VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

KRYPTOGRAFICKÉ KOPROCESORY PRO ZABEZPEČENÍ KOMUNIKACE IOT MIKROKONTROLERŮ

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — Michal Český

AUTHOR

VEDOUCÍ PRÁCE — doc. Mgr. Karel Slavíček, Ph.D.

SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Telekomunikační a informační systémy**

Ústav telekomunikací

Student: Michal Český ***ID:*** 230789

Ročník: 3 ***Akademický rok:*** 2022/23

NÁZEV TÉMATU:

Kryptografické koprocesory pro zabezpečení komunikace IoT mikrokontrolerů

POKYNY PRO VYPRACOVÁNÍ:

V oblasti IoT se používá řada mikrokontrolerů, které nemají dostatečný výpočetní výkon pro kryptografické zabezpečení datové komunikace. Cílem práce je prozkoumat dostupné kryptografické koprocesory, např. řady Microchip ATECC, a možnosti jejich využití pro podporu zabezpečení komunikace mikrokontrolerů, zejména protokolu TLS 1.2 a TLS 1.3. Úkolem návazné bakalářské práce bude praktická implementace zvoleného kryptografického koprocesoru.

DOPORUČENÁ LITERATURA:

podle pokynů vedoucího práce

Termín zadání: 6.2.2023 ***Termín odevzdání:*** 26.5.2023

Vedoucí práce: doc. Mgr. Karel Slavíček, Ph.D.

prof. Ing. Jiří Mišurec, CSc.

předseda rady studijního programu

Abstrakt

Tato bakalářská práce nastiňuje problematiku, jak je možné zabezpečit komunikaci v rámci IoT. Samotný mikrokontroler lze k tomuto účelu také využít nicméně nikdy nebude zajištěna taková bezpečnost, jako to bude při použití hardwarového kryptografického koprocesoru. Pro vygenerování klíčů a požadavku na certifikát je zvolen koprocesor ATECC608B, který v sobě uchovává potřebná „tajemství“, která se využívají pro zabezpečení komunikace. Jako demonstrace funkčnosti kryptografického koprocesoru, je připojení mikrokontroleru ESP32 k serveru AWS, kde veškerá navázání spojení se serverem probíhají pomocí kryptografického koprocesoru.

Klíčová slova

Kryptografický koprocesor, ATECC608, Internet věcí, TLS, ESP32, AWS, Kryptografie

Abstract

This bachelor thesis outlines the issue of how communication can be secured within the IoT. The microcontroller itself can also be used for this purpose, however, security will never be assured as it will be when using a hardware cryptographic co-processor. To generate the keys and certificate, the ATECC608B coprocessor is chosen which holds the necessary "secrets" used to secure the communication. As a demonstration of the cryptographic coprocessor functionality, the ESP32 microcontroller is connected to the AWS server where all the connections to the server are established using the cryptographic coprocessor.

Keywords

Cryptographic coprocessor, ATECC608, Internet of Things, TLS, ESP32, AWS, Cryptography

Bibliografická citace

ČESKÝ, Michal. *Kryptografické koprocesory pro zabezpečení komunikace IoT mikrokontrolerů.* Brno, 2023. Dostupné také z: https://www.vut.cz/studenti/zav-prace/detail/151046. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce Karel Slavíček.

Prohlášení autora o původnosti díla

Jméno a příjmení studenta:	*Michal Český*
VUT ID studenta:	*230789*
Typ práce:	*Bakalářská práce*
Akademický rok:	*2022/23*
Téma závěrečné práce:	*Kryptografické koprocesory pro zabezpečení komunikace IoT mikrokontrolerů*

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné 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 závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: 26. května 2023 ----------

podpis autora

Poděkování

Chtěl bych poděkovat za podnětné rady a připomínky při zpracování mé bakalářské práce vedoucímu této práce doc. Mgr. Karlu Slavíčkovi, Ph.D. Zejména za jeho trpělivé metodické a odborné vedení po celou dobu spolupráce na mé bakalářské práci. V neposlední řadě bych chtěl poděkovat své rodině a přátelům za podporu během celého mého studia.

V Brně dne: 26. května 2023 ----------

podpis autora

Obsah

SEZNAM OBRÁZKŮ

Úvod

Tato bakalářská práce se zabývá problematikou zabezpečení komunikace mikrokontrolerů v rámci IoT aplikací. Použitý mikrokontroler nemusí být dostatečně výkonný pro kryptografické funkce a nemusí poskytovat dostatečnou ochranu privátních klíčů pro asymetrickou kryptografii. Z tohoto důvodu je využit kryptografický koprocesor, který zajišťuje dostatečný výkon a správu bezpečnostních klíčů. Hlavní výhoda využívání kryptografického koprocesoru je, že privátní klíč nikdy neopustí zařízení.

Úvodní kapitola připomíná základní přehled kryptografie z pohledu IoT aplikací Druhá část této kapitoly se věnuje popisu jednotlivých algoritmů/kryptografií (RSA, EEC) a standardů (AES) pro šifrování dané komunikace. Závěr kapitoly se věnuje porovnání jednotlivých algoritmů, standardů a možností jejich využití v rámci IoT.

Ve druhé kapitole se seznamujeme s protokolem TLS a jeho užitím pro zabezpečení datové komunikace. Podrobně popsány a porovnávány jsou zde dvě nejpoužívanější verze a jejich takzvaný „handshake“.

Ve třetí kapitole je nastíněna problematika mikrokontrolerů. Zejména je zde popsána vývojová deska ESP32 použitá v praktické části bakalářské práce. Hlavní důraz je kladen na jednotlivé prvky a vlastnosti mikrokontroleru pro zabezpečení.

Ve čtvrté kapitole jsou popsány základní vlastnosti kryptografických koprocesorů a jejich uplatnění. Dále je zde uveden přehled koprocesorů na trhu. Důraz je kladen hlavně na kryptografické koprocesory od firmy Microchip a to zejména na typ CryptoAuthentication™ pro autentizaci v rámci IoT. V neposlední řadě jsou zde zmíněny základní vlastnosti koprocesoru typu ATECC608B, který je použit v praktické části bakalářské práce.

Poslední, pátá kapitola, je zaměřena na praktickou implementaci kryptografického koprocesoru. První část se věnuje volbě kryptografického koprocesoru, mikrokontroleru a vývojového prostředí. Druhá část se zabývá certifikáty, které lze pomocí koprocesoru vygenerovat. V závěrečné části je praktická ukázka, jak lze koprocesor implementovat do IoT.

1. ALGORITMY, STANDARDY A ZÁKLADNÍ TYPY KRYPTOGRAFIE

V běžném použití se šifrování dá rozdělit na dvě základní skupiny kryptografie, a to na symetrickou nebo asymetrickou kryptografii. Jednotlivé druhy kryptografie se navzájem liší algoritmem, který zajišťuje vypočítávání klíčů. Mezi nejznámější algoritmus patří RSA, kryptografie ECC a standart AES, ve kterém je použit algoritmus Rijndael. Klíče, které jsou vytvořeny je možno použít k šifrování, dešifrování a digitálnímu podpisu zpráv.

1.1 Symetrická a asymetrická kryptografie

Kryptografii lze rozdělit do dvou druhů podle typu šifrování. A to na šifru symetrickou a asymetrickou. Při využití symetrické šifry se pro šifrování a dešifrování dané zprávy používá stejný klíč. Uplatněním tohoto principu se docílí větší rychlosti spojení. Bezpečnostní klíče mají velikost nejméně 128 bitů, ale vyskytují se i klíče s velikostí 3072 bitů. Velikost klíče se odvíjí od použité šifry/kryptografie a od požadované míry zabezpečení. K nejpoužívanějším standardům symetrické kryptografie patří standard AES, který slouží pro šifrování dat. Pro AES platí že pokud dojde k vygenerování klíče opravdu náhodně a jeho velikost bude nejméně 256 bitů, tak je tento klíč definován jako prakticky neprolomitelný. Jeho výhodou je vysoká rychlost šifrování a následného dešifrování i pro velké množství dat a poměrně malá velikost klíče. Nevýhodou je však složitější distribuce daného klíče mezi dvěma uživateli. Tento způsob šifrování se také obtížně používá pro více jak dva uživatele, kteří potřebují dešifrovat stejnou zprávu. K tomuto případu by každá dvojice, která spolu komunikuje, musela mít vlastní symetrický klíč. Kdyby se totiž tohoto klíče zmocnil útočník, tak by mohl komunikovat s ostatními uživateli a vydávat se právě za původního vlastníka klíčů. [1; 2; 3]

Symetrické šifry lze rozdělit do dvou skupin. První, a nejpoužívanější, se nazývá bloková šifra. Tento typ šifry používají zejména dva známé algoritmy, a to Feistelova šifra, kterou využívá dnes již zastaralý standard DES, (ten již není považován za dostatečně bezpečný) a dále na SP síť neboli substituční permutační síť, kterou využívá například standart AES. [2; 3]

Bloková šifra rozdělí šifrovanou zprávu do bloků velikosti odpovídající délce klíče (například 128 bitů) a následně jej zašifruje do stejné velikosti jako původní text. Dále se symetrické šifry dělí na substituční a přestavovací. K nejznámější šifrám, které používají právě symetrickou blokovou šifru je AES nebo Twofish. [2; 3]

Dalším typem jsou šifry proudové. Proudové šifry fungují podobně jako šifry blokové, ale text rozdělí pouze po jednom znaku. Šifrovaný text, který je rozdělen po jednotlivých znacích a tyto znaky jsou následně zašifrovány pomocí operace XOR. Proudové šifry se používají na místech, kde není dostatečná kapacita pro výpočetní

paměť, nebo kdy začátek a konec přenosu je náhodně určován, což platí pro internetové hovory. Další možnost pro využití proudové šifry je tam kde lze předpokládat velké procento ztrátovosti. Největší nevýhodou proudových šifer je těžká synchronizace, pokud se nějaký bit ztratí nebo zpozdí. Když tato situace nastane, tak zapříčiní, že přijatou informaci není možné žádným způsobem dekódovat. Synchronizační chybu lze odstranit například pomocí synchronizačního bitu přidaného do informace. Šifry A5, Fish, RC4 nebo standard AES využívají právě funkce synchronizačního bitu. [2; 3]

Naopak u asymetrické šifry je dešifrovací klíč (privátní klíč) použit jiný než k šifrování (veřejný klíč). Mezi těmito dvěma šifrovacími klíči vzniká matematický vztah. Při využití asymetrické šifry se používá klíč s minimální délkou 224 bitů (pro ECC). Hlavní výhody šifrování asymetrickými šiframi je, že se při předávání klíčů více uživatelům nemusí dbát na zabezpečeni dané komunikace a že každý uživatel nemusí shromažďovat více klíčů ke komunikaci s více uživateli jako to je u symetrické kryptografie. Zásadní nevýhodou využití asymetrického klíče pro šifrování dat je delší doba pro šifrování a dešifrování daných dat. Proto se šifrování dat asymetrickým klíčem nahrazuje symetrickou kryptografií (standard AES). Nejznámější algoritmy, které využívají asymetrické šifrování jsou RSA, ECDH, ECDSA. [1]

1.2 Algoritmy a standardy

1.2.1 Algoritmus RSA

Algoritmus RSA (Rivest-Shamir-Adleman), který je pojmenován po svých tvůrcích, využívá asymetrické šifrování. Tento algoritmus se začal používat již v roce 1977 a používá se dodnes. Využívá vytvoření dvou klíčů, kde jeden klíč je veřejný a druhý je privátní. Mezi těmito klíči je matematický vztah. To znamená že veřejný klíč může být sdílen napříč sítí, ale privátní klíč musí zůstat v utajení. Pro šifrování zpráv je možné použít jakýkoliv z těchto dvou klíčů, ale pro dešifrování zprávy se musí použít opačný klíč než ten, kterým byla zpráva zašifrována. [4]

Bezpečnost tohoto algoritmu spoléhá na součin dvou velkých prvočísel, která jsou náhodně generována. Následně je prvočíslo pomocí Rabin-Millerova pravděpodobnostního testu zkontrolováno, zda se opravdu jedná o prvočíslo. Tento součin je následně pomocí operace modulo, zprávy a klíče výsledkem zašifrované nebo dešifrované zprávy. Pro dešifrování a šifrování se používají stejná dvě prvočísla. Proto je tento proces možné provádět obousměrně. Konkrétně se jedná o nesnadnou faktorizaci těchto velkých čísel. Faktorizací se rozumí proces, při kterém se z daného velkého čísla stanovují právě dvě prvočísla, která se násobila. Tento proces je velice zdlouhavý, a to je důvodem velmi častého využívání tohoto algoritmu. I když je tento algoritmus velmi používaný, tak obsahuje několik záporných vlastností. Jednak značně obtížnou faktorizaci čísel, což by v budoucnu mohly vyřešit výkonnější počítače

a jednak tvoření extrémně dlouhých klíčů, což zvyšuje potřebný výkon pro zdlouhavé matematické operace. [4; 5]

V dnešní době se běžně používají klíče o velkosti 2048 bitů. Je však již potvrzeno, že by klíč s velikostí 2048 bitů mohl být prolomen. Podle organizace NÚKIB (Národní úřad pro kybernetickou a informační bezpečnost) se doporučuje, aby klíče s délkou 2048 bitů byly nahrazeny silnějšími klíči o velikostí 3072 bitů, a to nejpozději do roku 2023. Tento fakt platí jak pro digitální podpis, procesy dohod, a tak i pro šifrování zpráv. [6]

Tento algoritmus se nejvíce používá pro digitální podpis (který se využívá při autentizaci) a pro šifrování zpráv. Při autentizaci se zjišťuje, jestli veřejný klíč patří ke správnému soukromému klíči a následně jestli držitel dvojice klíčů (veřejného a privátního) je skutečným vlastníkem nebo jestli se za něj jenom vydává. Další použití je u kryptografických protokolů SSL a TLS pro zabezpečení komunikace po síti mezi dvěma nebo více stranami. [4; 5]

1.2.2 Algoritmus ECC

Kryptografie pomocí eliptických křivek, zkráceně ECC pochází z anglického názvu Elliptical curve cryptography. ECC byla objevena již v roce 1985 Neaolem Koblitzem. ECC pracuje na bázi šifrování veřejným klíčem pomocí teorie eliptických křivek. Algoritmus vznikl jako alternativa již používaného algoritmu RSA. [7]

Celý algoritmus funguje na bázi takzvaného „trapdoor“, což znamená, že použitím jakékoli hodnoty a nějaké funkce lze lehce dosáhnout výsledku, ale naopak z výsledku dosáhnout původní hodnoty je velmi složité. Nejčastěji se ECC využívá pro digitální podpis na blockchainu (například pro kryptoměnu Bitcoin, v technologii DLT = Distributed Ledger Technology apod.) nebo pro zachování anonymity v prostředí Tor. [7; 8]

Zkratka ECDH neboli také Elliptical curve Diffie Hellman je založená na kombinaci dvou názvů, které znamenají využití kryptografie ECC a protokolu Diffie Hellman zkráceně DH. Protokol ECDH využívá asymetrickou kryptografii. Slouží pro výměnu klíčů mezi dvěma stranami. Celý algoritmus využívá vlastnosti multiplikativní grupy na modulu prvočísla. Značné výhody jsou větší rychlost, veliká podporovatelnost napříč platformami a malá velikost klíče. Značnou nevýhodou je, že kombinace ECC a DH tohoto protokolu nepodporuje autentizaci. S protokolem ECDH je možné se setkat v bezpečnostním protokolu TLS. [9]

Pro ověření komunikačních stran slouží algoritmus ECDSA (Elliptical curve Digital Signature Algorithm), který využívá algoritmus DSA neboli Digital Signature Algorithm postavený na eliptických křivkách. Algoritmus pracuje na procesu multiplikativní grupy stejně jako protokol DH. Privátní klíč slouží k podpisu hashe a veřejný klíč k ověření daného podpisu. Hash je vytvořen pomocí hashovací funkce,

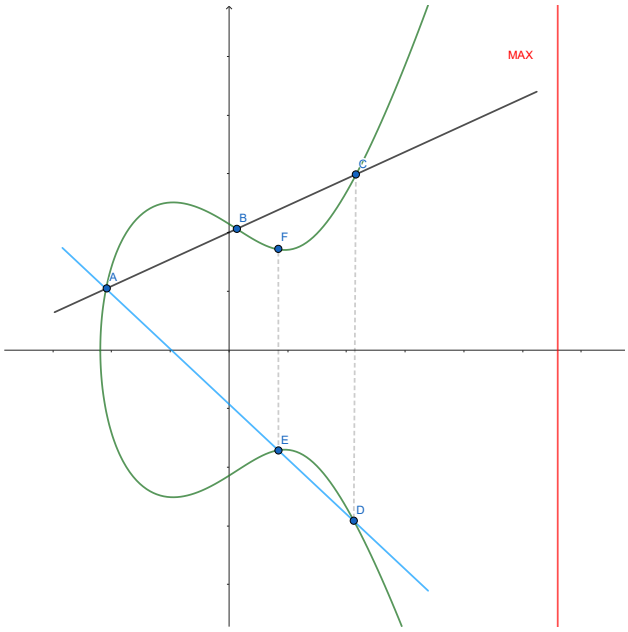
která převede certifikát na pevně danou velikost. Algoritmus ECDSA má stejné výhody jako protokol ECDH a taktéž se využívá v bezpečnostním protokolu TLS. [9]

Podle institutu NSA (národní bezpečnostní agentura) je postačující pouze použití klíče o velikosti 384 bitů pro přísně tajné utajení [8].

Vytváření klíče probíhá pomocí předem určeného vztahu,

$$y^2 = x^3 + ax + b\,. \qquad (1.1)$$

Následný graf 1.1 podle vztahu (1.1) po zvolení parametrů a = -2,7 a b = 4,6 bude vypadat následovně. [7]



Obrázek 1.1 Eliptická křivka využívaná ECC algoritmem [7]

Jak je vidět z obrázku 1.1, tak eliptická křivka je symetrická podél osy x což znamená, že pokaždé, když bude přidána další přímku, tak protne tuto křivku maximálně 3krát.

Pro šifrování si zvolíme počáteční bod A, ze kterého se budou odvíjet další přímky. Pomocí náhodné lineární matematické funkce je vedena z bodu A přímka (černá) přes bod B a takto je dosaženo výsledného bodu C. Bod D lze získat pomocí vlastnosti, že křivka je symetrická podél osy x. To znamená že bod D se bude nacházet svisle pod bodem C na eliptické křivce. Od počátečního bodu A bude vedena další přímka (modrá) do bodu D. Tímto postupem bude získán bod E. Tento postup se může opakovat po nějakou určitou dobu, dokud nebude dosažena výsledná hodnota. Při takto zvoleném postupu je možné, že body se budou nacházet až za maximální hodnotou (červená přímka). Pokud se tak stane, tak se vzdálenost od maximální hodnoty k danému bodu přesune zpět na počáteční bod A. Velikost klíče bude odpovídat velikosti maximální hodnoty. Jestliže klíč bude mít velikost 256 bitů bude maximální hodnota 256. [8]

Čím je delší maximální hodnota, tím bude algoritmus počítat s více čísly a bude bezpečnější. Z výše popsaných operací se získá privátní klíč, který bude roven počtu přímek pro získání bodů. Z pohledu bezpečnosti je velice těžké zjistit právě množství opakování pro získání konečného bodu. [8]

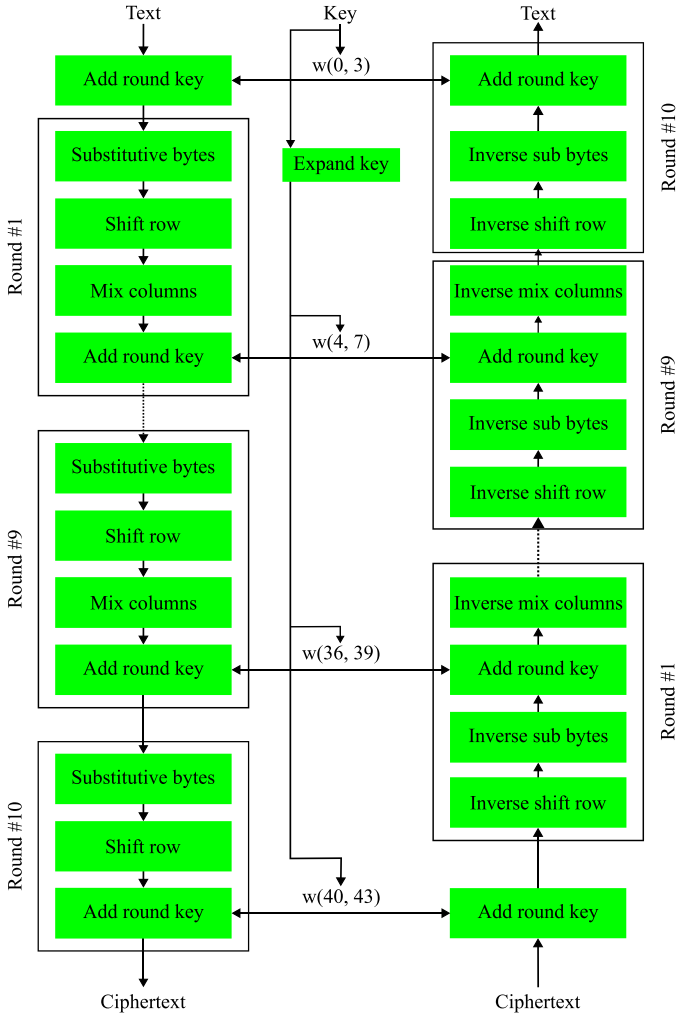
1.2.3 Standart AES

Zkratka AES značí Advanced Encryption Standart. Standart AES využívá symetrickou blokovou kryptografii. Vznikl jako náhrada svého předchůdce DES a byl vyvinut Institutem pro národní standardy a technologii, pro použití ve vládním sektoru Spojených států amerických. V praxi se lze setkat z různými velikostmi klíče, které se značí pomlčkou a zvolenou velikostí klíče, například AES-128. Pro bezpečnostní prověrku přísně tajné je ve Spojených státech amerických nutné použít minimální hodnotu klíče 192 bitů. Na rozdíl od USA v České republice jsou podle NÚKIB považovány klíče za bezpečné již od 128 bitů. Je však doporučována velikost 192 bitů. Dále je doporučeno přejít z algoritmu DES právě na algoritmus AES, a to do roku 2023. V praxi se u výpočetní techniky s dostatečným výkonem běžně používají klíče velikosti 256 bitů (například IPSEC tunely), u IoT aplikací je naopak obvyklá délka klíče pohybuje okolo 128 bitů. [6; 10]

Zpráva, kterou chceme zašifrovat pomocí AES-128, se rozdělí do bloků, kde každý blok bude mít velikost 128 bitů. Velikost bloku je vždy 128 bitů, přičemž nezáleží na velikosti klíče. Každý blok si lze představit jako tabulku o velikosti čtyř řad a čtyř sloupců, kde každá buňka tabulky má reprezentovat velikost 8 bitů. Dále je nutné vytvořit privátní klíč, který by se měl skládat z náhodně vygenerovaných znaků, kde velikost privátního klíče záleží na stupni zabezpečení. V tomto případě bude jeden blok o velikosti 128 bitů, to znamená že blok bude vypadat stejně jako jeden blok rozdělené zprávy. [11; 12]

Na bloky zprávy je použita operace XOR blokem původního privátního klíče (zvaného key 0) a tím je získaný upravený blok zprávy. Dále nastává proces repetice (round) čtyř funkcí. První operace Substitute Bytes je, že jednotlivé bloky zprávy jsou pomocí substituční tabulky (nazývané AES S-Box) zašifrovány. Substituční tabulka je navržena tak, že žádná hodnota v tabulce není stejná nebo inverzní původní hodnotě. Jako druhá operace je Shift Rows, kde dojde k posunu řádku v každém bloku o určitý počet bitů. První řádek tabulky zůstává, druhý se posune o jeden bit doleva, třetí o dva bity doleva a poslední o tři bity doleva. Třetí operací je operace Mix Columns, která prohazuje jednotlivé buňky bloku v jednom sloupci. Toto vzniká pomocí konstantní matice zvané Rijndael matrix, kterou se vynásobí blok zprávy. Nejedná se o klasické násobení, protože se zpráva nachází v takzvaném Finite field, takže operace sčítání je nahrazena operací XOR a operace násobení je nahrazena operací modulo polynom. Jako poslední se použije operace Add Round Key, která pomocí operace XOR zašifruje již upravené bloky zprávy. Celá operace je řízena speciální funkcí nazývanou

Key Derivation Function. Při repeticích není použit stále stejný privátní klíč. Z tohoto důvodu je tento standart velice bezpečný. Postup pro vytváření nového privátního klíče je následující. Vezme se původní privátní klíč a vybere se z něj poslední sloupec matice. Tento sloupec je posunut o jeden bit dolů a pak pomocí určitého sloupce substituční tabulky zašifrován. Dále se vybere požadovaný sloupec v konstantní tabulce Rcon. Výběr sloupce tabulky Rcon pro zašifrování privátního klíče se určuje podle toho, v jakém bodě se šifrování nachází. Následně stačí vzít první sloupec privátního klíče, upravený poslední sloupce privátního klíče a požadovaný sloupec tabulky Rcon. Pomocí operace XOR se dostane první sloupec nového privátního klíče. Další sloupce pak vznikají pomocí operace XOR a výběrem dalšího sloupce původního privátního klíče a posledního sloupce nově vytvořeného privátního klíče. Všechny tyto operace jsou navzájem inverzní a tím pádem je lze snadno dešifrovat. Dešifrování probíhá v opačném pořadí než šifrování při použití od posledního upraveného privátního klíče k úplně původnímu klíči. Celý postup je znázorněn na obrázku 1.2. [11; 12]



Obrázek 1.2 Zobrazení průběhu standardu AES pro klíč o velikosti 128 bitů [13]

Standart AES je open source, tím pádem je zde velký potenciál v bezpečnosti. Další výhodou je snadná implementace do různých softwarových i hardwarových zařízení. Předností je malá velikost klíče při zajištění vysoké bezpečnost nebo vysoká rychlost šifrování a dešifrování zpráv na počítačích. [10]

Tento postup čtyř funkcí se opakuje podle toho, jak dlouhý je klíč, u 128bitového klíče se postup opakuje 10krát, u 192bitového klíče 12krát a u 256bitového klíče 14krát. Operace Mix Columns se provádí vždy o jednou méně, než je celkový počet repetic. Samotný standart AES využívá operační módy. Nejvíce se využívá právě pět nejznámějších a to ECB, CBC, CFB, IFB, CTR, XTS. [11]

Celková bezpečnost standardu AES je založena na upravování a šifrování jednotlivých buněk v bloku, což je pomocí techniky brute-force zatím téměř nereálné. Největší zranitelností standardu AES je útok na vedlejší kanál (side-chanel). Tento útok se zaměřuje přímo na hardware a následný únik informací o průběhu šifrování. [10]

1.2.4 Shrnutí ECC, RSA a AES

RSA je asymetrický algoritmus, který je možné použít k šifrování zpráv a k digitálnímu podpisu dat. K tomuto algoritmu lze přirovnat kombinaci protokolu ECDH a algoritmus ECDSA, které jsou také asymetrické. Tato kombinace se hojně používá k ověření dvou stran, navázaného zabezpečeného spojení a šifrování dat stejně jako algoritmus RSA.

V rámci bezpečnosti lze pak uvažovat že klíč RSA o velikosti 7680 bitů je stejně bezpečný jako klíč ECDH/ECDSA s velikostí pouze 384 bitů, což velmi snižuje potřebný výkon pro výpočet klíčů [6; 8]. Tento fakt může způsobovat problém, jak dosáhnout dostatečného zabezpečení RSA u zařízení s menším výkonem [7]. Další výhodou využití kombinace ECDH/ECDSA je, že soukromý klíč a veřejný klíč jsou od sebe snadno rozeznatelné. Na rozdíl od dvou celočíselných, jak je to u RSA. Nelze také opomenout větší rychlost tvoření klíčů u ECDH/ECDSA.

Standart AES pracuje na symetrické kryptografii. Standart se využívá pouze pro šifrování nikoli pro autentizaci jako to bylo u výše zmíněných algoritmů/protokolů. V praxi je možné se setkat s kombinací ECDH/ECDSA, která zajišťuje výměnu klíčů a autentizaci pomocí digitálního podpisu. Důvodem nevyužívání ECDH pro šifrování dat pomocí asymetrické kryptografie je, že symetrická kryptografie zabírá mnohem méně času pro samotné šifrování a dešifrování dat. [14]

2. PROTOKOL TLS

Pro běžnou zabezpečenou komunikaci se v prostředí internetu využívá protokol TLS. Aktuálně jsou dvě využívané verze: 1.2, 1.3. TLS může využít algoritmus RSA nebo ECDHA/ECDSA pro autentizaci klienta a serveru a následné zabezpečené komunikace pomocí šifrování.

2.1 Historie protokolu TLS

TLS 1.0 což je první verze protokolu byla vydána již v roce 1999. Protokol byl původně zamýšlen jako nová verze již používaného protokolu SSL (Security Socket Layers), který vznikl 20 let poté, co byl ARPANET (první počítačová síť na světě) spuštěn do provozu. SSL protokol byl jeden z prvních protokolů, které měly zajišťovat zašifrování dat mezi serverem a klientem. Byl založen společností Netscape. Protokol TLS byl původně další verzí 3.1 známého protokolu SSL, ale protože jej již nevyvíjela společnost Netscape, tak byl tento protokol přejmenován na TLS. [15; 16]

2.2 Popis protokolu TLS

Protokol TLS se hojně používá pro přenos dat ve webových aplikacích i pro zabezpečení dalších služeb a protokolů. [15]

Je možné jej také využít s transportním protokolem UDP (dnes možno i s TCP) pro VPN (Virtual Private Network) sítě, například u protokolu OpenVPN. [16]

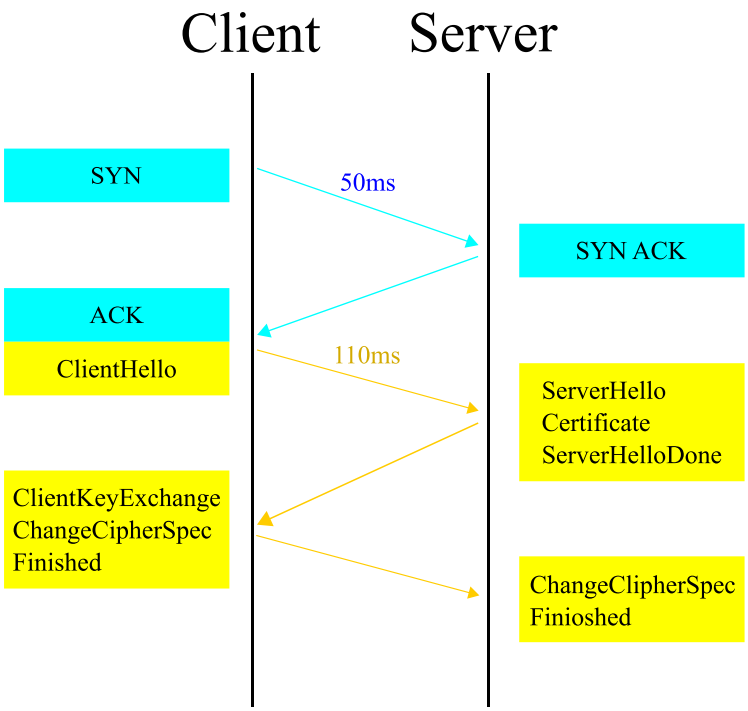
Protokol TLS využívá právě asymetrickou a symetrickou kryptografii dohromady (u verze 1.3 pouze asymetrickou) pro zabezpečení, generování a následné předání klíče druhé straně. Po zakončení dané relace se klíč odstraní. [16]

Celkem se protokol TLS dělí do třech částí. První částí je šifrování. To zajišťuje zabezpečení například proti odposlechům. Druhá část je autentizace. Ta má za úkol ověřit, že dvě strany, které spolu komunikují jsou ty správné, nikoli ty, které by se za ně případně mohly vydávat. Poslední částí je integrita, která kontroluje, zda odeslaná data nebyla upravena. [15]

TSL protokol využívá takzvané šifrovací sady, které jsou nedílnou součástí každé komunikace. Pod pojmem šifrovací sada se rozumí různé šifrovací algoritmy, které se následně vybírají při takzvanému „handshake“. [15]

Standartně, v prostředí internetu pro komunikaci mezi stanicemi, je využíván protokol TLS za použití RSA nebo kombinace protokolu ECDH a algoritmu ECDSA. V této práci se bude protokol TLS využívat pro správu klíčů, kde se bude používat protokol ECDH pro generování klíčů a algoritmus ECDSA pro vygenerování digitálního podpisu.

2.3 Průběh spojení TLS protokolu 1.2

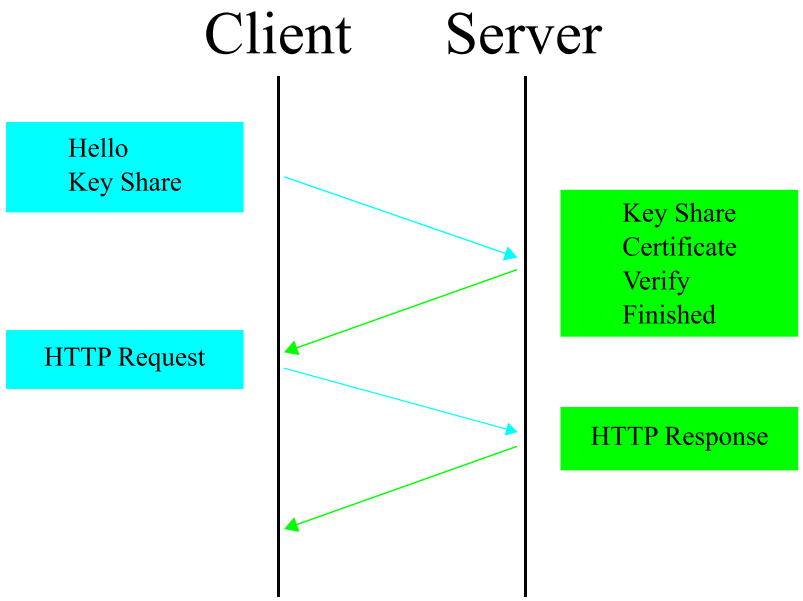


Obrázek 2.1 Navázání spojení TLS protokolu verze 1.2 [17]

Na obrázku 2.1 je znázorněn takzvaný „handshake“ (žlutá barva), který musí proběhnout vždy před zahájením relace. Zpráva SYN značí, že začala nová sekvence číslování bajtů a následná zpráva ACK od serveru pro potvrzení přijetí u transportního protokolu TCP. „ClientHallo“ navazuje komunikaci mezi serverem a klientem a posílá základní informace: verzi protokolu TLS, použitou sadu pro šifrování (list algoritmů pro šifrování) a náhodně vygenerovanou řadu bajtů klientem „Client random“. Server odešle zprávu „ServerHello“, kde ověřuje nastavenou verzi protokolu TLS, vybraný algoritmus pro šifrování, identifikační číslo přenosu (ID) a certifikát daného webu (pro webové stránky signalizace zeleného zámku). Zpráva „ChangeClipherSprec“ vytváří náhodně vygenerovanou řadu bajtů serverem (veřejný klíč). Dále klient provede autentizaci certifikátu, který obdržel od serveru ve zprávě „Certificate“ a odešle serveru další vygenerovanou řadu bajtů (zpráva ClietnKeyExchange) „premaster secret“ (tajný klíč), který je použitím matematických operací zašifrován pomocí veřejného klíče přijatého od serveru. Bajtovou řadu lze dešifrovat pouze pomocí správného privátního klíče, který vlastní odpovídající server. Společně s bajtovou řadou je klientem odeslaná zpráva „Finish“. Pokud se následné odeslané řady bajtů shodují, tak server odešle zprávu „Finished“. Následně server a klient mohou začít spolu šifrovaně komunikovat pouze pomocí tejného klíče. Tím pádem následná komunikace pracuje na bázi symetrické šifry. [17; 18]

2.4 Průběh spojení TLS protokolu 1.3

Verze 1.3 je nástupcem verze 1.2 a přináší jednodušší „handshake“. U verze 1.2 musel být jakýkoliv „handshake“ zašifrovaný a následně dešifrovaný což vyžadovalo více odeslaných paketů pro „handshake“. Touto změnou došlo ke zkrácení komunikace a rychlejšího navázání spojení. [17; 19]



Obrázek 2.2 Navázání spojení TLS protokolu verze 1.3 [19]

Jak je patrné z obrázku 2.2, klient posílá v prvním paketu serveru verzi protokolu (v tomto případě 1.3), „Client random“ (vygenerovanou řadu bitů) a následně předem zvolenou sadu pro šifrování. Pakliže server dostane od klienta „Client random“, tak již může vytvořit tajný klíč. Ve stejném paketu pak odešle zprávu „Finished“ pro správné nastavení TLS protokolu mezi serverem a klientem. V tomto případě klient požaduje od serveru „HTTP Request“ a server odpovídá „HTTP Response“. [17; 19]

2.5 Porovnání verzí TLS

V současné době disponujeme celkem čtyřmi verzemi a to TLS 1.0, 1.1, 1.2, 1.3. Dvě první verze jsou již považovány za nebezpečné a z tohoto důvodu se již nepoužívají. V současnosti se již můžeme setkat pouze s verzemi 1.2 a 1.3. [20]

V roce 2018 byla spuštěna verze 1.3, která obsahuje spoustu nových zdokonalených prvků. Jednou z jejích prvních výhod je rychlejší a zároveň bezpečnější „hasndake“ a to pomocí redukovaní paketů. Z dříve používaných celkových až 7 paketů, jejichž využití trvalo nějakých 300ms, až na pouze 3 pakety s dobou využití okolo 200ms. Mezi další zdokonalené prvky verze 1.3 patří jednodušší a zároveň i bezpečnější sady pro následné šifrování. Této vlastnosti bylo dosaženo využíváním pouze těch šifrovacích algoritmů,

které doposud nebyly prolomeny nebo byly bez žádné nalezené chyby. Pomocí těchto sad bylo odstraněno riziko pro opakované sjednávání již určených požadavků pro připojení. Poslední hlavní výhodou verze 1.3 je, že se zde přestává používat statický klíč, čímž je možné již v prvním paketu „Hallo“ odeslat zároveň i tento klíč a není potřeba přijímat další paket od serveru pro „handshake“. [17; 19]

3. MIKROKONTROLERY

3.1 Co je mikrokontroler

Pod pojmem mikrokontroler se dají představit zařízení, která lze připodobnit k miniaturizovanému počítači, který bývá ve velmi malém provedením. Jako účelem je provádět malé programovací operace.

Mikrokontroler se většinou skládá z mikroprocesoru, pamětí, I/O vstupů a výstupů a mnoha dalších prvků. Dále se liší počtem IO pinů, externími nebo interními moduly Wi-Fi nebo Bluetooth. Mikrokontroler ESP32 obsahuje procesor Tensilica s vlastní architekturou. Jiné typy mikrokontrolerů (například Raspberry) využívají architekturu ARM. Mikrokontrolery s touto architekturou bývají často větší, výkonnější ale zároveň jejich cena je mnohem vyšší (okolo 50€). [21; 22; 23]

Typů mikrokontrolerů je velmi mnoho. Liší se v programovacím jazyce, kterým lze mikrokontroler naprogramovat. Jako nejběžnější programovací jazyk se využívá jazyk C/C++, ale není výjimkou i Python, microPython, nebo JavaScript. [21]

Mezi nejznámější výrobce vývojových desek, které obsahují právě mikrokontroler, patří Arduion, Raspberry a Espressif zkráceně (ESP).

3.2 Popis funkce mikrokontroleru

Každý mikrokontroler obsahuje tyto základní prvky.

Procesor, (CPU), který zpracovává a následně vyhodnocuje předem naprogramované operace, které mohou být dále předány dalším modulům v dané sestavě. Procesory jsou zpravidla 8, 16, 32 nebo až 64bitové. Další prvek, který mikrokontroler obsahuje, je paměť. Jedná se většinou o dva typy pamětí. A to ROM, která uchovává data i při přerušení napájení, a paměť RAM, která se vymaže pokaždé, když mikrokontroler ztratí napájení. Paměť ROM se dále dělí na EPROM, tedy paměť, která slouží pouze pro čtení a nedá se vymazat. Dále na EEPROM, kterou je možné elektronicky vymazat. K výměně dat mezi procesorem a paměťmi se využívají dvě základní architektury. Jako první architektura je architektura Harvard, která využívá oddělitelnosti datové sběrnice a dané instrukce. Na rozdíl od architektury Von Neumann, která tuto oddělitelnost nevyužívá, a proto se data a instrukce odesílají po jedné sběrnici. Architektura Von Neumann se využívá u větších a výkonnějších zařízení (počítače) k zachování dat o instrukcích procesoru. Tento typ architektury mikrokontrolery nevyužívají. Paměť RAM se využívá jako dočasné úložiště pro provádění určité funkce. I/O periferie pak zprostředkovává komunikaci mezi dalšími přídavnými komponenty a procesorem mikrokontroleru. Může to být sériový port pro komunikaci a programování mikrokontroleru přes počítač. [21; 22]

3.3 ESP32

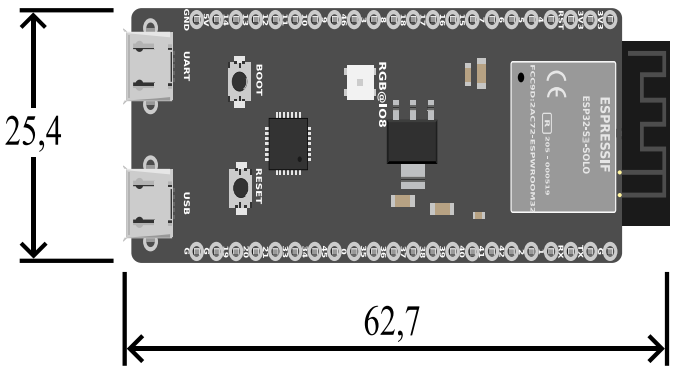
ESP32 je mikrokontroler. Jednotlivé vývojové desky od výrobce Espressive obsahují mikrokontroler ESP32 nebo jeho předchůdce ESP8266. Od svého předchůdce se liší jednak větším výkonem, což se hodí na kryptografické zabezpečení datové komunikace, a jednak obsahuje modul pro bezdrátové rozhraní WiFi a Bluetooth. ESP32 disponuje dvoujádrovým 32bitovým Tensilica Xtensa LX6 procesorem s maximální frekvencí 240MHz, který řídí všechny operace vývojové desky. Vývojová deska je velice málo energeticky náročná. Z tohoto plyne, že je možné ji napájet pomocí klasického USB nebo pomocí baterií. Bateriové napájení je velice výhodné k IoT aplikacím. [24; 25]

Vývojovou desku ESP32 lze programovat pomocí více programovacích prostředí. Lze využívat například VS Code s komponentou ESP-IDF nebo klasické Arduino IDE.

Vývojová deska ESP32 byla zvolena, poněvadž disponuje větším výkonem procesoru, větší velikostí paměti, vyšší podporovatelností vývojových prostředí a programovacích jazyků než jiné mikrokontrolery ve stejné cenové relaci. Velkou roly hraje i vlastní zkušenost s touto řadou vývojových desek.

3.3.1 Přehled vývojových desek na bázi ESP32

Na trhu se objevují vývojové desky ESP32 od různých výrobců. Hlavním výrobcem je firma Espressif, ale existuje i mnoho dalších značek. Nejprodávanější řadou od firmy Espressif jsou modely označeny symbolem S3, které se velice hodí pro použití v IoT.



Obrázek 3.1 ESP32-S3 rozložení pinů [26]

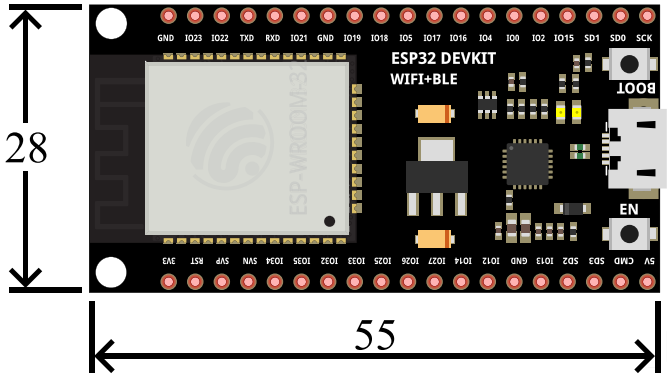
Vývojová deska disponuje mikrokontrolerem ESP32-S3-WROOM-1/2, ve kterém jsou zabudovány síťové funkce jako WiFi nebo Bluetooth LE. Od mikrokontroleru ESP32 se liší především větším výkonem, větší velikostí pamětí nebo vestavěným USB portem, který podporuje napájení, USB-OTG nebo komunikační protokol USB 1.1. Dále vývojová deska disponuje 45 GPIO piny, 3 páry UART pinů nebo 2 páry komunikační sběrnice I^2C. V rámci bezpečnosti disponuje pokročilými funkcemi šifrování jako AES-256, RSA, ECDSA, RNG. [27]

Další možností je zvolit si variantu s označením C. Novinkou na trhu je verze ESP-32-C6. Největší výhodnou tohoto modelu oproti ESP-32-S3 je, že propojuje verzi WiFi 6 a Bluetooth LE 5.3. Pro konektivitu v rámci IoT aplikací disponuje protokoly Thread nebo ZigBee. V rámci bezpečnosti nesmějí chybět šifrovací standardy jako AES-256, RSA-3072 nebo ECDSA. Hlavním rozdílem je, že typ C má méně rozhraní GPIO (celkem 30). Další výhodou může být, že není tolik energeticky náročný. Další rozdíl může být v jeho umístění. Řada C3 nebo nejnovější C6 ve verzi SoC je vyrobena pro výměnu starých čipů u ESP8266.

Vývojová řada typu H2 disponuje možností využívat ZigBee protokol, což se nejvíce využije u chytrých domácností a u komunikací s chytrými asistenty.

ESP se vyrábí ve třech druzích, a to jako vývojová deska, jako modul nebo SoC. Rozdíl mezi nimi je ten, že vývojová deska obsahuje vlastní modul, a jsou v ní přidány věci navíc jako USB pro komunikaci s PC, tlačítka, LED diody apod. Možné je koupit různé kombinace vývojových desek, třeba s integrovaným displejem nebo s externí kamerou. [28; 29]

3.3.2 ESP-WROOM-32 2.4GHz Dual-Mode WiFi+Bluetooth rev.1, CP2102



Obrázek 3.2 ESP-WROOM-32 rozložení pinů [30]

Jak již bylo zmíněno, ESP32 disponuje modulem Wi-Fi, který komunikuje pouze na pásmu 2,4GH a modulem pro Bluetooth v4.2. U většiny typů ESP je možné využívat novou technologií používanou na větší vzdálenosti zvanou Bluetooth Low Energy. [31]

Toto ESP má dvoujádrový koprocesor s maximální frekvencí 240MHz s velmi nízkou spotřebou, kterou lze dosáhnout pomocí uvedení procesoru do hlubokého spánku. Každé jádro procesoru má jinou funkci. První jádro s názvem Protocol CPU má za úkol všechnu správu periferií jako například Wi-Fi. Druhé jádro se nazývá Application CPU, a slouží ke zpracování a výpočtu aplikačního kódu. Tento koprocesor je velmi efektivní při provádění matematických operací s velkými čísly, což je potřeba pro rychlé vynásobení a generování dvou prvočísel pro RSA algoritmus. [31]

Na rozdíl od jiných vývojových desek, ESP32 disponuje pěti typy pamětí. A to pamětí ROM o velikosti 448 kB, kterou nelze přepisovat. Obsahuje základní nastavení pro moduly a zavaděč pro získání a možné spuštění naprogramovaného kódu z externí paměti. Další pamětí je SRAM s velikostí 520 kB pro ukládání kódu a dat pro procesor. Výhodou je velmi rychlé spojení mezi těmito prvky. Další pamětí je RTC SRAM, což je paměť, která je využívána v hlubokém spánku koprocesorem. Čtvrtou pamětí je Flash embedded pro ukládání aplikačního kódu. Další rozdíl od jiných vývojových desek je, že obsahuje možnost přidat externí paměť až do velikosti 16 MB, které se využívá pro zlepšení výkonu nebo když není dostatek paměti v dané vývojové desce. [24; 25; 32]

Významným blokem jsou pak akcelerátory. Ty zabezpečují aplikační kód pomocí šifrování například pomocí AES standardu, SHA-2, ECCDH a RSA. [24; 31]

A nakonec jsou to periferie. Mezi nejdůležitější patří piny, které umožňují přidání externích modulů jak pro vstupní prvky, tak zároveň i pro výstupní prvky. Pro IoT to může být modul pro Lora-WAN, čidlo okolního prostředí nebo kryptografický čip. Celkem má tento modul 34 GPIO vstupů. Dále se zde nacházejí piny s digitálním analogovým převodníkem, a to celkem 18krát a s analogově digitálním převodníkem o počtu dvou, dotyková tlačítka, která fungují na bázi měření odchylky odporu, periferie pro komunikaci s SD nebo microSD kartami, LED PWM, hallowy senzory nebo třeba tři (páry) piny UART, I^2C s dvěma piny například pro komunikaci s kryptografickým čipem. [24; 25; 31]

Pro připojení kryptografického čipu lze použít dvě sběrnice: UART a I^2C, které se na vývojové desce nachází vícekrát. I^2C sběrnice využívá dva obousměrné vodiče. Ten první pro převod dat a druhý pro synchronizaci hodinového signálu. Celá komunikace funguje na bázi master-slave. [31; 33]

Jako další výhodnou možností je využití odděleného oscilátoru s maximální frekvencí 32 kHz [31].

Pro připojení k bezdrátové síti je mikrokontroler vybaven modulem Wi-Fi 2,4GHz o maximální rychlosti 150 Mbps. Nebo je možné taktéž využít Bluetooth v4.2. [31]

4. KRYPTOGRAFICKÉ KOPROCESORY

4.1 Co je to kryptografický koprocesor

Kryptografický koprocesor se využívá pro dosažení většího výkonu zařízení. Podle typu koprocesoru se může jednat o zpracování řetězců, zpracování signálů nebo v tomto případě o šifrování zpráv a pro dosažení zabezpečené komunikace. Při použití TLS protokolu, kde se využívá algoritmu RSA nebo kombinace ECDH/ECDSA a šifrování pomocí velkých prvočísel, je zapotřebí použít právě kryptografický koprocesor pro odlehčení výkonu a větší míry zabezpečení vývojové desky.

4.2 Základní vlastnosti kryptografického koprocesoru

Na trhu se objevují různé druhy kryptografických čipů. Hlavní rozdíl mezi nimi je způsob a použití algoritmů pro šifrování. Mnohé z nich nemají pouze jeden typ šifrovacího algoritmu například RSA, ale lze si vybrat i z jiných dostupných. Mezi nejpoužívanější algoritmy patří algoritmus ECDH, SHA256 pro hashovací funkce nebo standart AES. Algoritmus RSA se příliš nepoužívá, protože je zapotřebí vygenerovat dlouhý klíč nebo pro větší výkonovou náročnost. Například chip ATEEC608B obsahuje tři typy algoritmů nebo standardů. [34]

Další důležitou vlastností je způsob, jakým koprocesor bude komunikovat s vývojovou deskou. Většina koprocesorů využívá připojení pomocí jednoho páru vodičů připojeného na pin s I^2C sběrnicí dané vývojové desky. Není výjimkou připojovat koprocesor pomocí sběrnice UART nebo I^2C. [34]

Postup pro samotné osazování čipu je také důležitým faktorem. Existují řady typů pouzder čipů, například SOIC, UDFN nebo BGA. Nejběžnější je takzvaný SOIC. Pod tímto názvem si lze představit malé pouzdro, ve kterém je umístěn samotný čip. Toto pouzdro umožňuje lehčí vlastnoruční pájení na tištěný spoj nebo možné uchycení na nepájivé pole. [35]

Druhým typem je standart UDFN. Je to typ montáže přímo na tištěný spoj, kde čip nemá žádné fyzické vývody, ale pouze malou vodivou destičku pro uchycení na tištěný spoj. UDFN je jedna z možných variant SMD montáže. U typu osazení každého čipu se vyskytuje pomlčka s číslem, která udává počet vývodů nebo malých destiček daného čipu. [36]

Pro kryptografické čipy mohou existovat celkem tři základní bezpečnostní zranitelnosti. První je chyba v designu a následné prolomení pomocí side-chanel. Druhá je špatně vyškolený personál, který využívá hardware. Jako třetí, ale velmi nepravděpodobné, je možnost prolomení algoritmu. K zamezení této chyby je možné použití algoritmů, které ještě nebyly prolomeny nebo zvolení větší délky klíčů. [37]

4.3 Přehled kryptografických koprocesorů na trhu

Na trhu se objevují kryptografické koprocesory od různých výrobců. K nejznámějším značkám patří Microchip, Infineone, Maxim Integrated a NXP.

Od firmy Microchip existují koprocesory pro různé způsoby využití. Pro zabezpečení a autentizaci slouží typ AT (CryptoAuthentication™). Tato skupina se dále dělí podle šifrovacích algoritmů, kterými koprocesor disponuje. ATECC, který využívá algoritmus EEC s kombinací Diffie Hellman pro ustanovení klíčů, DSA pro digitální podpis, možnost vytvoření hashovací funkce SHA pro reprezentaci daného textu v předem definované velikosti, nebo podle řady může obsahovat i standart AES pro samotné šifrování zpráv. Nejpoužívanější koprocesory této skupiny jsou ATECC508A a ATECC608B. ATECC608B je nejnovější verzí typu ATECC a disponuje více funkcemi než jeho předchůdce ATECC508. Dále je k dispozici ATAES, který využívá pouze standart AES pro šifrování zpráv ale pouze s maximálním klíčem o velikosti 128 bitů. Koprocesor je schopen využívat módy CCM a ECB pro standart AES. Při použití algoritmu AES je kladen velký důraz na velikost paměti, kterou koprocesor disponuje. Paměť dosahuje 32 kB. Do této skupiny můžeme také zařadit koprocesory označené ATSHA pro hashovací funkce. Z nich se nejvíce využívá typ ATSHA204A. Posledním typem autentizační skupiny je ATxxSCxx. Tento typ koprocesoru využívá algoritmus RSA pro šifrování zpráv, pro digitální podpis, nebo hashovací funkce s typy SHA-1 nebo SHA-2. [34; 38; 39]

Další možností výběru na trhu je kryptografický koprocesor od firmy Infineon. U této značky se pro zabezpečení a autentizaci v rámci IoT využívá typ Optiga Trust™. Koprocesory této značky se dále dělí na CHARGESLS32 pro zabezpečení externích zařízení jako třeba kamery nebo na zařízení M EXPRESS, které se používají pro smart home. Pro zabezpečení komunikace IoT se nejvíce hodí typ M SLS32AIA. Tento typ kryptografického koprocesoru má velmi rozmanité množství použitelných algoritmů k šifrování. Jsou schopny využívat algoritmus RSA do maximální velikosti klíč 2048, kryptografii ECC, standart AES-256, hashovací funkci SHA 256 nebo méně používané HMC (Hamiltonian Monte Carlo). Nevýhodou je že koprocesor je schopen využívat pouze protokol TLS verze 1.2. Naopak výhodou je velikost vnitřní paměti, a to 10 kB, 16bitový procesor nebo certifikací CC EAL6+ high HW. [40; 41]

Maxim Integrated je další značkou, která vyrábí kryptografické koprocesory. Jedním z jejích typů je typ MAX. Nejnovější model z této řady je MAX66301, který využívá algoritmus SHA-3 pro autentizaci. Tento typ dále disponuje integrovanou čtečkou RFID karet. Možné využití je například pro ruční čtečku RFID karet. Nevýhodou je, že koprocesor komunikuje pouze přes sběrnici UART. Dalším je typ DS28Exx, který lze využívat v rámci IoT. Nejnovější typ DS28E40 podporuje jak symetrickou, tak i asymetrickou kryptografii. To znamená že podporuje kryptografii ECC (ECDH, ECDSA) a hashovací funkci SHA-256. [42; 43]

Poslední ze zmíněných značek je značka NXP. Jako zajímavá volba se jeví kryptografický koprocesor SE050C. Koprocesor obsahuje jak softwarové, tak hardwarové zabezpečení. Jeho největší předností je množství využitelných algoritmů nebo standardů. Jako hlavní je využití kryptografie ECC s použitím DH/DSA a dalších. Nesmí chybět také standart AES s módy CBC, ECB, CTR a GCM s key derivation funkcí, hashovací funkce až do SHA-512, RSA nebo již méně používaný algoritmus 3DES a také HMC. Další výhodou je jeho veliká paměť a to 50 kB. Jisté nevýhody mohou vzniknout v komunikaci, protož koprocesor podporuje jenom rozhraní I^2C, nebo ve větším provedení koprocesoru a složitějšímu zapojení z důvodu 16 pinů. Další možností od této značky je koprocesor A1006TL/TA1NXZ, který má pouze 6 pinů, využívající méně využívané algoritmy k šifrování, ale stále podporuje pouze jedno komunikační rozhraní I^2C. [44]

4.4 Microchip ATECC

Firma Microchip je výrobcem mnoha různých čipů, mimo jiné kryptografických koprocesorů řady ATECC pro kryptografické zabezpečení klíčů a autentizaci. [45]

Zkratka ATECC je označení pro typ mikročipů CryptoAuthentication™ pro kryptografii pomocí ECC. Tyto mikročipy mají za úkol ochránit privátní klíče před útočníky. Jeho využití lze najít pro IoT, kde umožňuje celkové zabezpečení pro autentizaci, důvěrnost a následně integritu dat. Hlavní výhodou je že nevyžaduje velké množství energie pro provoz a lze jej připojit k jakékoli vývojové desce nebo mikroprocesoru, které disponují I^2C sběrnicí. Rozdíl od dalších značek je, že obsahuje optimalizované akcelerátory, které zajišťují rychlý provoz a zároveň nízkou spotřebu. Velmi důležitá vlastnost je, že má v sobě zabudován generátor velkých náhodných číslech, ze kterých se poté pomocí algoritmu generují privátní klíče. Privátní klíče lze ukládat právě do mikročipů, které obsahují paměť typu EEPROM. Dále obsahují ochranu proti útoku ochranným kanálem, kde se nesoustředí útok na matematickou chybu nebo na prolomení pomocí brute-force ale na fyzickou část mikročipu. ATECC disponuje také ochranou pomocí anti-tampering softwarem. [45]

Značka Microchip nabízí zákazníkům svoji knihovnu CryptoAuthLib, která slouží k práci s mikročipy CryptoAuthentication™. Tuto knihovnu lze použít například k čipům typu ATECC508 a ATECC608. Podporuje platformy Arm ®, Cortex ® a mnoho dalších. Pro osobní počítač podporuje operační systémy Linux a Windows. [45]
Nejnovější model od této firmy je typ ATECC608B, který podporuje symetrickou i asymetrickou kryptografii. [34; 45]

Jeho hlavní výhody jsou, že disponuje velkým výkonem pro matematické operace, velmi malé provedení a rozhraní API pro spojení s TLS protokolem [45].

Pro typ mikročipu ATECC firma Microchip představila takzvaný Trust Platform, který se skládá celkem ze 3 základních částí. [37]

Nejnovější kryptografický čip od firmy Microchip je ATECC608B. Jeho předchůdcem byl čip ATECC508, který má jistá omezení. Čip ATECC508 je schopen využít pouze tři algoritmy pro šifrování a autentizace, a to ECDH pro šifrování, ECDSA pro digitální podpis a SHA256 pro hashovací funkce. Jak je zde možné vidět, neobsahuje již dnes velice rozšířený standart AES pro šifrování zpráv. Výhody novější verze ATECC608B jsou vylepšený generátor náhodných čísel, možnost generovat dočasný klíč v paměti SRAM pro algoritmus ECDH nebo také rychlejší generování hash funkce. Samotný výrobce doporučuje přejít právě na novou verzi tohoto čipu pro vývoj IoT zejména, když je zajištěna kompatibilita mezi těmito prvky. [46; 47]

Jak bylo zmíněno výše, uvedené koprocesory se vyrábějí ve třech verzích. (ATECC108, ATECC508, ATECC608). Verze ATECC108 se v dnešní době již nepoužívá. Jednotlivé verze lze dále rozdělit na ATECCx08A nebo na ATECCx08B. Vždy platí že verze označená písmenem B je vylepšenou verzí svého předchůdce. Vylepšením se nejedná o změnu rozložení pinů nebo změnu algoritmů pro šifrování, nýbrž o odstranění vzniklých chyb zařízení. U koprocesoru ATECC608B tomu bylo přidáním těchto vlastností:

1. Nízkofrekvenční problém I^2C je problém, kdy zařízení se může chovat nepředvídatelně, to znamená poškodit odesílaná data, za určité situace (stejná I^2C sběrnice pro více zařízení, frekvence < 300 kHz).
2. Zařízení ATECC nemá na svém obalu označení, podle jakého by se zařízení mohlo identifikovat. Pro zjištění je zaveden režim Revision, který lze vyvolat pomocí příkazu `Info`.
3. U některých bezpečnostních vylepšení docházelo ke změně časového provedení příkazů.
4. Možnost objednat zařízení s větším teplotním rozsahem o +15°C. To znamená že maximální teplota může dosahovat až 100°C.
5. Zařízení je možné objednat i v tří pinovém provedení s pouzdrem RBH. [48]

Pro navazující praktickou část projektu byla zvolena verze koprocesoru ATECC508A, zejména z důvodu dostupnosti.

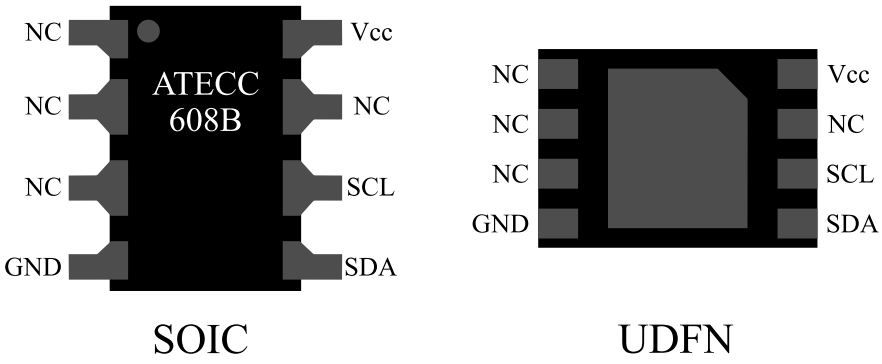
4.5 Koprocesor ATECC608B

Koprocesor ATECC608B je velice vhodný k použití pro tento projekt. Je totiž možné ho osadit do jakéhokoli mikrokontroleru, který disponuje I^2C sběrnici a tím pádem je kompatibilní s téměř všemi vývojovými deskami. V provedení s vývojovou deskou ESP32 mají koprocesory stejnou vlastnost, a to tu, že mají velmi malou spotřebu energie. Koprocesor je schopen pracovat celkem se 3 algoritmy/standardy. Pro využití v IoT se jeví jako nejlepší použít asymetrickou kryptografii s podporou ECC s 256bitovým klíčem, který může být rychlejší až o tisícinásobek oproti softwaru na

vývojové desce. Koprocesor využívá asymetrickou kryptografii pro protokol ECDH (Elliptic Curve Diffie Hellman) nebo pro ECDSA (Elliptic Curve Digital Signature Algorithm). Dále je možné používat symetrickou kryptografii s užitím AES128-GCM s 128bitovým klíčem nebo hashovací algoritmus SHA256 s 256bitovým klíčem. K dosažení zabezpečení obsahuje kryptografický koprocesor akcelerátor pro generování velikých náhodných čísel. Jednotlivé klíče ukládá na paměť EEPROM, kam je jich možné uložit až 16. Privátní klíče lze uložit na sloty 0 - 4. Sloty 8 - 15 slouží k ukládání potřebných dat pro self Signed Certificate nebo pro JWT token. Protože je paměť typu EEPROM, dá se dále využít pro ukládání certifikátů nebo pouze pro čtení. [47; 49]

Pro komunikaci v IoT se používá právě jeden z nejpodporovanějších protokolů, a to verze TLS 1.3 nebo jeho předchůdce verze 1.2. Pro šifrování a dešifrování zpráv je zde použit standart AES s módem ECB, kde je zpráva rozdělena do stejně velkých bloků a všechny bloky jsou šifrovány postupně. Je zde ale možné, že vznikne situace kdy, stejné bloky budou mít stejný text. V tomto případě je možné zaútočit pomocí opakování zpráv, protože jednotlivé bloky jsou dešifrovány stejným postupem. Pro zajištění autentizace se při šifrování zpráv využívá AES-GCM (Galois Counter Mode), protože mód ECB nepracuje s inicializačním vektorem. [47; 49]

Pro komunikaci s mikrokontrolerem, v tomto případě s vývojovou deskou, se využije sběrnice I^2C, s maximální rychlostí přenosu 1 Mbps. Pro mikrokontrolery, které nejsou vybaveny I^2C, je možné koprocesor připojit pomocí sběrnice SWI na jakýkoli pin GPIO. [47; 49]



Obrázek 4.1 Koprocesor ATECC608B zobrazení pinů [47]

V nejběžnějších pouzdrech se koprocesor vyrábí s 8 piny pro SOIC nebo UDFN montáž. Piny označené NC značí nevyužité piny, jako V_{CC} se označuje napájecí pin, o velikosti napětí od 2V až 5,5V, pin označený GND udává uzemnění, pin SDA je pak označení pro sériová data a poslední pin s označením SCL je označení pro sériové hodiny. [47]

Koprocesor může v základu již obsahovat předdefinované varianty bezpečnosti nazývané Trust Platform. Celkem obsahuje tři varianty. První varianta je Trust&GO kde jsou před konfigurované prvky bezpečnosti. Druhou variantou je TrustFLEX. Zde jsou

již do koprocesoru nahrány požadované přihlašovací údaje a námi určená předdefinovaná konfigurace. Třetí varianta je TrustCustom u níž je možné si koprocesor nechat přednastavit úplně celý podle námi určené konfigurace. V této práci se jeví jako nejlepší variantu zvolit již celkově předdefinované nastavení tudíž variantu Trust&GO. Z důvodu nedostupnosti byl zvolen typ TrustCustoms. [47; 49]

Microchip vyrábí celkem dva základní druhy koprocesorů pro autentizaci, a to ATECC508 a ATECC608. Tyto dva typy mají dále své poddruhy zakončením písmeny A nebo B. Jak bylo zmíněno výše, tyto koprocesory mají drobné rozdíly. Nejlepší vlastností těchto koprocesorů je, že se u nich nemění rozmístění pinů nebo třeba velikosti pamětí. Toto vytváří možnost snadno nahrazovat koprocesor novějšími nebo předešlými verzemi koprocesorů. I když bylo nutné pro tento projekt zvolit starší verzí koprocesoru (ATECC508A) z důvodu nedostupnosti produktu, není problém tento koprocesor vyměnit za novější verzi ATECC608B.

5. PRAKTICKÁ ČÁST

5.1 Výběr mikrokontroleru

Jeden ze dvou stěžejních prvků této práce je volba mikrokontroleru. V tomto výběru bylo nutné uvažovat několik faktorů, zejména výkon mikrokontroleru, struktura podporovaných rozhraní a jeho dostupné modely. Důležitým faktorem, který hraje roli, je připojení kryptografického koprocesoru k samotnému mikrokontroleru, který disponuje komunikací po I^2C sběrnici. Lze tedy vyloučit nejlevnější řady modelů, které právě nedisponují těmito vlastnostmi.

Z výše uvedených důvodů se jako nejlepší volba jeví mikrokontroler od firmy Espressif a to model ESP32, který nejen že má výkonnější čip než běžně používané modely jiných značek, jako například Arduino, nebo STM32. Mikrokontroler je možné programovat v různých vývojových prostředích, ale také u něj není určen jeden programovací jazyk jako třeba u Arduina. Za zmínku také stojí, že cena onoho mikrokontroleru se pohybuje okolo 13€, je tedy více než o polovinu levnější než u konkurenčních výrobců. V neposlední řadě je třeba zohlednit předchozí zkušenosti s tímto mikrokontrolerem.

5.2 Výběr kryptografického koprocesoru

Na trhu se objevují různé modely od různých výrobců. Pro zabezpečenou komunikaci v rámci IoT se nejvíce hodí kryptografické koprocesory z řad použití pro autentizaci. Pro následnou implementaci byl zvolen kryptografický koprocesor od firmy Microchip z odvětví CryptoAuthentication™ přesněji model ATECC508A. Na trhu již existuje jeho nástupce s názvem ATECC608B, ale z důvodu aktuální nedostupnosti byl zvolen jeho předchůdce. Firma Microchip spolupracuje s mnoha firmami z různých odvětví. To může zahrnovat snažíš komunikaci s IoT servery nebo implementaci v knihovně Mbed-TLS pro komunikaci s mikrokontrolerem a snadnějším navázáním zabezpečeného spojení. Dalším kritériem při výběru byla dostupnost dokumentace. V mnoha případech nebyla dokumentace dostatečná k realizaci projektu nebo dokumentace byla pouze v čínském jazyce.

5.3 Vývojové prostředí

K celkovému programování kryptografického koprocesoru a mikrokontrolerů bylo zvoleno vývojové prostředí od Microsoftu s názvem Visual Studio Code. V tomto prostředí lze programovat ve Wiringu pomocí Platform ide. Jeho výhodou je již předchozí zkušenost s ním a jednodušší struktura celého projektu. Nicméně toto prostředí lze využít pro ESP-IDF, do kterého je možné naimplementovat Cryptoauthlib. Pro všestrannost tohoto vývojového prostředí byl zvolen program Visual Studio Code.

5.4 Pár klíčů a certifikáty

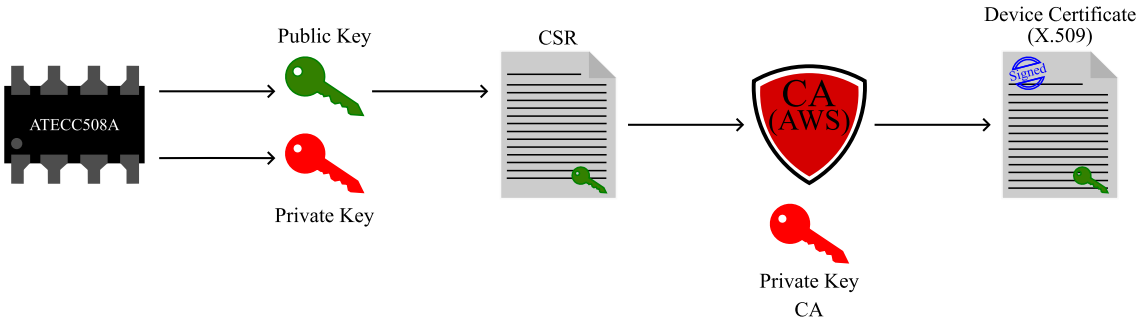
Pro navázání zabezpečeného spojení mezi mikrokontrolerem a zařízením (což může být server nebo i třeba další mikrokontroler) slouží právě certifikáty, které generuje zařízení ATECC. Pokud by se nevyužíval kryptografický koprocesor, tak certifikát a pár klíčů se musí vygenerovat samostatně a musejí být uloženy v samotném mikrokontroleru v NVS oblasti. I když je tato oblast šifrována, je možné je snadno získat pomocí proudové analýzy. Z tohoto důvodu se využívají kryptografické koprocesory, kde vygenerovaný privátní klíč neopustí zařízení.

5.4.1 Požadavek na certifikát (CSR)

Požadavek na certifikát CSR (Certificate Signing Request) využívá AWS Iot Core od Amazonu [50].

Jde o typ požadavku, který obsahuje veřejný klíč a další parametry jako: název státu a města, jméno organizace, doba platnosti požadavku, vlastní název (CN) nebo další. Z těchto parametrů musí požadavek obsahovat CN, avšak ostatní parametry jsou nepovinné. [51]

Koprocesor ATECC vygeneruje dva klíče (veřejný a privátní). Poté vytvoří požadavek na certifikát, který obsahuje předdefinované parametry a vloží do něj veřejný klíč. Tímto spojením se vytvoří požadavek na certifikát CSR, který využívá CA (Certificate Authority). V tomto případě je CA Amazon AWS. CSR je poskytnut CA, která ověří pravost parametrů. Pokud jsou parametry podle CA pravé, tak vygeneruje certifikát a podepíše jej svým privátním klíčem. Z tohoto postupu vznikne Device Certificate, který je potřeba nahrát do mikrokontroleru. [52]



Obrázek 5.1 Postup vytvoření Device Certificate [52]

5.4.2 Certifikát Self-signed

Tento certifikát požaduje například Azur IoT Hub od Microsoftu [53]. Od požadavku na certifikát CSR se liší tím, že není podepsán důvěryhodnou certifikační autoritou, ale je podepsán „sám sebou“, což znamená, že vydávání a podpis certifikátu dělá sám vlastník. Pokud by byl certifikát vygenerován pomocí koprocesoru ATECC, tak se

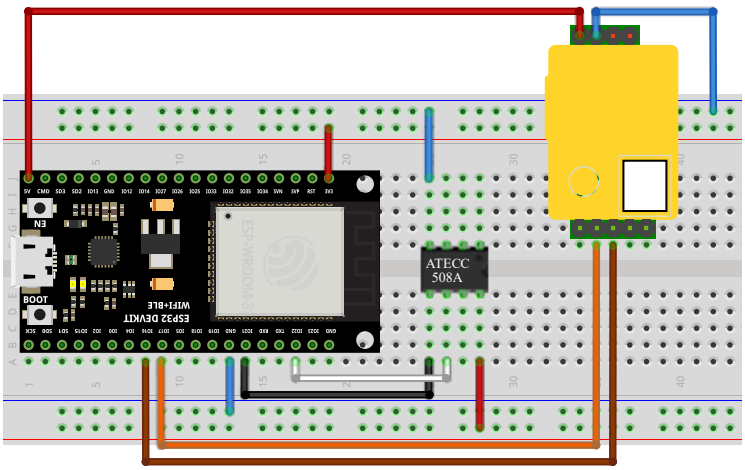
k podpisu certifikátu použije jeho jedinečné sériové číslo. Při generování se musejí zadat parametry, od jakého dne a od jakého času bude certifikát platný, a kdy jeho platnost vyprší. Na web Azure se nahraje v podobě hashe.

5.4.3 JWT token

Autentizaci pomocí JWT tokenu využívá kupříkladu Google Cloud IoT Core [54]. Mikrokontroler vygeneruje JWT token a ten je následně pomocí privátního klíče v ATECCu podepsán. Tento podepsaný JWT token slouží k navázání zabezpečené komunikace.

5.5 Schéma zapojení Mikrokontroleru a kryptografického koprocesoru ATECC508A

Jako první je nutné si definovat piny mikrokontroleru, které podporují I^2C komunikaci. ESP32 má hardwarově nastavené SDA na pinu 21 a SCL 22. Toto nastavení lze změnit pomocí softwaru na téměř všechny piny, které deska obsahuje. V tomto projektu byl pro demonstraci funkčnosti kryptografického koprocesoru využit senzor ovzduší. Nicméně je možné tento senzor nahradit jinými senzory nebo aktivními prvky (například servomotor) v rámci IoT aplikací. Pro praktickou část projektu bylo zvoleno schéma zapojení součástek podle obrázku 5.2.



Obrázek 5.2 Schéma zapojení ESP32 a ATECC508A [55]

5.6 Generování páru klíčů a certifikačního požadavku

Pro generování certifikačního požadavku a páru klíčů vydal Microchip speciální knihovnu s podporou API. Knihovnu lze využít pro téměř všechny typy kryptografických koprocesorů, které firma nabízí. Pro tento projekt byly testovány celkem dvě knihovny. A to knihovna od společnosti Arduino a knihovna od Espressif,

která využívá původní Microchip knihovnu, ale implementuje jí do vývojového prostřední ESP-IDF a zprostředkovává komunikaci s mikrokontrolerem.

5.6.1 Knihovna pro Arduino

Pro generování CSR, Self-signed Certifikát nebo JWT token lze využít knihovnu od Arduina ECCX08, která disponuje nástroji pro samotné generování certifikátů [56]. Knihovna je kompatibilní jak s předchozí verzí koprocesoru ATECC508, tak i s novější ATECC608. Knihovna byla napsaná primárně pro Arduino, tudíž musejí být provedeny následující změny.

Arduino má jiné rozložení pinů pro I^2C sběrnici než ESP, a proto se tyto piny musí na deklarovat samostatně. Toto se provede následující úpravou.

```
#include <Wire.h>

#define SDA_PIN     32
#define SCL_PIN     33

void setup() {
  Wire.begin(SDA_PIN, SCL_PIN);
```

Po nahrání kódu do ESP se stane následující.

1. Zahájí se sériová komunikace mezi mikrokontrolerem a PC. Jako další se zjišťuje, zda se ATECC nachází na daných pinech I^2C sběrnice.
2. Přes sériový monitor se musejí zadat parametry pro vygenerování CSR (některé parametry mohou zůstat prázdné). Nakonec se zvolí jeden z pěti slotů, na kterém se vygeneruje pár klíčů.
3. Koprocesor vygeneruje pár klíčů na daném slotu (pokud již slot obsahuje pár klíčů, využijí se již vygenerované). Požadavek na certifikát CSR v sobě obsahuje veřejný klíč a je podepsán příslušným soukromým klíčem.
4. Mikrokontroler vrátí po sériové lince vygenerovaný CSR.

Výhodou využívání právě této komponenty je, že lze snadno zvolit jeden z pěti slotů pro vygenerování párů klíčů a snadné zadání parametrů pro CSR. Dále disponuje snadnou implementací pro jiné typy mikrokontrolerů (pokud podporují jazyk Wiring).

Výhodnou i nevýhodnou je, že komponenta neumožňuje využít možnosti zamknutí koprocesoru. To znamená, že se může donekonečna generovat CRS pomocí párů klíčů. Nicméně privátní klíč nelze měnit, tudíž se vygenerují pouze jednou.

Nevýhodou je, že pro generování CSR je nutné zkompilovat celý program a nahrát jej do mikrokontroleru. Toto může být velice zdlouhavé při potřebě nakonfigurovat více zařízení. Za další nevýhodu lze považovat, že výsledný CSR je zobrazen na sériový

monitor a neukládá se do souboru. Zde mohou vzniknout nepříjemnosti se špatným zkopírováním.

5.6.2 Knihovna od ESP

Pro generování CSR byl využit komponent esp-cryptoauthlib, který byl původně navržen pro jiný typ mikrokontroleru, a to pro ESP32-SE a pro novější typ kryptografického koprocesoru ATECC608A [57]. Aby bylo možné tento komponent využít, musí se provést důležité změny v kódu a v nastavení pro sestavení projektu.

Pro změní defaultních hodnot pinů je nutné v **`secure_cert_mfg.py`** najít konfiguraci těchto pinů a změnit jejich defaultní hodnoty na námi požadované. Následně upravený kód bude vypadat takto.

```
parser.add_argument(
    ..
     default=32,type=int,
    ..

parser.add_argument(
    ..
     default=33,type=int,
    ..
```

Dále je třeba upravit adresu I^2C sběrnice podle toho jaký kryptografický koprocesor se využije. Koprocesor ATECC608 disponuje celkem třemi různými hodnotami I^2C adres podle toho, o jaký typ koprocesoru jde. Podle dostupných dokumentů bylo zjištěno, že starší verze ATECC508 disponuje pouze jednou hodnotou, a to 0xC0 což odpovídá typu ATECC608 Trust Custom z novější řady. Výběr koprocesoru, adresa I^2C a použité piny pro I^2C komunikaci je možné nastavit v `Kconfig` nebo v GUI IDF pomocí `idf.py menuconfig`.

```
choice ATECC608A_TYPE
       ..
        default ATECC608A_TCUSTOM
       ..
endchoice
```

Z tohoto důvodu, že tato komponenta vychází z knihovny Cryptoauthlib od firmy Microchip, která podporuje všechny jejich řady koprocesorů, a fakt že novější verze disponuje jenom vylepšením staré verze, bylo docíleno že komponentu lze využít i pro nakonfigurování ATECC508A a zprostředkování komunikace s mikrokontrolerem.

Poslední problém nastal při sestavování projektu u definování knihovny. Konkrétně u `mbedtls/pk_internal.h` se zobrazováním chyby „No such file or direktory“. Tuto chybu lze odstranit přímo v nastavení ESP-IDF nebo v `idf.py menuconfig`, kde

u komponentu esp-cryptoauthlib stačí povolit funkci hardwarové ECDSA klíče pro knihovnu mbedTLS.

K samotné konfiguraci ATECC508A slouží obslužný program s názvem esp_cryptoauth_utility, který je přímo integrovaný v komponentu esp-cryptoauthlib. Jeho výhodnou je, že ke konfiguraci koprocesoru není potřeba nahrávat celý program do mikrokontroleru, nýbrž komunikace probíhá po sérové lince. Tudíž ušetří čas při konfiguraci více zařízeních. Nejdříve je nutné tento program nainstalovat pomocí příkazu.

```
pip install esp-cryptoauth-utility
```

5.6.3 Generování požadavku na certifikát a párů klíčů pomocí esp-cryptoauthlib

Při využívání této utility se naskytují dvě možnosti. První možností je, že požadavek na certifikát (CSR) si lze nakonfigurovat podle vlastních zvolených parametrů jako je název CSR, název organizace apod. a s těmito parametry kryptografický koprocesor vygeneruje CSR. Nebo existuje druhá možnost, že se využije „certifikát“ s předem definovanými parametry, který se nachází v `sample_certs` s názvem `sample_signer_cert` a `sample_signer_key`. Obrovskou nevýhodou esp-cryptoauthlib je, že dokáže vygenerovat pouze požadavek na certifikát CSR.

5.6.4 Nastavení vlastních parametrů pro CSR pomocí esp-cryptoauthlib

K samotnému generování klíčů je možné využít mnoho generátorů. V tomto případě byla zvolena knihovna openssl, která je již zahrnut jako komponent v ESP-IDF.

Pro vygenerování CSR se musí provést tyto dva příkazy.

```
openssl ecparam -out sample_signer_key.pem -name prime256v1 -genkey

openssl req -new -x509 -key sample_signer_cert.pem -out signercert.pem
        -days 365
```

U prvního příkazu podle parametru `ecparam` lze poznat že, jde o generování klíčů pro ECC kryptografii. Podle parametru `prime256v1` lze říci, že se bude jednat o klíče o velikosti 256 bitů. Tato dvojice klíčů bude následně uložena do souboru signerkry.pem pro podepsání požadavku na certifikát (CSR).

Druhý příkaz podle parametru `req` zajišťuje generování certifikátu konkrétně CSR pro standard x509. To znamená, že CSR bude vytvořen pomocí privátního a veřejného klíče podle .pem formátu. Jako poslední parametr lze nastavit, jak dlouho bude certifikát platný. V tomto případě to bude 365 dní. Pokud parametr nebude zadán, bude tato hodnota nastavena na 30 dní.

5.6.5 Vygenerování CSR pomocí esp-cryptoauthlib

Podle toho, jaký typ koprocesoru máme, tak se provede jeho prvotní konfigurace. Pro typ TrustCustoms se nejdříve vygeneruje CSR, který v sobě obsahuje veřejný klíč. Tento certifikát bude následně podepsán pomocí soukromého klíče. U typu Trust&Go a TrustFles podpisující certifikát mikročipu podepíše certifikát zařízení. Následně jsou pak vygenerovány prvky jako soukromý klíč, veřejný klíč a podpisový certifikát. Těmito prvky je následně podepsán soubor manifest.

Pokud bylo zvoleno využití již přednastavených parametrů pro vygenerování CSR, stačí provést tento příkaz.

```
python secure_cert_mfg.py --port COMx.
```

V této práci byla využita možnost si vygenerovat CSR s vlastními parametry. To znamená že certifikáty, které se vygenerovaly pomocí openssl nahradí již defaultní certifikáty ve složce sample_certs. Příkaz pro vygenerování CRS zůstává stejný.

1. Jako první se testuje a následně se navazuje připojení pomocí sériového spojení mezi počítačem a mikrokontrolerem. Následně s kryptografickým koprocesorem. K tomuto slouží parametr `--port COMx` (kde x značí číslo sériového portu). Toto navázání je velmi důležité, protože celý skript funguje na bázi API a jeho obsah není nahráván do samotného mikrokontroleru.
2. Probíhá testování spojení mezi mikrokontrolerem a kryptografickým koprocesorem.
3. Mikrokontroler zažádá o provedení příkazu pro vygenerování privátního a veřejného klíče pro kryptografii ECC o velikosti 256 bitů. Pokud vše proběhne správně, kryptografický koprocesor předá mikrokontroleru soukromý klíč. A zobrazí jej v terminálu.
4. V tomto kroku je vygenerován požadavek na certifikát CSR, který je podepsán vygenerovaným párem klíčů.
5. Po provedení předchozího kroku je vygenerovaný CSR poslán do mikrokontroleru a následně do počítače. Vygenerovaný CSR se zobrazí na sériovém monitoru a následně je uložen do složky `output_files` jako soubor .pem s názvem device_cert.pem.

Výhodou této komponenty je, že umožňuje rychlé nakonfigurování a vygenerování CSR a následné jeho uložení do souboru.

Nevýhodou se může zdát, že komponenta nedisponuje možností celkového uzamčení koprocesoru. Další nevýhodou je složitější a zdlouhavý proces při nastavování vlastních parametrů pro CSR při využívání openssl. Tato komponenta byla dále testovaná u mikrokontroleru typu ESP32-S3. Výsledkem bylo zjištění, že komponenta nerozpoznala mikrokontroler a tím nebylo možné vygenerovat CSR.

5.7 IoT servery

V tomto projektu pro názornou ukázku funkčnosti kryptografického koprocesoru byla zvolena možnost navázání zabezpečeného spojení s IoT serverem.

Na trhu se objevují jak veřejné (zdarma), tak i privátní (placené) servery. Mezi veřejné servery se řadí například Eclipse nebo Mosquitto. Všechny veřejné IoT servery obsahují nějaké omezení. Může to být například: nepodpora TLS portu nebo omezování odeslaných zpráv za den. Z těchto důvodů se vybíral privátní IoT server.

K nejznámějším privátním serverům, které byly po vizuální stránce, přívětivosti pro klienta, funkcemi a nejlepším přizpůsobením pro využití s ESP32 a kryptografickým koprocesorem testovány, patří AWS, Azure a Google IoT Core. I kdyby se mohl zdát jako dobrá volba server od Googlu, tak samotný Google přestane tuto službu poskytovat od září 2023 [54].

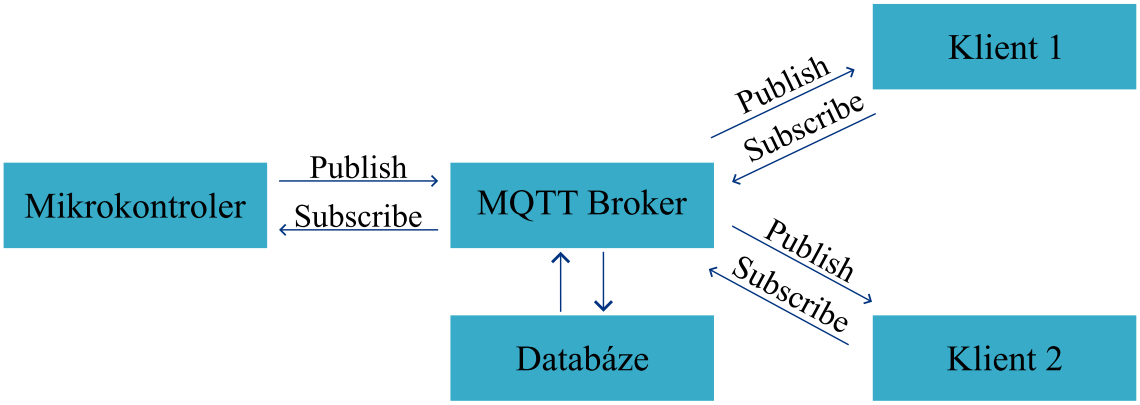
Pro názornou ukázku funkce kryptografického koprocesoru byl vybrán server od Amazonu AWS z důvodu snazšího uživatelského nastavování, přívětivější grafiky a intuitivního ovládání.

5.8 Protokoly pro přenos zpráv

Existuje velké množství protokolů pro přenos zpráv. Mezi nejznámější patří HTTP (HyperText Transfer Protocol), LoRa-WAN (Long Range Wide Area Network), ZigBee, MQTT (Message Queue Telemetry Transport) a mnohé další. Z důvodu využití AWS se výběr vhodného protokolu eliminuje pouze na dva typy, a to na HTTP a MQTT.

Protokol HTTP se řadí mezi nejpoužívanější protokoly pro využití v IoT i když má mnohé nedostatky. Jeden ze zásadních nedostatků je, že nepodporuje takzvanou komunikaci Publish/Subscribe, která podporuje vzájemnou komunikaci mezi serverem a mikrokontrolerem, ale podporuje pouze komunikaci zvanou Request/Response. Ta je schopna odeslat zprávu na požadavek serveru. Další nevýhody mohou být větší spotřeba energie nebo cena provozu. [58]

MQTT má naproti tomu mnoho výhod. Využívá se pro shromažďování různých hodnot, které se pak následně odesílají přes MQTT broker, na kterém je možné si dané hodnoty zobrazit nebo je odeslat na další zařízení. Celková komunikace MQTT protokolu probíhá nad TCP protokolem ve stylu Publish/Subscribe 5.3 [58].



Obrázek 5.3 Znázornění komunikace MQTT protokolu [59]

MQTT protokol také není tak energeticky a výkonově náročný jako HTTP [58]. Tím pádem je ho možné využívat u levnějších zařízení. Z těchto výše uvedených důvodů byl zvolen protokol MQTT.

5.9 Průběh připojení na Amazon AWS

Pro připojení byla původně vybrána komponenta esp-aws-iot od EspressiF. Nicméně tato komponenta nepočítala s využitím ATECC TrustCustoms. Pro využití v tomto projektu, byla nalezena upravená verze této komponenty s názvem PBearson/esp-aws-iot pro využití s ATECC TrustCustoms [60]. Do této komponenty byla naimplementovaná nejnovější verze esp-cryptoauthlib.

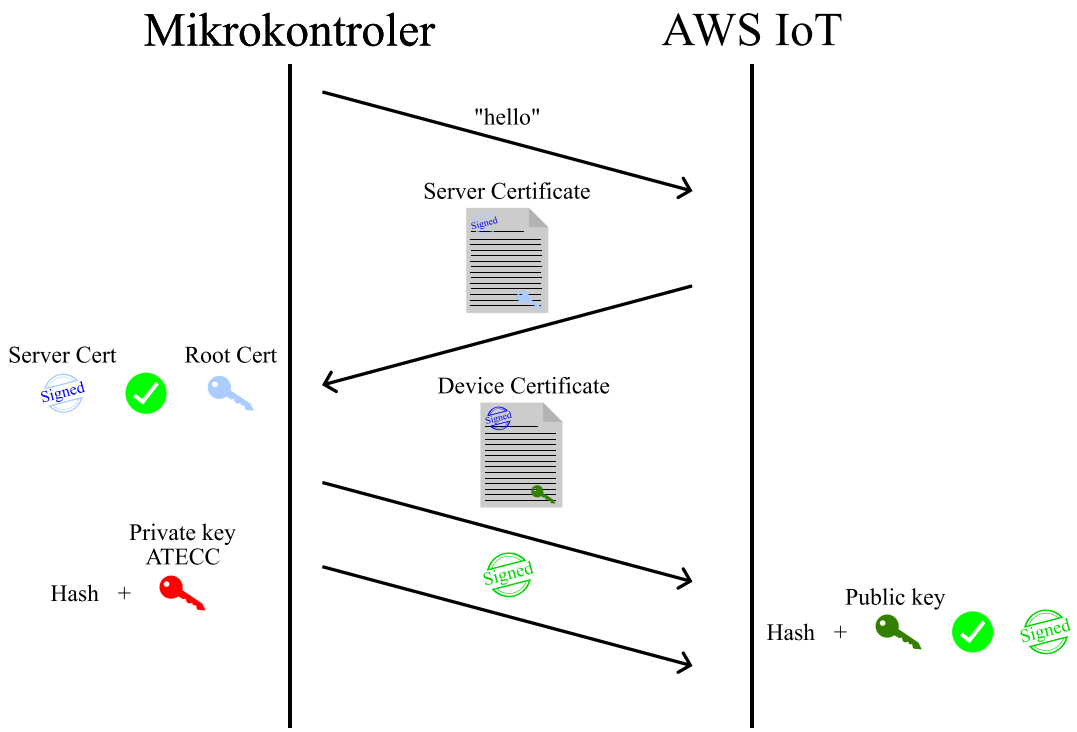
Nejprve je nutné připojit daný koprocesor k mikrokontroleru. V tomto případě je použita I^2C sběrnice. Zda je koprocesor přítomen a správně nakonfigurován zodpovídá komponent esp-cryptoauthlib.

Pro připojení k AWS je nutné vybrat port. Pro využívání MQTT protokolu, zabezpečené komunikace pomocí TLS a využití certifikátu zařízení X.509 se nabízejí dvě možnosti. Jako první je využít portu číslo 443, zde se však musí poskytnout ještě nastavený ALNP (Application Layer Protocol Negotiation) protokol pro navázání TLS spojení. Druhou a schůdnější možností je využít portu číslo 8883 kde není zapotřebí využívat ALPN protokol. [61]

V současné době AWS disponuje celkem 27 regiony. Pro projekt tvořený v Evropě byl použit nejbližší server v Paříži. Každý klient po registraci do AWS IoT Core obdrží klient „endpoint“, který je unikátní pro každého uživatele a pro každý region serveru. Endpoint se skládá z unikátního klíče, obsahu služby (IoT core), regionu a záhlaví. Pro správou funkčnost je nutné nastavit ještě „Thing“, „Policy“ a nahrát CSR. [62]

Pro navázání zabezpečeného spojení musejí být na mikrokontroleru nahrány dva certifikáty. První je Device Certificate, který se získá od CA a druhý je „Server Root certificate“, který poskytne AWS IoT.

Jako první odešle mikrokontroler zprávu „hello“ s endpointem na komponentu autentizace AWS IoT Core. Na tento požadavek reaguje server odesláním svého „Server certificate“ na mikrokontroler. Certifikát je poté ověřován mikrokontrolerem pomocí Server Root certificate, který vlastní. Ověření probíhá pomocí veřejného klíče obsaženého v Server Root certificate, který ověřuje digitální podpis přijatého certifikátu Server certificate. Pokud je podpis pravý, tak mikrokontroler má jistotu, že komunikuje s AWS a pokračuje v komunikaci. Následně se mikrokontroler musí autentizovat. To provede pomocí Device certificate, který odešle na AWS IoT. K autentizaci ještě mikrokontroler odesílá digitální podpis. Podpis je vypočítán pomocí privátního klíče, který se nachází v koprocesoru a nikdy neopustí zařízení, a hashe. Hash se vypočítává pomocí všech záznamů komunikace. Stejným způsobem vypočítává hash také AWS. V tuto chvíli AWS IoT vlastní digitální podpis, který zkontroluje pomocí kombinace svého hashe a veřejného klíče, který získal pomocí Device certificate. Pokud „handshake“ proběhl úspěšně, tak obě strany mohou spolu komunikovat pomocí šifrované komunikace. [52]



Obrázek 5.4 Znázornění navázání zabezpečeného spojení s AWS IoT [52]

5.10 Senzor oxidu uhličitého

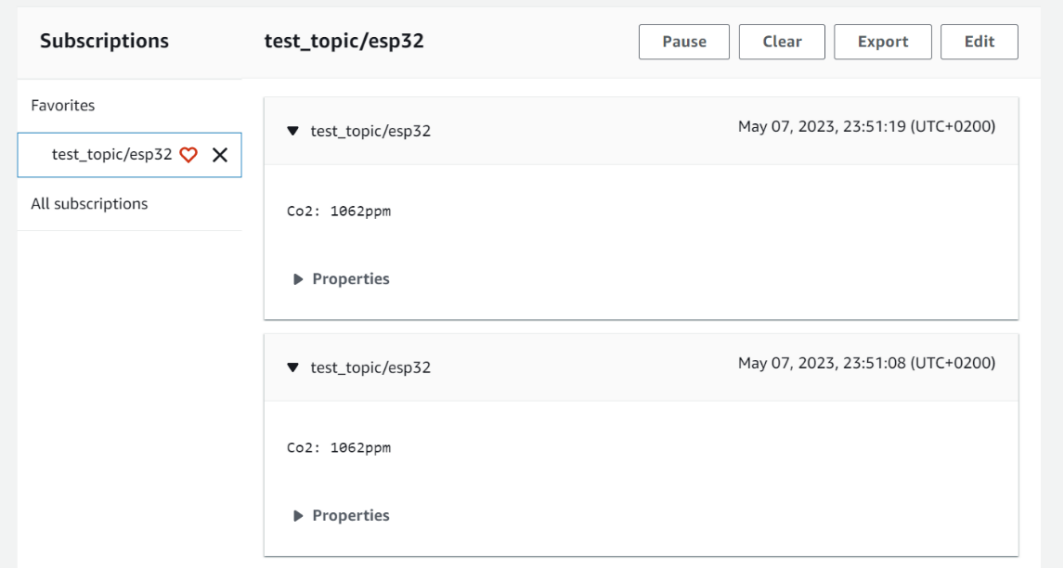
Pro demonstraci funkčnosti navázání spojení a následného odesílání hodnot byl naimplementován senzor oxidu uhličitého MH-Z19B, na bázi infračerveného záření, kde okamžitá hodnota oxidu uhličitého je odesílána v intervalu deseti sekund na server AWS.

Senzor je možné připojit různými způsoby k mikrokontroleru. Pro zajištění komunikace mezi mikrokontrolerem a senzorem byl využit komponent pro ESP-IDF s názvem esp-idf-lib [63]. Pro senzor je nutné zajistit napájení 5V. Senzor disponuje analogovým výstupem, PWM a pinem pro kalibraci teplotního kompenzačního prvku. V této práci byl zvolen výstup UART. Pro výstup TX se využil pin číslo 17 a u výstupu RX pin číslo 16. Pokud by byla zvolena hlavní (hardwarová) UART linka musel by se tento senzor vždy odpojovat při nahrávání programu do mikrokontroleru. Proto byly zvoleny piny, které se nastavují pomocí softwaru. [64]

Senzor se při prvotním startu nahřívá několik jednotek až desítek sekund. Následně sbírá hodnoty oxidu uhličitého v rozsahu 0 – 5000ppm v daném uzavřeném prostředí. [64]

5.11 Zobrazení komunikace mezi mikrokontrolerem a AWS

Mikrokontroler hodnoty ze senzoru přijímá a pomocí zabezpečené komunikace v podobě stringu předává AWS serveru. Hodnoty v prostředí se dají na AWS zobrazit nebo stáhnout v souboru Json pro další využití. Pro zobrazení hodnot je nutné zadat „subscription topic“, podle kterého lze filtrovat přicházející hodnoty z různých zařízení.



Obrázek 5.5 Ukázka přijatých dat v AWS IoT

Data je možné stáhnout z AWS IoT v podobě Json. Formáty jedné přijaté hodnoty budou vypadat následovně.

```
{
      "format": "string",
      "topic": "test_topic/esp32",
      "timestamp": 1683496368553,
      "payload": "Co2: 1467ppm",
      "qos": 0
},
```

Za použití protokolu MQTT je možná komunikace mezi serverem a mikrokontrolerem v obou směrech 5.6.

```
I (155046) subpub: test_topic/esp32     {
  "message": "Hello from AWS IoT console"
}
```

Obrázek 5.6 Ukázka přijetí zprávy mikrokontrolerem od AWS IoT

6. ZÁVĚR

Bakalářská práce zkoumá možnosti zabezpečení přenosu dat v rámci IoT. To znamená, jaké protokoly se použijí na přenos dat nebo pomocí jakých algoritmů bude šifrován celý přenos.

Prví část popisuje algoritmy nebo standardy se kterými se můžeme setkat při běžném použití. Důraz je kladen na RSA, ECDH, ECDSA a AES. Tyto algoritmy a standardy byly vybrány proto, že je nejčastěji využívají kryptografické koprocesory.

Druhá část se věnuje popisu verzí aplikačního protokolu TLS v rámci TCP/IP. Zde bylo zjištěno mnoho výhod u verze 1.3. I přes všechny výhody byla zvolena verze 1.2 z důvodu lepší spolupráce mezi serverem AWS a mikrokontrolerem ESP32.

Třetí část se zaměřuje na výběr mikrokontroleru. V kapitole jsou popsáni různí výrobci a jejich modely mikrokontrolerů. Z těchto výrobců byl vybrán výrobce Espressif se svým mikrokontrolerem ESP32, na kterém je postavena vývojová deska použitá v této bakalářské práci. Tato vývojová deska byla vybrána z důvodu velkého výkonu, podporovatelnosti vývojových prostředí nebo většího množství podporovaných programovacích jazyků.

Čtvrtá kapitola pojednává o kryptografických koprocesorech. V úvodu je základní popis kryptografických koprocesorů a základní vlastnosti, kterými disponují. Poté následuje přehled výrobců na trhu. Z tohoto hodnocení vyplynulo, že nejlepší volbou bude výrobce Microchip se svým modelem pro autentizaci ATECC608B. Nicméně koprocesor ATECC608B je v současné době nedostupný. Z tohoto důvodu byl zvolen jeho předchůdce ATECC508A, protože bylo zjištěno že koprocesor je zpětně kompatibilní s novější verzí a je možné ho za ní kdykoli vyměnit.

Závěrečná kapitola je praktická implementace kryptografického koprocesoru. Jako první byl zvolen server pro IoT, jímž byl AWS IoT Core od Amazonu. V druhém kroku bylo nutné vygenerovat požadavek na certifikát CSR, který je nutné nahrát na AWS. V následující části byly nalezeny dvě možnosti, jak vygenerovat CSR prostřednictvím koprocesoru ATECC a jak z něho získat Device Certificate, který je nutný nahrát do mikrokontroleru, aby moha být zajištěna autentizace. Poté je nutné zvolit protokol pro přenos zpráv. Vybrán byl protokol MQTT z důvodu komunikace Publush/Subscribe. V předposlední části této kapitoly je podrobně popsáno navázání spojení a výměna klíčů mezi AWS, mikrokontrolerem a kryptografickým koprocesorem ATECC. Na závěr této kapitoly se demonstruje funkčnost koprocesoru, který zajišťuje navázání komunikace se serverem AWS. Na úplném závěru je popis odesílání dat ze senzoru prostředí na server AWS a zobrazení těchto hodnot v konzoly AWS.

7. BIBLIOGRAFIE

[1] JANKO, David. Lekce 2 - Symetrická a asymetrická kryptografie. In: *Itnetwork* [online]. Copyright © 2022 itnetwork.cz [cit. 2022-11-17]. Dostupné z: https://www.itnetwork.cz/bezpecnost/symetricka-a-asymetricka-kryptografie

[2] Symetrická kryptografie. In: *Wikisofia* [online]. wikisofia.cz [cit. 2022-11-17]. Dostupné z: https://wikisofia.cz/wiki/Symetrick%C3%A1_kryptografie

[3] BEZPALEC, Pavel. Kryptografie: Techniky realizace kryptografických systémů. In: *Publi* [online]. © 2015 ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE, 2015 [cit. 2022-11-19]. Dostupné z: https://publi.cz/books/232/04.html

[4] COBB, Michael. RSA algorithm (Rivest-Shamir-Adleman). In: *TechTarget* [online]. © 2000 - 2020 TechTarget All Rights Reserved, 2021 [cit. 2022-11-17]. Dostupné z: https://www.techtarget.com/searchsecurity/definition/RSA

[5] What is RSA? How does an RSA work?. In: *Encryption Consulting* [online]. © 2022 Encryption Consulting LLC. All Rights Reserved [cit. 2022-11-17]. Dostupné z: https://www.encryptionconsulting.com/education-center/what-is-rsa/

[6] Doporučení v oblasti kryptografických prostředků. In: *Národní úřad pro kybernetickou a informační bezpečnost* [online]. Brno: Národní úřad pro kybernetickou a informační bezpečnost, 2022 [cit. 2022-11-23]. Dostupné z: https://www.nukib.cz/download/publikace/podpurne_materialy/Kryptograficke_prostredky_doporuceni_v2.0.pdf

[7] FROEHLICH, Andrew. Elliptical curve cryptography (ECC). In: *TechTarget* [online]. © 2000 - 2022 TechTarget All Rights Reserved, 2022 [cit. 2022-11-17]. Dostupné z: https://www.techtarget.com/searchsecurity/definition/elliptical-curve-cryptography

[8] F5 DEVCENTRAL. Elliptic Curve Cryptography Overview. In: *YouTube* [YouTube video]. 2015 [cit. 2022-11-17]. Dostupné z: https://www.youtube.com/watch?v=dCvB-mhkT0w&ab_channel=F5DevCentral

[9] HAJNÝ, Jan. *Přednáška předmětu BZKR - Základy kryptografie 7*. Brno, 2022. Přednáška. Vysoké učení technické v Brně.

[10] BERNSTEIN, Corinne a Michael COBB. Advanced Encryption Standard (AES). In: *TechTarget* [online]. © 2000 - 2022 TechTarget All Rights Reserved, 2022, 2021 [cit. 2022-11-26]. Dostupné z: https://www.techtarget.com/searchsecurity/definition/Advanced-Encryption-Standard

[11] COMPUTERPHILE. AES Explained (Advanced Encryption Standard) - Computerphile. In: *YouTube* [YouTube video]. 2019 [cit. 2022-11-26]. Dostupné z: https://www.youtube.com/watch?v=O4xNJsjtN6E&ab_channel=Computerphile

[12] AES Animation. In: *CryptTool-Online* [online]. Copyright © 1998 - 2022 CrypTool Contributors [cit. 2022-11-26]. Dostupné z: https://www.cryptool.org/en/cto/aes-animation

[13] GHANIM WADDAY, Ahmed, Salim WADI, Hayder MOHAMMED a Ali ABDULLAH. *Study of WiMAX Based Communication Channel Effects on the Ciphered Image Using MAES Algorithm* [online]. 2018, [cit. 2022-12-03]. ISSN 0973-4562 Volume 13.

[14] What's the difference between Elliptic Curve OpenPGP keys and AES-256. In: *DidiSoft* [online]. Copyright © DidiSoft Inc 2006 - 2022, 2013 [cit. 2022-11-26]. Dostupné z: https://didisoft.com/2013/06/27/difference-ecc-keys-aes-256/

[15] What is TLS (Transport Layer Security)?. In: *CloudFlare* [online]. © 2022 Cloudflare, Inc. [cit. 2022-11-17]. Dostupné z: https://www.cloudflare.com/learning/ssl/transport-layer-security-tls/

[16] TLS Basics. In: *Internet Society* [online]. © 2022 Internet Society [cit. 2022-11-17]. Dostupné z: https://www.internetsociety.org/deploy360/tls/basics/

[17] What happens in a TLS handshake? | SSL handshake. In: *CloudFlare* [online]. © 2022 Cloudflare, Inc. [cit. 2022-11-17]. Dostupné z: https://www.cloudflare.com/learning/ssl/what-happens-in-a-tls-handshake/

[18] SUNNY CLASSROOM. SSL/TLS handshake Protocol. In: *YouTube* [YouTube video]. 2018 [cit. 2022-11-17]. Dostupné z: https://www.youtube.com/watch?v=sEkw8ZcxtFk&ab_channel=SunnyClassroom

[19] Key differences Between TLS 1.2 and TLS 1.3. In: *A10 Networks* [online]. ©2022 A10 Networks, Inc. All rights reserved [cit. 2022-11-17]. Dostupné z: https://www.a10networks.com/glossary/key-differences-between-tls-1-2-and-tls-1-3/

[20] ZECHMEISTER, Jindřich. Blíží se konec podpory TLS 1.0 a 1.1. In: *SSLmarket* [online]. ©ZONER a.s., 2019 [cit. 2022-11-17]. Dostupné z: https://www.sslmarket.cz/blog/blizi-se-konec-podpory-tls-1-0-a-1-1

[21] LUTKEVICH, Ben. Microcontroller (MCU). In: *TechTarget* [online]. © 2000 - 2022 TechTarget All Rights Reserved, 2019 [cit. 2022-11-17]. Dostupné z: https://www.techtarget.com/iotagenda/definition/microcontroller

[22] GHARGE, Pranav. What Is a Microcontroller? – Simply Explained. In: *All3DP* [online]. © 2022 All3DP. All right reserved., 2022 [cit. 2022-11-26]. Dostupné z: https://all3dp.com/2/what-is-a-microcontroller/

[23] The Powerful ESP32 Processor for IoT - Is it ARM based?. In: *ESPBoards* [online]. © 2023 ESPboards.dev - All Rights Reserved., 2023 [cit. 2023-05-20]. Dostupné z: https://www.espboards.dev/blog/is-esp32-arm-based/

[24] CIRCUITSCHOOLS STAFF. What is ESP32, how it works and what you can do with ESP32?. In: *Circuit Schools* [online]. © Copyright 2022, All Rights Reserved | CircuitSchools.com, 2022 [cit. 2022-11-17]. Dostupné z: https://www.circuitschools.com/what-is-esp32-how-it-works-and-what-you-can-do-with-esp32/

[25] BY RAVI TEJA. Getting Started with ESP32 | Introduction to ESP32. In: *Electronics Hub* [online]. Copyright © 2022 Electronicshub.org, 2021 [cit. 2022-11-17]. Dostupné z: https://www.electronicshub.org/getting-started-with-esp32/

[26] Looking for ESP32-S2/S3 Devkit C parts. In: *Fritzing Forum* [online]. 2022 [cit. 2023-05-20]. Dostupné z: https://forum.fritzing.org/t/looking-for-esp32-s2-s3-devkit-c-parts/15266/2

[27] ESP32-S3 Series Datasheet. In: *Espressif* [online]. Espressif Systems Copyright © 2023, 2022 [cit. 2023-05-20]. Dostupné z: https://www.espressif.com/sites/default/files/documentation/esp32-s3_datasheet_en.pdf

[28] ESP vývojové desky. In: *LáskaKit* [online]. Copyright 2022 LaskaKit. Všechna práva vyhrazena [cit. 2022-12-08]. Dostupné z: https://www.laskakit.cz/esp/

[29] What ESP32 to buy & use? ESP32 S2,S3,C3,C6,H2... In: *YouTube* [YouTube video]. 2022 [cit. 2022-12-08]. Dostupné z: https://www.youtube.com/watch?v=MEhoZ--nOgw&ab_channel=ChipInDetail

[30] Fritzing_Parts. In: *SparkFun Electronics* [online]. Copyright (c) 2016 SparkFun Electronics, 2023 [cit. 2023-05-20]. Dostupné z: https://github.com/sparkfun/Fritzing_Parts/tree/main

[31] IoT ESP-WROOM-32 2.4GHz Dual-Mode WiFi+Bluetooth rev.1, CP2102. In: *Láskakit* [online]. Copyright 2022 LaskaKit. Všechna práva vyhrazena. [cit. 2022-12-07]. Dostupné z: https://www.laskakit.cz/iot-esp-32s-2-4ghz-dual-mode-wifi-bluetooth-rev-1--cp2102/#relatedFiles

[32] Overview of ESP32 features. What do they practically mean?. In: *Explore Embedded* [online]. [cit. 2022-11-17]. Dostupné z: https://www.exploreembedded.com/wiki/Overview_of_ESP32_features._What_do_they_practically_mean%3F

[33] Rozhraní I²C a displej LCD s ESP32. In: *Fyzikální kabinet FyzKAB* [online]. © 2001–2011, 2016–2022 Fyzikální kabinet FyzKAB [cit. 2022-11-17]. Dostupné z: http://kabinet.fyzika.net/ESP32/ESP32-LCD/ESP32-a-I2C-LCD.php

[34] ATECC608A. In: *Microchip* [online]. © Copyright 1998-2022 Microchip Technology Inc. All rights reserved. [cit. 2022-11-17]. Dostupné z: https://www.microchip.com/en-us/product/atecc608a

[35] SOIC. In: *Amkor Technology* [online]. © Copyright 2022 Amkor Technology [cit. 2022-12-05]. Dostupné z: https://amkor.com/packaging/leadframe/soic-2/

[36] Flat no-leads package. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2022 [cit. 2022-12-05]. Dostupné z: https://en.wikipedia.org/wiki/Flat_no-leads_package

[37] BIGNALET, Xavier. Build Confidence in Security with Microchip. In: *Microchip* [online video]. © Copyright 1998-2022 Microchip Technology Inc. All rights reserved. [cit. 2022-12-05]. Dostupné z: https://media.microchip.com/6/7/2/672ccf7bbb7269007824058dfe5c0d8b.mp4

[38] ATSHA204A. In: *Microchip* [online]. © Copyright 1998-2022 Microchip Technology Inc. All rights reserved. [cit. 2022-12-07]. Dostupné z: https://www.microchip.com/en-us/product/ATSHA204A

[39] AT97SC3205T. In: *Microchip* [online]. © Copyright 1998-2022 Microchip Technology Inc. All rights reserved. [cit. 2022-12-07]. Dostupné z: https://www.microchip.com/en-us/product/AT97SC3205T

[40] OPTIGA™ Trust. In: *Infineon* [online]. © 1999 - 2022 Infineon Technologies AG [cit. 2022-12-07]. Dostupné z: https://www.infineon.com/cms/en/product/security-smart-card-solutions/optiga-embedded-security-solutions/optiga-trust/

[41] OPTIGA™ TRUST M SLS32AIA. In: *Infineon* [online]. © 1999 - 2022 Infineon Technologies AG [cit. 2022-12-07]. Dostupné z: https://www.infineon.com/cms/en/product/security-smart-card-solutions/optiga-embedded-security-solutions/optiga-trust/optiga-trust-m-sls32aia/

[42] MAX66301. In: *AnalogDevices* [online]. ©1995 - 2022 Analog Devices, Inc. All Rights Reserved [cit. 2022-12-07]. Dostupné z: https://www.analog.com/en/products/max66301.html#product-overview

[43] DS28E40. In: *AnalogDevices* [online]. ©1995 - 2022 Analog Devices, Inc. All Rights Reserved [cit. 2022-12-07]. Dostupné z: https://www.analog.com/en/products/ds28e40.html

[44] Product data sheet 504934. In: *Mouser Electronics* [online]. © NXP B.V. 2022. All rights reserved, 2022 [cit. 2022-12-07]. Dostupné z: https://cz.mouser.com/datasheet/2/302/SE050_DATASHEET-1620446.pdf

[45] CryptoAuthentication™ Family. In: *Microchip* [online]. © Copyright 1998-2022 Microchip Technology Inc. All rights reserved. [cit. 2022-11-19]. Dostupné z: https://www.microchip.com/en-us/products/security/security-ics/cryptoauthentication-family

[46] Compare ATECC508A and ATECC608B. In: *Microchip* [online]. © Copyright 1998-2022 Microchip Technology Inc. All rights reserved. [cit. 2022-12-05]. Dostupné z: https://www.microchip.com/en-us/product-comparison.atecc508a.atecc608b#

[47] ATECC608B: zSummary Datasheet. In: *Microchip* [online]. © 2020-20021 Microchip Technology [cit. 2022-12-05]. Dostupné z: https://ww1.microchip.com/downloads/aemDocuments/documents/SCBU/ProductDocuments/DataSheets/ATECC608B-CryptoAuthentication-Device-Summary-Data-Sheet-DS40002239B.pdf

[48] BOOMER, James. Migrating from the ATECC608A to the ATECC608B. In: *Microchip* [online]. © 2020 Microchip Technology Inc, 2020 [cit. 2023-05-13]. Dostupné z: https://ww1.microchip.com/downloads/en/Appnotes/Migrating-from-the-ATECC608A-to-the-ATECC608B-DS40002237A.pdf

[49] Fully Customizable Secure Element ATECC608B. In: *Microchip* [online]. © Copyright 1998-2022 Microchip Technology Inc. All rights reserved. [cit. 2022-12-05]. Dostupné z: https://www.microchip.com/en-us/product/atecc608b

[50] *AWS Amazon* [online]. [cit. 2023-05-08]. Dostupné z: https://aws.amazon.com/

[51] ADMIN. Beginners guide on PKI, Certificates, Extensions, CA, CRL and OCSP. In: *GoLinuxCloud* [online]. Copyright © 2023 | Hosted On Rocket.net [cit. 2023-05-08]. Dostupné z: https://www.golinuxcloud.com/tutorial-pki-certificates-authority-ocsp/

[52] CORBETT, Nick. Understanding the AWS IoT Security Model. In: *AWS Amazon* [online]. © 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved., 2017 [cit. 2023-05-08]. Dostupné z: https://aws.amazon.com/blogs/iot/understanding-the-aws-iot-security-model/

[53] *Azure* [online]. [cit. 2023-05-08]. Dostupné z: https://portal.azure.com/

[54] *Google IoT Core* [online]. [cit. 2023-05-08]. Dostupné z: https://cloud.google.com/iot-core

[55] Fritzing parts. In: *Fritzing* [online]. [cit. 2023-05-08]. Dostupné z: https://fritzing.org/parts/

[56] *ArduinoECCX08* [online]. [cit. 2023-05-08]. Dostupné z: https://github.com/arduino-libraries/ArduinoECCX08

[57] *Esp-cryptoauthlib* [online]. [cit. 2023-05-08]. Dostupné z: https://github.com/espressif/esp-cryptoauthlib

[58] KELLTON. How IoT Protocols and Standards Support Secure Data Exchange in the IoT Ecosystem?. In: *Kellton* [online]. © 2023 Kellton, 2023 [cit. 2023-05-08]. Dostupné z: https://www.kellton.com/kellton-tech-blog/internet-of-things-protocols-standards

[59] IT Explained: MQTT. In: *Paessler* [online]. ©2023 Paessler AG [cit. 2023-05-08]. Dostupné z: https://www.paessler.com/it-explained/mqtt

[60] PEARSON, Bryan. *Esp-aws-iot* [online]. [cit. 2023-05-08]. Dostupné z: https://github.com/PBearson/esp-aws-iot

[61] Device communication protocols. In: *Device communication protocols* [online]. © 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved. [cit. 2023-05-08]. Dostupné z: https://docs.aws.amazon.com/iot/latest/developerguide/protocols.html

[62] Regions, Availability Zones, and Local Zones. In: *AWS User Guide* [online]. © 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved. [cit. 2023-05-08]. Dostupné z: https://docs.aws.amazon.com/iot/latest/developerguide/protocols.html

[63] *Esp-idf-lib* [online]. [cit. 2023-05-08]. Dostupné z: https://github.com/UncleRus/esp-idf-lib

[64] FRAJDL, Martin. CO2 čidla MH-Z19B a MH-Z14. In: *Láska Kit* [online]. LaskaKit, 2021 [cit. 2023-05-08]. Dostupné z: https://blog.laskakit.cz/co2-cidla-mh-z19b-a-mh-z14/

Seznam symbolů a zkratek

Zkratky:

FEKT	Fakulta elektrotechniky a komunikačních technologií
VUT	Vysoké učení technické v Brně
RSA	Rivest, Shamir, Adleman
ECC	Elliptic Curve Cryptography
AES	Advanced Encryption Standart
DES	Data Encryption Standard
SP	Substitučně Permutační
NÚKIB	Národní úřad pro kybernetickou a informační bezpečnost
SSL	Secure Sockets Layer
TLS	Transport Layer Security
DLT	Distributed Ledger Technology
DH	Diffie Hellman
ECDH	Elliptical Curve Diffie Hellman
ECDSA	Elliptical Curve Digital Signature Algorithm
NSA	National Security Agency
IPSEC	Internet Protocol Security
USA	Spojené státy americké
IoT	Internet věcí
VoIP	Voice over Internet Protocol
TCP/IP	Internet Protocol Suite
ISO/OSI	International Organization for Standardization, Open Systems Interconnection
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VPN	Virtual Private Network
HTTP	Hypertext Transfer Protocol
I/O	Input, output
WiFi	Wireless Fidelity
ARM	Advanced RISC Machines
ESP	Electronic Stability Program
USB	Universal Serial Bus
LED	Light-Emitting Diode
CPU	Central Processing Unit
ROM	Read-Only Memory
RAM	Random-Access Memory
EPROM	Erasable Programmable Read-Only Memory
EEPROM	Electrically Erasable Programmable Read-Only Memory

SRAM	Static Random Access Memory
MB	Mega Bajt
SD	Secure Digital
UART	Universal Asynchronous Receiver-Transmitter
I^2C	Inter-Integrated Circuit
SHA	Secure Hash Algorithm
ATECC	CryptoAuthentication™
SOIC	Small Outline IC
UDFN	Dual Flat No Leads
SMD	Surface Mount Device
API	Application Programming Interface
SWI	Serial Peripheral Interface
GPIO	General-purpose Input/Output
NC	No Connection
GND	Ground
SDA	Serial Data
SCL	Seriál Clock
NVS	Non-volatile Storage
CSR	Certificate Signing Request
CN	Common name
CA	Certificate Authority
JWT	JSON Web Token
GUI	Graphical User Interface
COM	Communication Port
HTTP	Hypertext Transfer Protocol
MQTT	Message Queuing Telemetry Transport
Lora-WAN	Long Range Area Network
ALNP	Application Layer Protocol Nagotiation
TX	Transmit Pin
RX	Recive Pin

Symboly:

U	napětí	(V)

SEZNAM PŘÍLOH

Příloha A - AWS-IoT

Využitý program pro komunikaci a konfiguraci kryptografického koprocesoru a navázání se serverem Amazon AWS se nachází v elektronické příloze.

Příloha B - Generování CSR pomocí Arduino IDE

Využitý program pro generování CSR v Arduino IDE se nachází v elektronické příloze.

Příloha C - Vygenerované certifikáty, Json

Vygenerované certifikáty (potřebné certifikáty pro vygenerování CSR, CSR, Server Root certificate, Device certificate) potřebné pro autentizaci s AWS a zkrácený Json soubor stažený z AWS se nacházejí v elektronické příloze.