. 6.1: Graf výpočtu SHA-256 pro 1 MHz

Další tři křivky vyobrazují výpočet hashe pomocí popsaných knihoven. Pro představu jak probíhalo měření je v grafu zaznamenáno i probuzení z EM2, kdy je vidět kompletní oblast spotřeby každé knihovny. Výsledky spotřeby jednotlivých knihoven a akcelerátoru jsou v následující tabulce:6.1.

Tab. 6.1: Tabulka měření výpočtu hashe pro 1 MHz

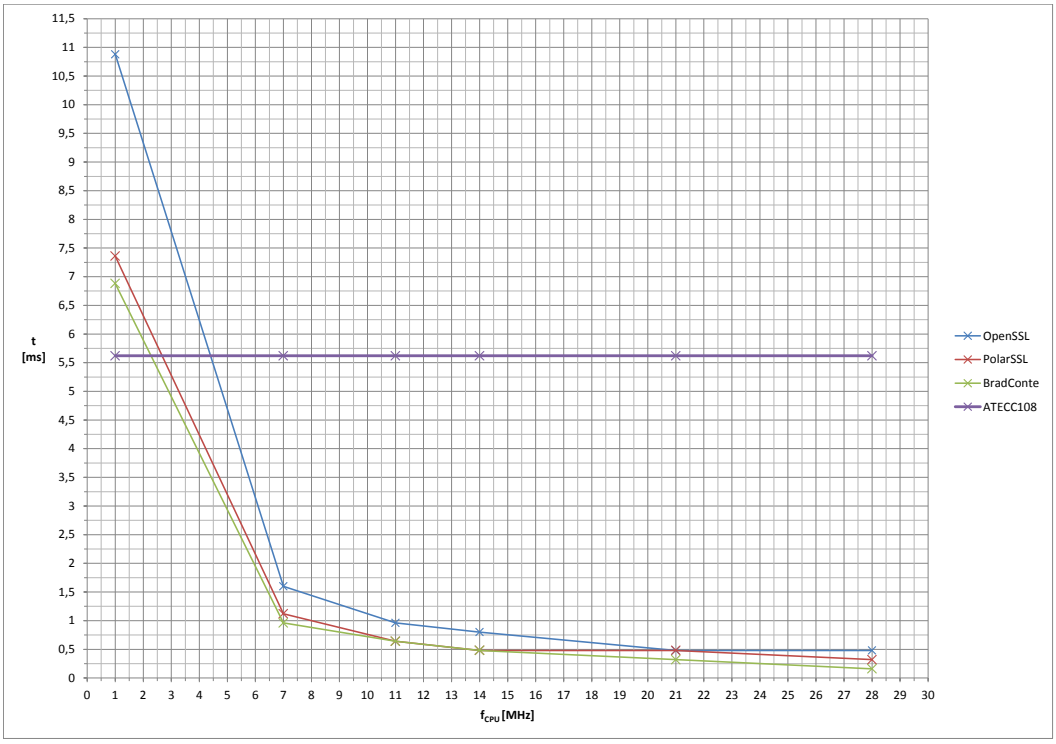
Knihovna	**délka výpočtu**	**počet cyklů**	**změřená spotřeba**
OpenSSL	10,88 ms	13 280	35,05 μJ
PolarSSL	7,36 ms	9 220	23,58 μJ
BradConte	6,88 ms	8 466	20,43 μJ
ATECC108	4,96 ms	21 367	46,3 μJ

Po změření výpočtu pro všechny frekvence procesoru byl vypracován graf závislosti spotřeby výpočtů na frekvenci.6.2 V grafu vyšla nejnižší spotřeba funkce PolarSSL pro frekvenci 14 MHz, což by neměla, protože při vyšší frekvenci sice dochází ke zvýšení spotřeby, ale doba výpočtu je podstatně kratší. Chyba měření může



Obr. 6.2: Graf spotřeby jednotlivých knihoven v závislosti na frekvenci procesoru

být způsobena především nepřesným měřícím ústrojím, kvůli malé vzorkovací frekvenci, která je 0,16 ms. Proto také dochází ke zkresleným hodnotám při měření dély výpočtu v závislosti na frekvenci procesoru jak uvádí graf 6.3. Doby výpočtu při vyšších frekvencích se pohybují v rozmezí desetin milisekund, což odpovídá jednotkám vzorků za tuto dobu. Tento příklad ukazuje graf měření spotřeby výpočtu

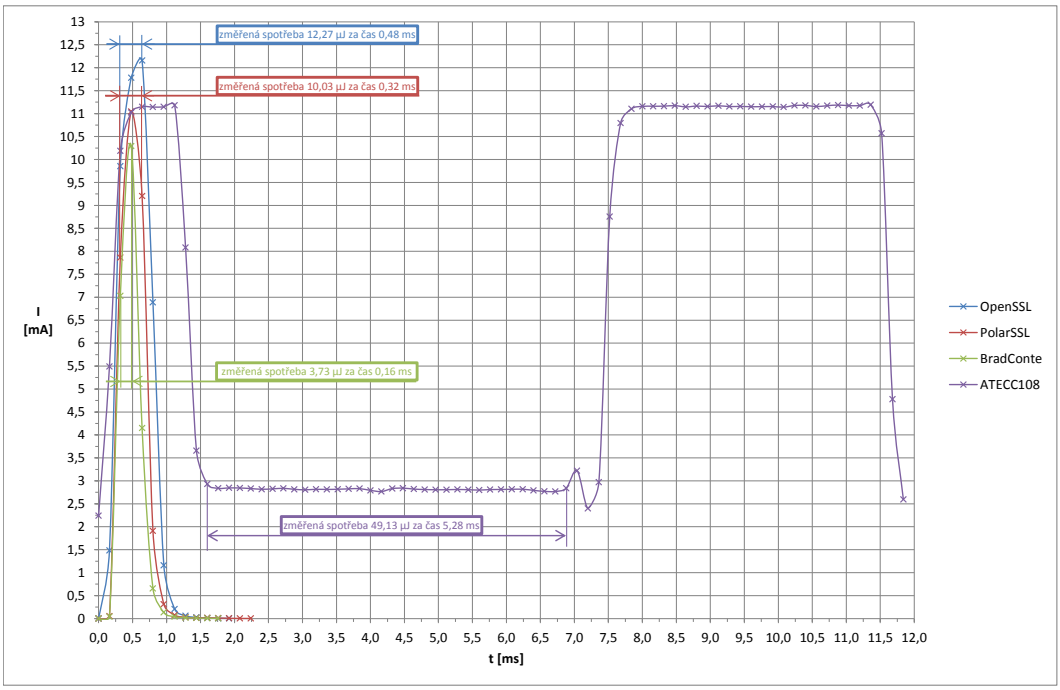


Obr. 6.3: Graf doby výpočtu jednotlivých knihoven v závislosti na frekvenci procesoru

hashe knihoven pro frekvenci 28 MHz.6.4 Jednotlivá měření jsou natolik krátká, že samotné výpočty jsou vyznačeny dvěma až čtyřmi vzorky měřícího přístroje.

6.2 Energetická náročnost protokolu s SHA-256

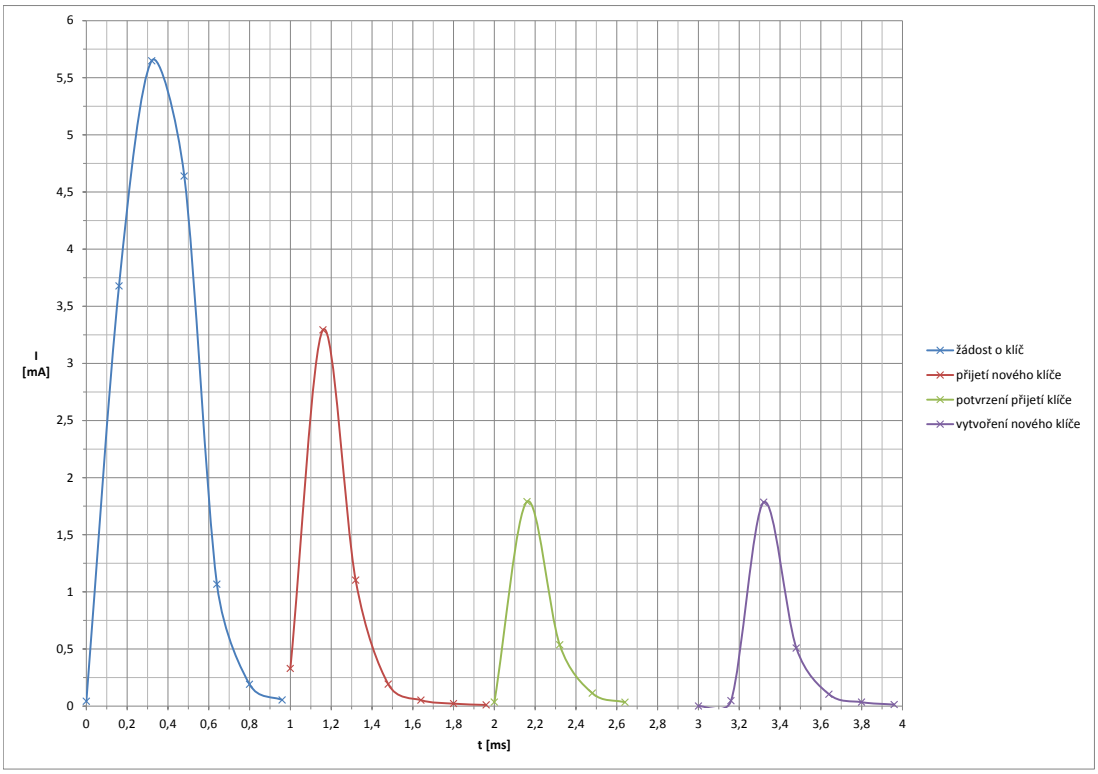
Měření energetické náročnosti protokolu autentizovaného předání klíče probíhá na jednotlivých úsecích programu. Pro tento účel byla po předchozí analýze vybrána nejvhodnější dostupná knihovna pro výpočet SHA-256 hashe, jíž je knihovna BradConte. Při měření byl program doplněn o vypínání procesoru mezi každou měřenou funkcí. Pro probouzení procesoru se využívá nízkoenergetický časovač LETIMER s nastaveným rozestupem 500 ms. Po každém přerušení se časovač vypíná pro získání odpovídajících hodnot spotřeby. Ve výsledném grafickém znázornění 6.5 jsou



Obr. 6.4: Graf měření spotřeby jednotlivých knihoven pro frekvenci 28 MHz

vyznačeny postupně jednotlivé průběhy proudů pro každou část protokolu, které byly vyňaty z celkového průběhu programu. Mezi těmito úseky se nachází komunikace přes UART s PC, odkud byly postupně odesílány instrukce pro simulaci jednotlivých částí protokolu. Začátky funkcí byly pro přehlednost posunuty na nejbližší celé číslo, jinak délka celého běhu závisí na použité komunikaci a podle situace která nastane.

První úsek grafu znázorňuje funkci preInit, která slouží pro prvotní ustanovení klíče. Jedná se o vytvoření žádosti o klíč společně s vygenerováním náhodného čísla sloužící jako AES klíč, dále se vytvoří náhodné číslo sloužící pro ustanovení relačního klíče, který se následně také vytvoří. Náhodné číslo se skombinuje s ROOTkey a z této kombinace se vytvoří hash sloužící jako relační klíč. Pomocí tohoto klíče se tedy XORováním zašifruje předem vytvořený vlastní AES klíč společně s jeho parametry jako pořadové číslo, počet použití a popularita, který se takto šifrovaně odešle nejbližšímu uzlu. První bajt zprávy vždy označuje o jakou zprávu se jedná, proto tady je nastaven na informaci o žádost klíče. Ke zprávě je přiložen ještě jeden bajt sloužící jako kontrolní značka vlastního uzlu. Jedná se o XORovaný bajt zašifrovaného klíče, jehož první čtyři bity označují bajt který bude zaxorován stejným



Obr. 6.5: Graf měření spotřeby protokolu

způsobem vybraným bajtem relačního klíče. Celkem tedy odesílaná zpráva obsahuje 40 bajtů. Spotřeba sestavení takové zprávy je celkem 8,1 μJ a doba trvání této funkce je 0,96 ms.

Druhý úsek ukazuje spotřebu potřebnou pro akceptaci tohoto klíče. Nejprve se zkontroluje o jaký typ zprávy se jedná, v tomto případě již víme že zpráva obsahuje žádost o klíč, která již jeden klíč nabízí. Dále se zkontroluje stejným výpočtem kontrolní bajt. Z přijatého náhodného čísla se vytvoří opět pomocí ROOTkey a hashe relační klíč, kterým se přijatý klíč dešifruje a uloží do paměti. Tato situace spotřebuje celkem 2,65 μJ za dobu 0,96 ms.

Další křivka znázorňuje spotřebu vytvoření zprávy s odpovědí pro daný uzel. Jedná se o vytvoření kladné odpovědi s patřičným označením a kontrolním bajtem. Tato funkce trvá 0,64 ms a spotřebuje 1,31 μJ.

Poslední část demonstruje vypršení klíče a jeho obnovu. Ta se porvádí pomocí původního klíče a náhodného čísla ze kterých se vytvoří hash. Takto vytvořený klíč se zašifruje pomocí původního klíče a odešle dalším uzlům. Spotřebuje se při tom 1,32 μJ za 0,96 ms.

7 ZÁVĚR

Zadáním bakalářské práce bylo vytvoření protokolu pro zabezpečení komunikace vestavěného systému. Pro tento účel byl vybrán hadrwarový akcelerátor, jehož výpočetní potenciál měl být využit pro výpočty asymetrických šifer. S tímto akcelerátorem měl být navržen systém autentizace mezi jednotlivými uzly bezdrátových senzorových sítí s ohledem na nízkou energetickou náročnost.

Při realizaci návrhu jsem postupoval podle dostupných prostředků, které akcelerátor nabízí. Již z počátku nastaly potíže způsobené nedostupnou dokumentací vybraného akcelerátoru. Po zjištění podmínek pro získání dokumentace bylo nutné podepsat smlouvu týkající se dodržení mlčenlivosti. Před získáním dokumentace jsem vycházel z dokumentace předešlé verze čipu, ze kterého byl používaný akcelerátor vyvinut. Získáním aktuální dokumentace jsem se seznámil s dostupnými funkcemi akcelerátoru, kde ale chyběly předpokládané funkce asymetrické kryptografie. Protože akcelerátor nabízí z oblasti asymetrické kryptografie pouze algoritmus elektronického podpisu ECDSA, byl navržen protokol ve kterém mohl být aplikován.

S ohledem na energetickou náročnost byl navržen druhý protokol s SHA-256, který již asymetrickou kryptografii nevyužíval, ale dal se aplikovat s pomocí dostupných funkcí akcelerátoru. Tento protokol byl navržen také pro softwarovou verzi, jehož spotřeba měla být srovnána s akcelerovaným protokolem. Pro oba případy, ECDSA i SHA bylo navrženo nastavení konfigurační zóny, celkem 512 bitů pro 16 paměťových slotů akcelerátoru, kterou bylo nutné pro další použití akcelerátoru nevratně uzamknout.

Po zhotovení všech návrhů protokolů a ujištění se o správnosti nastavení konfigurační zóny, bylo pro analýzu energetické náročnosti a dokončení praktické části nutné konfigurační zónu uzamknout. Uzamčení mělo podle dokumentace zpřístupnit zápis do paměťových slotů, které sloužily pro ukládání a manipulaci s klíči. Po několika neúspěšných pokusech o zápis do paměti bylo zjištěno, že zápis stále podléhá některému nastavení konfigurační zóny. Pro zápis je nutné znát tovární nastavení uložených klíčů, které bylo možno získat na vyžádání u výrobce. Výrobce nám však do této chvíle potřebné informace nepředal. Z tohoto důvodu nebylo možné realizaci návrhu s použitím akcelerátoru dokončit.

Řešením nastalé situace bylo srovnání energetické náročnosti akcelerovaných a softwarových výpočtů nejčastěji používané funkce systému, kterou je algoritmus SHA-256.

Analýzou jsme dospěli k závěru, že s ohledem na energetickou náročnost systému se nejlépe jeví softwarová knihovna BradConte, kterou jsem následně aplikoval do navrženého protokolu. Měřením bylo zjištěno, že tato knihovna má nejnižší spotřebu při frekvenci 28 MHz. Z důvodu omezení měřícího přístroje jsem měření protokolu

provedl při frekvenci 14 Mhz.

Z výše uvedených důvodů nebylo možné systém s akcelerovanými výpočty bez znalosti potřebných klíčů dále analyzovat.

Vzhledem k překážkám, které nastaly v průběhu při provádění práce by bylo vhodné se problematikou zabývat i nadále při rozšiřujícím studiu. Námětem pro další studium problematiky kryptografie v bezdrátových senzorových sítích by bylo srovnání energetické náročnosti akcelerovaných a softwarových řešení, jenž je možné až po získání potřebných údajů od výrobce čipu.

LITERATURA

[1] DOSTÁLEK, L., VOHNOUTOVÁ, M. *Velký průvodce infrastrukturou PKI a technologií elektronického podpisu*, Brno: Computer Press, 544, 2006, p. 24–26 ISBN 80-251-0828-7 [cit. 4. 6. 2014].

[2] KOO, W., K. : Implementation and Analysis of New Lightweight Cryptographic Algorithm Suitable for Wireless Sensor Networks. *Information Security and Assurance, 2008. ISA 2008. International Conference on*, Busan, 586, 2008, p. 73–76 ISBN 978-0-7695-3126-7 [cit. 4. 6. 2014].

[3] KLEIDERMACHER, D., KLEIDERMACHER, M. *Embedded Systems Security: Practical Methods for Safe and Secure Software and Systems Development*, Oxford, 416, 2012, ISBN 978-0-12-386886-2 [cit. 4. 6. 2014].

[4] KRIEF, F. *Communicating Embedded Systems*, Velká Británie, 340, 2010, ISBN 978-1-84821-144-5 [cit. 4. 6. 2014].

[5] MORÁVEK, P., KOMOSNÝ, D. *Asymetrická kryptografie v bezdrátových senzorových sítích* [online]. 2009, [cit. 4. 6. 2014]. Dostupné z URL: <`http://access.feld.cvut.cz/view.php?cisloclanku=2009110005`>.

[6] *ATECC108 Atmel CryptoAuthentication SUMMARY DATASHEET* [online]. 2013 [cit. 4. 6. 2014]. Dostupné z URL: <`http://www.atmel.com/Images/Atmel-8873S-CryptoAuth-ATECC108-Datasheet-Summary.pdf`>.

[7] *MSP430x5xx and MSP430x6xx Family User's Guide* [online]. 2008, poslední aktualizace 2013 [cit. 4. 6. 2014]. Dostupné z URL: <`http://www.ti.com/lit/ug/slau208m/slau208m.pdf`>.

[8] BRACAMONTES, R. *4 Different Authentication Models—Which One is Right for You?* [online]. 2013, [cit. 4. 6. 2014]. Dostupné z URL: <`http://atmelcorporation.wordpress.com/2013/03/25/4-different-authentication-models-which-one-is-right-for-you/`>.

[9] *ATSHA204 Atmel CryptoAuthentication DATASHEET* [online]. 2013, [cit. 4. 6. 2014]. Dostupné z URL: <`http://www.atmel.com/Images/Atmel-8740-CryptoAuth-ATSHA204-Datasheet.pdf`>.

[10] *ADVANCED ENCRYPTION STANDARD (AES)* [online]. 2001, [cit. 4. 6. 2014]. Dostupné z URL: <`http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`>.

[11] *Digital Signature Standard (DSS)* [online]. 2013, [cit. 4. 6. 2014]. Dostupné z URL: <http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf>.

[12] *The Elliptic Curve Digital Signature Algorithm (ECDSA)* [online]. 2001, [cit. 4. 6. 2014]. Dostupné z URL: <http://cs.ucsb.edu/~koc/ccs130h/notes/ecdsa-cert.pdf>.

[13] *Knihovna PolarSSL* [online]. [cit. 4. 6. 2014]. Dostupné z URL: <https://polarssl.org/>.

[14] *Knihovna OpenSSL* [online]. [cit. 4. 6. 2014]. Dostupné z URL: <https://www.openssl.org/about/>.

[15] *Knihovna Brad Conte* [online]. [cit. 4. 6. 2014]. Dostupné z URL: <http://bradconte.com/about>.

[16] Atmel Corporation *Atmel CryptoAuthentication DATASHEET*, San Jose, 92, 2013, [cit. 4. 6. 2014].

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

ECC Kryptografie nad eliptickými křivkami – Eliptic curve cryptography

ECDSA Protokol digitálního podpisu s využitím eliptických křivek – The Elliptic Curve Digital Signature Algorithm

EEPROM Elektricky vymazatelná PROM paměť – Electrically Erasable PROM

FIPS Federální (USA) norma pro zpracování informací – Federal Information Processing Standard

HMAC Typ autentizačního kódu zprávy s použitím hešovací funkce – Keyed-hash Message Authentication Code

MAC autentizační kód zprávy – Message Authentication Code

NIST Národní institut standardů a technologie – National Institute of Standards and Technology

NSA národní bezpečnostní agentura – National Security Agency

OTP Jednorázově programovatelná permanentní paměť – One Time Programmable

PKI Infrastruktura veřejných klíčů – Public Key Infrastructure

RNG Generátor náhodných čísel – Random Number Generator

SHA-256 Jednocestná hešovací funkce o pevné délce 256 bitů – Secure Hash Algorithm

AES Standard pokročilého šifrování – Advanced Encryption Standard

WSN bezdrátové senzorové sítě – Wireless sensor network

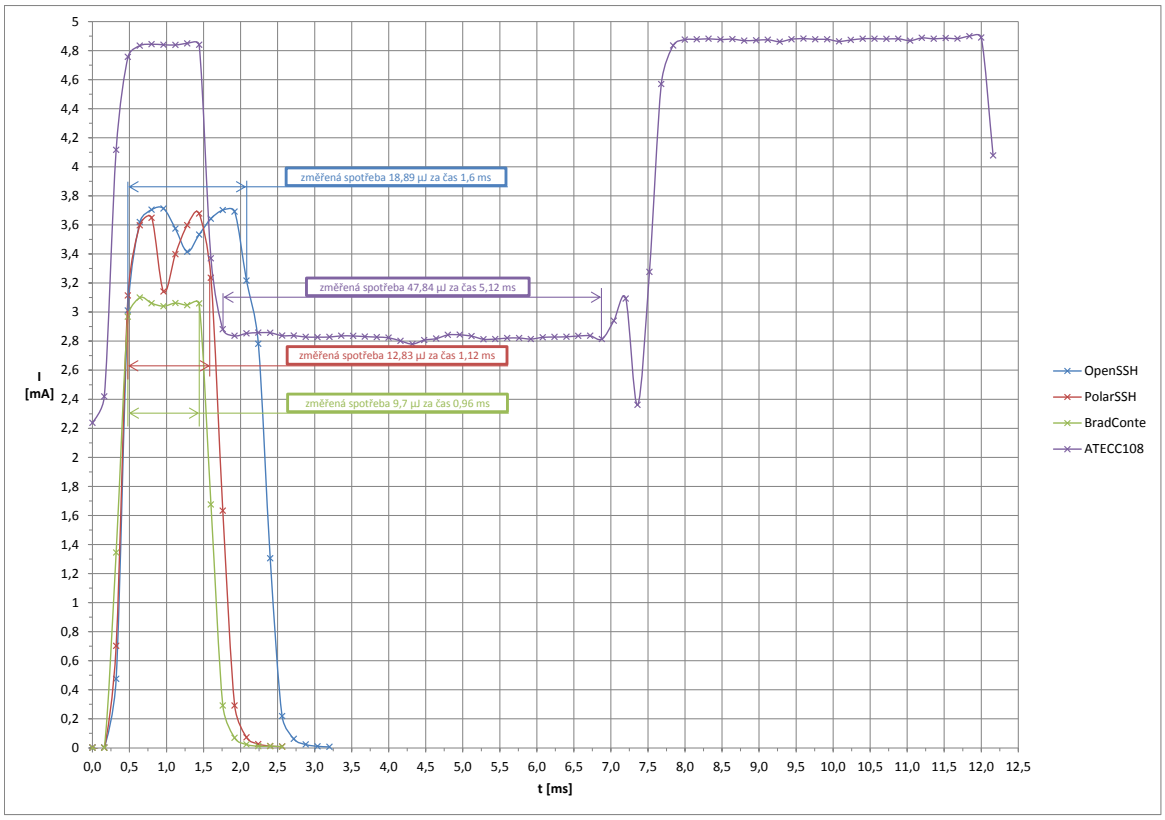
DSA algoritmus digitálního podpisu – Digital signature algorithm

ECDH Protokol Diffieho-Hellmana s elyptickými křivkami – Elliptic curve Diffie–Hellman

SEZNAM PŘÍLOH

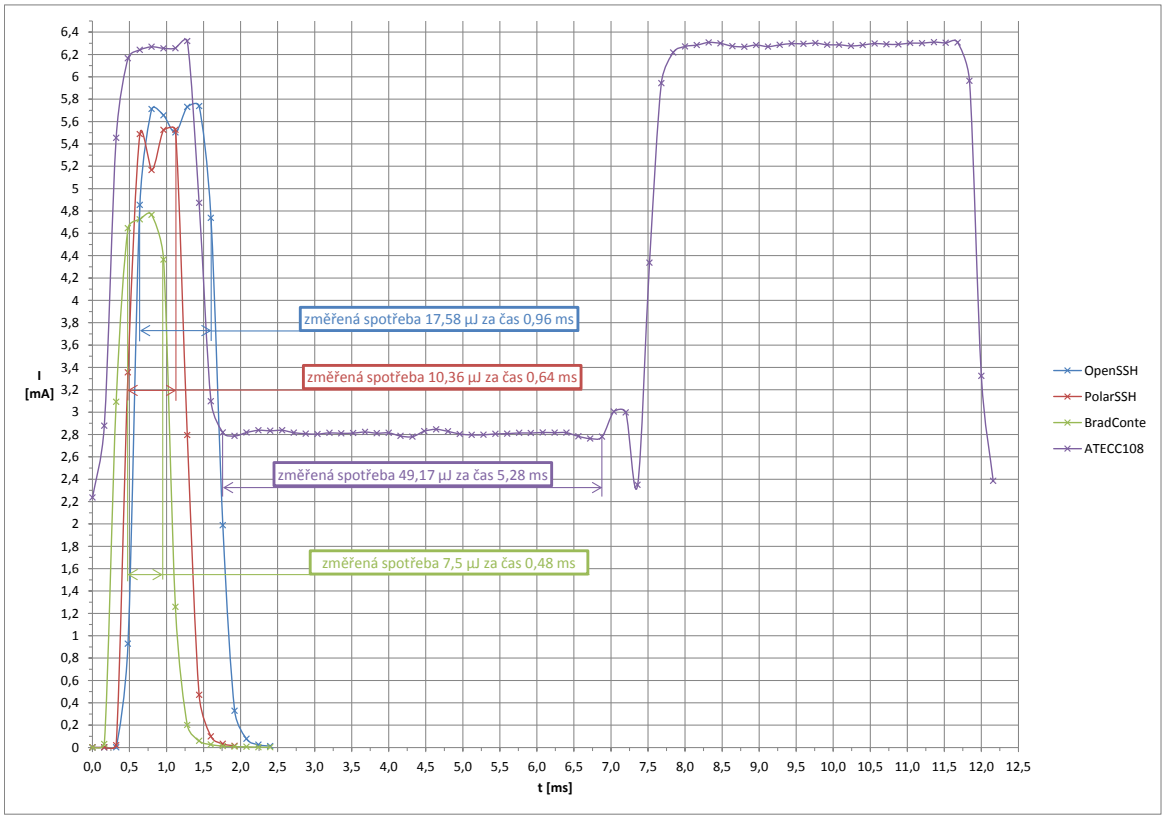
A MĚŘENÍ KNIHOVEN SHA ALGORITMU PRO ZBYLÉ FREKVENCE

A.1 Graf měření pro 7 MHz



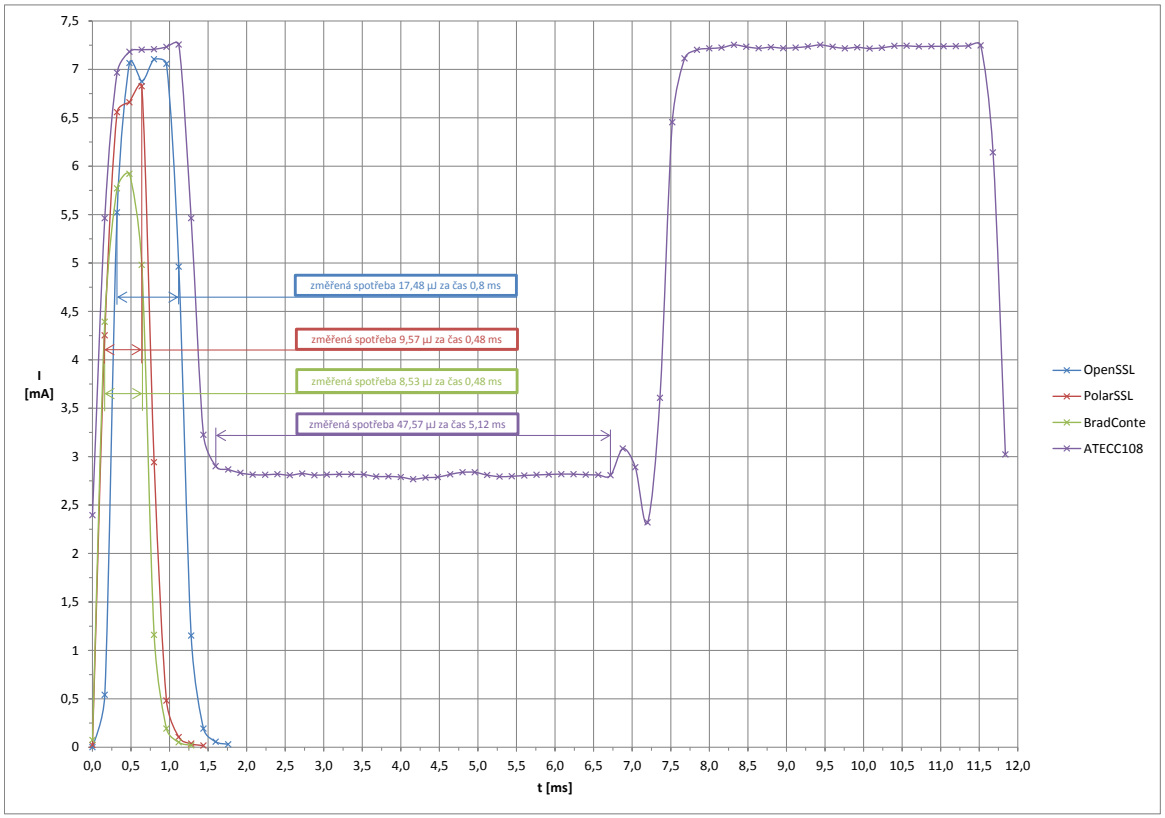
Obr. A.1: Graf měření spotřeby jednotlivých knihoven pro frekvenci 7 MHz

A.2 Graf měření pro 11 MHz



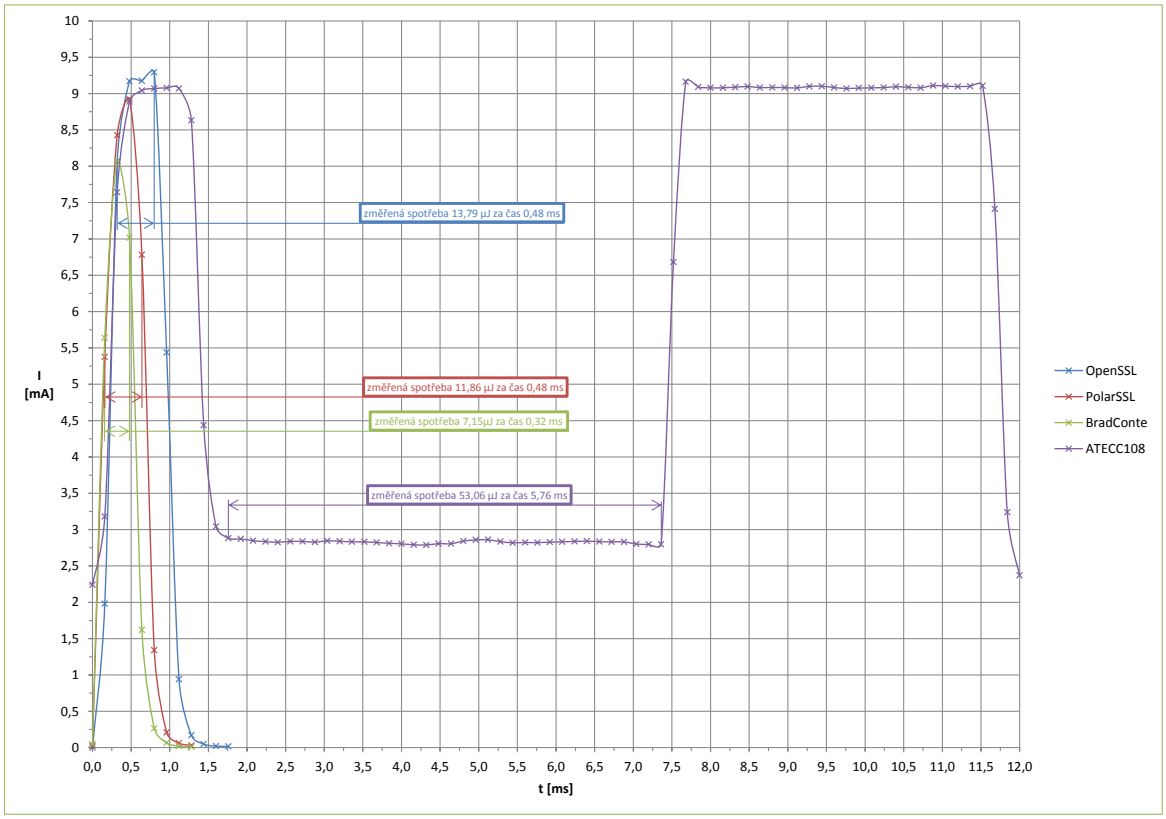
Obr. A.2: Graf měření spotřeby jednotlivých knihoven pro frekvenci 11 MHz

A.3 Graf měření pro 14 MHz



Obr. A.3: Graf měření spotřeby jednotlivých knihoven pro frekvenci 14 MHz

A.4 Graf měření pro 21 MHz



Obr. A.4: Graf měření spotřeby jednotlivých knihoven pro frekvenci 21 MHz

B OBSAH PŘILOŽENÉHO CD

Na CD se nachází zdrojový kód protokolu s SHA-256 bez akcelerátoru a zdrojový kód programu pro měření jednotlivých knihoven. Dále jsou zde vyexportované tabulky měření z programu EnergyAware Profiler a vyhotovené grafy ve formátu PDF.