VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Rozšíření přístupového systému o nové vlastnosti

Extension of the Access System With New Features

Bc. Lukáš Klein

Diplomová práce

Vedoucí práce: Ing. David Seidl, Ph.D.

Ostrava, 2021

Abstrakt

Diplomová práce se zabývá rozšířením SW pro přístupový systém na VŠB – Technické univerzitě Ostrava, FEI. Přístupový systém je rozšířen o zabezpečenou komunikaci se smart kartami MIFARE DESfire bez použití SAM modulu. Také je vylepšen proces vzdálených aktualizací OTA pro mikrokontrolér ESP32. Dále byly provedeny optimalizace odezvy přístupového systému. Jednou z hlavních optimalizací bylo využití protokolu WebSocket s podporou zabezpečeného spojení prostřednictvím protokolu HTTPS.

Klíčová slova

ESP32, DESfire, MIFARE, Bezpečnost, Přístupový systém, OTA

Abstract

This diploma thesis is about extending existing software for an access system running on on VŠB – Technická univerzita Ostrava, FEI. New features for access system are encrypted communication with smart cards MIFARE DESfire which enables more secure autentification of users. OTA updates for microcontrollers ESP32 are also improved. In order to apply new features to the access system, optimalizations are made. One of the main optimalization is the usage of WebSockets with the support of secure connection via HTTPS protocol.

Keywords

ESP32, DESfire, MIFARE, Security, Access System, OTA

Poděkování

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Davidu Seidlovi, Ph.D. za konzultace a cenné rady při návrhu softwaru, dále také za rady při psaní diplomové práce.

Obsah

Seznam použitých zkratek a symbolů

-O2	–	Režim kompilace v jazyce C pro optimalizaci s prioritou na rychlost aplikace bez ohledu na velikosti aplikace
3DES	–	Triple DES je bloková šifra založená na šifrování DES, které se aplikuje třikrát za sebou, čímž zvyšuje její odolnost proti prolomení
ACK	–	Acknowledgement Code, jsou řídící znaky přenosu užívané k označení toho, že vyslaná zpráva byla přijata nepoškozená nebo bez chyb
AES	–	Advanced Encryption Standard je standardizovaný algoritmus používaný k šifrování dat v informatice. Jedná se o symetrickou blokovou šifru šifrující i dešifrující data stejným klíčem. Data jsou rozdělená do bloků pevně dané délky.
AID	–	Aplikační identifikátor ve standardu ISO/IEC 7816-4
APDU	–	Application protocol data unit je komunikační protokol mezi čtečkou karet a smart kartou
API	–	Application Programming Interface označuje v informatice rozhraní pro programování aplikací
ASK modulace	–	Amplitude-shift keying je druh modulace pomocí změn amplitudy nosného signálu
ATQA	–	Answer To re-Quest type A je odpověď smart karty nebo jiného objektu s vazbou na blízko definovaná ve standardu ISO/IEC 14443
ATR	–	Answer To Reset je zpráva smart karty dle standardu ISO/IEC 7816 po restartu smart karty
ATS	–	Answer To Select je potvrzení výběru dané smart karty pro komunikaci ve standardu ISO/IEC 14443, obdoba ATR
BI	–	Business intelligence jsou strategie a technologie využité v podnikání pro datovou analýzu
BLE	–	Bluetooth low energy je standard pro bezdrátovou komunikaci propojující dvě a více elektronických zařízení

CAN – Controller Area Network je typ sběrnice, využívaná nejčastěji pro vnitřní komunikační síť senzorů a funkčních jednotek v automobilech

CMAC – Cipher-based Message Authentication Code odpovídá algoritmu OMAC1

CMake – CMake je multiplatformní volně šiřitelný software pro automatizaci překladu programu v různých operačních systémech

CRC – Cyklický redundantní součet je hašovací funkce, používaná k detekci chyb během přenosu či ukládání dat

DES – Data Encryption Standard je symetrická šifra v kryptografii vyvinutá v 70. letech 20. století

DHCP – Dynamic Host Configuration Protocol je internetový protokol pro automatickou konfiguraci počítačů připojených do počítačové sítě

DNS – Domain Name System je hierarchický, decentralizovaný systém doménových jmen

DoS – Denial of service je typ útoku, jehož cílem je znepřístupnit službu ostatním uživatelům

DPH – Daň z přidané hodnoty

DTSL – Datagram Transport Layer Security zajišťuje podobné zabezpečení jako TSL, ale pro datagramové protokoly jako je UDP

E-INK – Zobrazovací zařízení, které odráží světlo jako normální papír a je schopné uchovat text i obrázky natrvalo bez spotřeby elektřiny

EAL – Evaluation Assurance Level je číselné ohodnocení úrovně jistoty bezpečnosti systému podle mezinárodní standardu Common Criteria security evaluation ISO/IEC 15408

ECC – Kryptografie eliptických křivek (Elliptic-curve cryptography) je metoda šifrování veřejných klíčů založená na algebraických strukturách eliptických křivek nad konečnými tělesy.

eFUSE – Elektronická pojistka umožňující dynamické real time přeprogramování čipu

ESP-IDF – Oficiální vývojový framework pro mikrokontroléry ESP32

ESP32 – ESP32 je řada nízkonákladových mikrokontrolérů s malou spotřebou a s integrovanou technologií Wi-Fi a Bluetooth

FEI – Fakulta elektrotechniky a informatiky

Framework – Je vývojové prostředí, struktura pro vývoj a programování SW projektů

Git – Je v informatice distribuovaný systém správy verzí kódu

HA režim – High Availability je režim pro vysokou dostupnost. Zajišťuje dostupnost bez výpadků

HCE – Host card emulation je softwarová architektura, která de facto nahrazuje elektronické průkazy totožnosti pouze pomocí softwaru

HDLC – Je komunikační protokol linkové vrstvy vycházející z protokolu Synchronous Data Link Control

HMAC – Keyed-hash Message Authentication Code je MAC počítaný s použitím kryptografické hašovací funkce v kombinaci s tajným šifrovacím klíčem

HTTP – Hypertext Transfer Protocol je internetový protokol určený pro komunikaci s WWW servery.

HTTPS – Hypertext Transfer Protocol je internetový protokol určený pro komunikaci s WWW servery pomocí zabezpečené komunikace s protokolem TSL

Hz – Hertz je jednotkou frekvence v soustavě SI

I2C – Inter-Integrated Circuit je multi-masterová počítačová sériová sběrnice vyvinutá firmou Philips

ID – Identifikátor, identifikační značka ve výpočetní technice

INS – Typ instrukce APDU ve standardu ISO/IEC 7816-4

IP – Internet Protocol je základní protokol pracující na síťové vrstvě používaným v počítačových sítích a Internetu

IRQ – Interrupt ReQuest je signál, kterým požádá zařízení procesor o pozornost (požádá o přerušení)

IV – Inicializační vektor se používá pro definici počátečního stavu kryptografické funkce

JS – JavaScript je multiplatformní, objektově orientovaný, událostmi řízený skriptovací jazyk

JSON – JavaScript Object Notation je datový formát pro přenos dat vycházející z jazyka JavaScript

LAN – Local area network je termín používaný pro lokální sítě

LDAP – Lightweight Directory Access Protocol je protokol pro ukládání dat a přístup k datům na adresářovém serveru ve stromové struktuře, jedná se o adresářovou informační službu se zajištěným ověřováním přístupu

LSFR – Linear-feedback shift register je posuvný register, kde vstupní bit je lineární funkcí předchozích stavů

lwIP – Lightweight IP je open-source TCP/IP stack určený pro vestavěné systémy

MAC – Message authentication code je funkce zajišťující jak integritu, tak autentizaci přenášených zpráv

Make – Je utilita pro automatizaci překladu zdrojových kódů do binárních souborů

MbedTLS – Mbed TLS je implementace protokolů TLS a SSL určená pro vestavěné systémy

MD5 – Je v kryptografii skupina hašovacích funkcí, která z libovolného vstupu dat vytváří výstupní data fixní délky

NDA – Non-disclosure agreement je typ smlouvy používající se pro zachování obchodního tajemství

NETIF – Je v frameworku ESP-IDF abstraktní vrstva mezi aplikací a TCP/IP stackem

NFC – Near Field Communication je modulární technologie rádiové bezdrátové komunikace mezi elektronickými zařízeními na velmi krátkou vzdálenost

NFCIP-1 – Near Field Communication Interface and Protocol-1 je rozšířením standardu ISO/IEC 14443

NIST – Národní institut standardů a technologie je laboratoř měřicích standardů při ministerstvu obchodu USA

NodeJS – Prostředí pro vykonávání aplikací v jazyce Javascript používané například pro webové servery

NVS – Non-volatile storage je úložiště, které uchovává informaci i po odpojení napájení

OMAC – One-key MAC je MAC vytvořený z blokové šifry

OOK – On-off keying je forma ASK modulace, která reprezentuje data pomocí přítomnosti nebo nepřítomnosti nosného signálu

OTA update – Over the air update je vzdálená distribuce nové verze firmware bez nutnosti fyzické přítomnosti u zařízení

PHP – Je programovací skriptovací jazyk používaný především pro dynamické internetové stránky

PICC – Proximity cards or objects jsou karty nebo jiné objekty s možností komunikace na blízko

PING – Typ zprávy v protokolu Websocket pro ověření, že spojení s druhou stranou není přerušeno

PN512 – Je vysoce integrovaný vysílač-přijímač pro bezkontaktní komunikaci NFC

PN532 – Modul typu transceiver (vysílač i přijímač) pro bezdrátovou komunikaci pracující na frekvenci 13,56 MHz

PONG – Odpověď v protokolu Webscoket pro ověření, že spojení není přerušeno

QR kód – Quick Response kód je typ maticového čárového kódu

RAM – Random access memory jsou polovodičové paměti s přímým přístupem procesoru umožňující čtení i zápis bez zachování dat při přerušení napájení

Ratchet – Knihovna pro jazyk PHP umožňující práci s protokolem WebSocket

RATS – Request for Answer To Select je výzva k výběru dané smart karty ve standardu ISO/IEC 14443

REQA – Request command je příkaz k aktivaci smart karet ve standardu ISO/IEC 14443

RF – Radio frequency je v elektronice a radiotechnice termín používaný nejčastěji pro kmitočty nad oblastí slyšitelných zvukových vln

RFID – Radio-frequency identification využívá elektromagnetické pole k identifikaci a sledování objektů

ROL – Operace otočení bitů v proměnné doleva

ROM – Read only memory je paměť určená pouze ke čtení

RS422 – Standard sériové komunikace určující elektrické vlastnosti digitálních obvodů, pomocí využití rozdílu potenciálů mezi vodiči

RSA – Je šifra s veřejným klíčem pro zabezpečený přenos

SAK – Select AcKnowledge je potvrzení výběru smart karty při antikolizní smyčce ve standardu ISO/IEC 14443

SAM – Secure access module je modul pro zabezpečený přístup. Obsahuje zabezpečenou paměť a podporuje kryptografické operace

SHA-2 – Secure Hash Algorithm je rozšířená hašovací funkce, která vytváří ze vstupních dat výstupní data fixní délky o velikosti výstupu nejméně 224 bitů.

SIM – Subscriber identity module je účastnická identifikační karta sloužící k identifikaci účastníka v mobilní telefonní síti

SIP – Session Initiation Protocol je internetový protokol určený pro přenos signalizace v internetové telefonii

SNTP – Simple Network Time Protokol je protokol pro synchronizaci času

SPI – Serial Peripheral Interface je sériové periferní rozhraní. Používá se pro komunikaci mezi řídícími mikroprocesory a ostatními integrovanými obvody

SPIFFS – Je souborový systém pro flash zařízení typu NOR (skládá se z hradel negovaného logického součtu) pomocí rozhraní SPI

SSL – Předchůdce protokolu TSL, který poskytuje zabezpečení komunikace šifrováním a autentizaci komunikujících stran

TCP	–	Transmission Control Protocol je protokol transportní vrstvy v sadě TCP/IP používaný v síti internet
TSL	–	Transport Layer Security je protokol a nástupce protokolu SSL umožňující zabezpečenou komunikaci v síti internet
UART	–	Universal Asynchronous Receiver-Transmitter je počítačová sběrnice sloužící k asynchronnímu sériovému přenosu dat, přičemž formát a rychlost tohoto přenosu jsou konfigurovatelné
UDP	–	User Datagram Protocol je internetový protokol používající datagramy bez záruky doručení
UID	–	Je unikátní identifikační značka
URL	–	Uniform Resource Locator je řetězec znaků, který definuje přesnou specifikaci umístění zdrojů informací na internetu
VPN	–	Virtual private network je prostředek pro vytvoření privátní sítě prostřednictvím nedůvěryhodné počítačové sítě
WPA2	–	Standard IEEE 802.11i je dodatek standardu IEEE 802.11 vylepšující autentizační a šifrovací algoritmus pro bezdrátové sítě Wi-Fi
WPA3	–	Náhrada standardu WPA2, který zvyšuje úroveň zabezpečení
WS	–	Protokol WebSocket je plně duplexní internetový protokol přes TCP spojení
WSS	–	WebSocket Secure je protokol WebSocket se zabezpečenou komunikaci pomocí protokolu TSL

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

Kapitola 1

Úvod

Tato diplomová práce se dle zadání zabývá problematikou rozšíření přístupového systému na Fakultě elektrotechniky a informatiky VŠB – Technické univerzitě Ostrava. Hlavním záměrem je rozšířit stávající přístupový systém o funkcionalitu vyčítání dat ze zabezpečených bloků smart karet typu MIFARE DESfire a tím zásadně zvýšit bezpečnost celého systému při ověřování uživatele. Další částí této práce je vylepšení procesu aktualizací modulů přístupového systému a to tak, aby byl průběh bezpečnější s minimalizací množství chyb při aktualizaci, které by mohly ovlivnit možnost vzdáleně aktualizovat firmware jednotlivých modulů.

Hlavním tématem práce je vývoj nového SW v části aplikace a implementace úprav existujícího zdrojového kódu tak, aby bylo nově možné číst data z karet typu MIFARE DESfire. Data z těchto karet jsou zašifrovaná, a proto je nutné použít zabezpečenou komunikaci s těmito kartami. Důležitým přínosem této změny bude zamezení možnosti kopírovat nebo napadnout nově používané karty. Načtená data z karet typu MIFARE DESfire je dále nutno zabezpečeně zpracovat a uživatele ověřit a ve výsledku povolit nebo zamítnout vstup do místnosti.

V menším rozsahu se práce zabývá optimalizací procesu vzdálených aktualizací modulů a to tak, aby se odstranily problémy, které již v průběhu provozování přístupového systému nastaly. Je navržen nový způsob práce s aktualizacemi modulů, který minimalizuje možnost výskytu chyb, které byly zjištěné při provozu stávajícího přístupového systému.

Práce je rozdělena na čtyři části. První část předkládá teoretický základ práce. V další části je navržen zlepšený proces vzdálených aktualizací modulů. Ve třetí části je řešena základní a nejnutnější implementace ověření uživatele pomocí zabezpečených dat ze smart karet typu MIFARE DESfire. Poslední část práce je zaměřená na optimalizace odezvy celého systému a analýzu naměřených dat, aby bylo nové změny možné nasadit do reálného provozu.

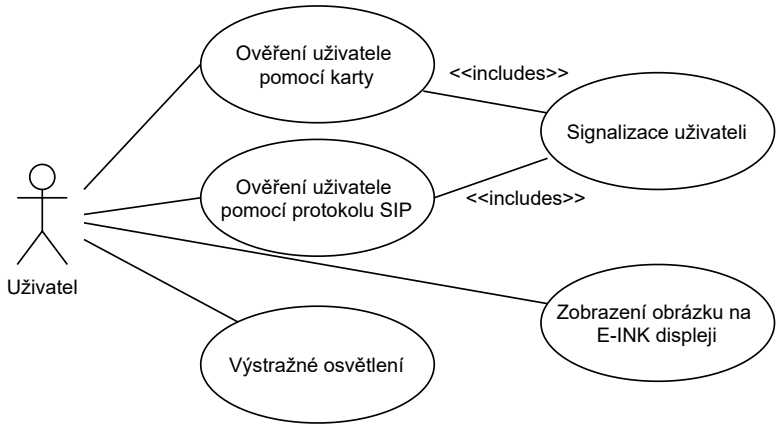
Kapitola 2

Teoretický základ

Tato část práce má za cíl představit použité technologie, algoritmy, programy a postupy, a tak uvést do problematiky, kterou se zabývá tato práce.

2.1 Stav přístupového systému před změnami

Současný přístupový systém je nasazen na univerzitě VŠB - Technické univerzitě Ostrava od roku 2019. Je nainstalován na více jak 200 místnostech a v pracovní dny proběhne více než 2000 žádostí o přístup. Systém nahradil předchozí systém používající karty založené na technologii RFID, se kterým byl současný přístupový systém zpětně kompatibilní. Přístupový systém má, jak je jasné již z názvu, za hlavní úkol povolovat nebo odmítat přístup uživatelům do místnosti a to pomocí přístupové karty nebo prozvoněním definovaného telefonního čísla. Zbývající funkce, jako například zobrazení informací na E-INK displeji či doplňkové výstražné upozornění v případě požáru nebo jiné události, jsou nad rámec hlavního účelu systému. Tyto funkcionality jsou zachyceny na diagramu užití obrázek 2.1.

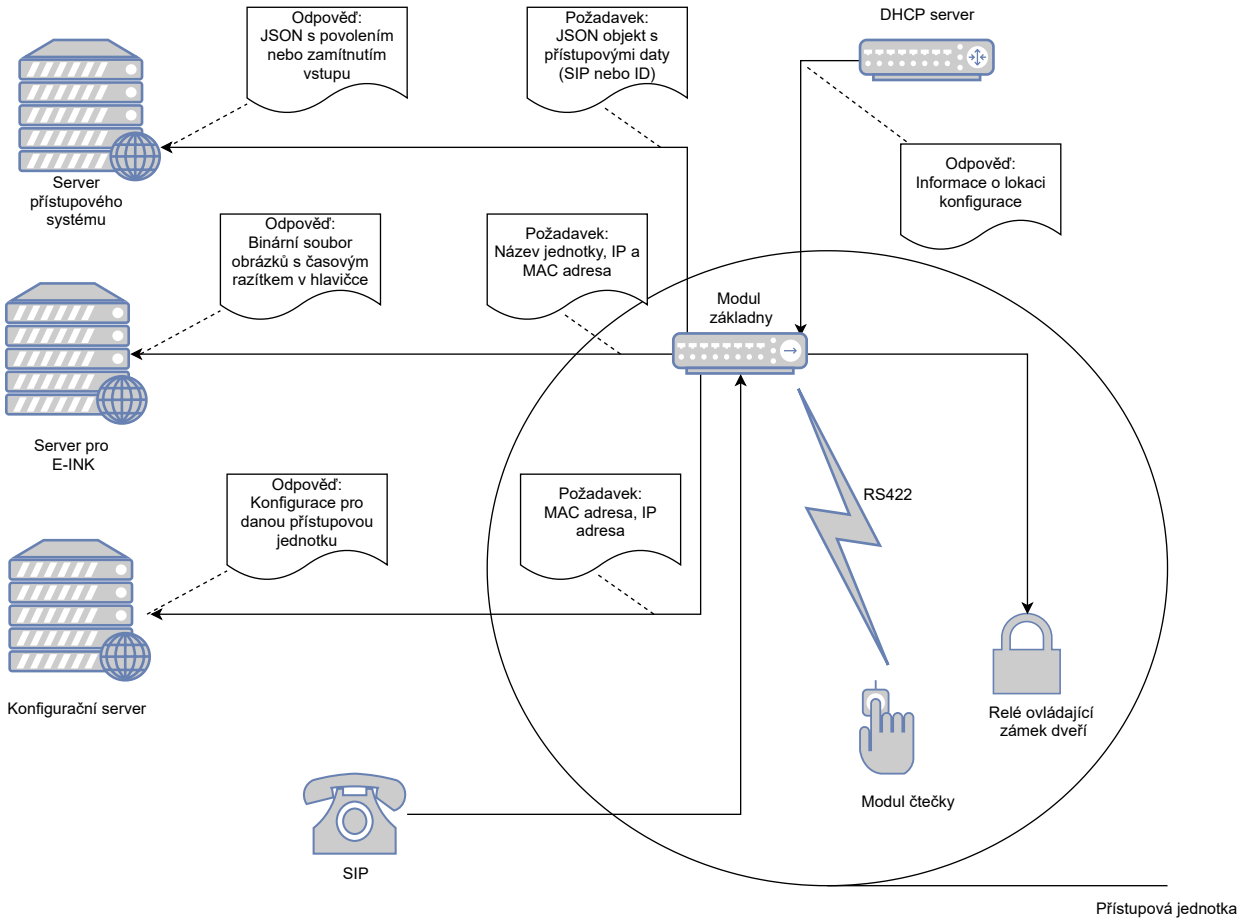


Obrázek 2.1: Use case diagram přístupového systému

Přístupový systém má tři hlavní komponenty:

- Server přístupového systému
- Přístupové jednotky
  - Modul základny
  - Modul čtečky

Přehled celého systému je zachycen na obrázku 2.2, kde je také naznačeno jaký druh informací si jednotlivé části vyměňují. Je zde taktéž zachycen konfigurační server, který je však totožný se serverem pro přístupový systém. Není to ale nutně vyžadováno a lze je takto oddělit.



Obrázek 2.2: Přehled přístupového systému

2.1.1 Přístupová jednotka

Přístupová jednotka se skládá ze dvou samostatných modulů komunikujících mezi sebou pomocí sběrnice RS422. Posláním prvního modulu, jenž je nazvaný základna, je komunikace se síťovými servery sítě LAN. Interakci s uživatelem a komunikaci s kartami zajišťuje modul nazvaný čtečka. Důležitá rozhraní, které modul základny obsahuje a využívá, jsou Ethernet, spínací relé a RS422. Modul čtečky používá sběrnici typu RS422, piezoměnič, NFC čtečku PN532 a E-INK displej podsvícený RGB LED diodami.

Oba moduly jsou vyvinuty s použitím mikrokontroléru ESP32 vyráběném firmou Espressif Systems (SHANGHAI) CO. LTD.. Mikrokontrolér ESP32 disponuje dvou-jádrovým procesorem Tensilica Xtensa LX6 s frekvencí až 240 MHz, velikost paměti RAM je 520 KB. Mikrokontrolér ESP32 je osazen třiceti čtyřmi všeobecně použitelnými piny a dalšími periferiemi jako jsou Ethernet, SPI, I2C, UART, ale i CAN. Mikrokontrolér ESP32 podporuje hardwarovou akceleraci pro kryptografické algoritmy AES, SHA-2, RSA a ECC. [1]

Software obou modulů je vyvinut na open-source frameworku ESP-IDF ve verzi 3.3 long term, který je vyvíjen a podporován výrobcem mikrokontroléru ESP32, firmou Espressif Systems. Framework ESP-IDF využívá real-time operačního systému FreeRTOS, který je upraven pro podporu dvou jader procesoru. [2]

Pro komunikaci mezi moduly je navržen vlastní bajtově orientovaný protokol. Pro potvrzování zpráv se používá jednotlivé potvrzování (stop and wait). Každý paket má svou adresu, která původně odpovídala jednotlivé periferii modulu čtečky.

Oba moduly využívají společný kód pro komunikaci pomocí vlastního protokolu na rozhraní UART (Universální asynchronní přijímač a vysílač). Tento kód obsahuje definice a tvorbu zpráv, zpracování paketů a jejich posílání a potvrzování. Dále také sdílí kód pro ovládání piezoměniče pro zvukovou signalizaci. Moduly využívají nezávislé úlohy (task), které zajišťují multitasking. Je také hojně využito synchronizačních mechanismů, jako jsou semafory a fronty. Fronty se například využívají k rozdělování zpráv přijatých z modulu základny, ale také pro řazení zpráv modulu čtečky z jednotlivých úloh. Semafory se používají k přístupu k UART, který není „thread safe“, nebo k spouštění melodií prostřednictvím piezoměniče, kdy je nutné, aby hrála v jeden čas pouze jedna melodie. Synchronizace úloh je nutná, protože úlohy mohou být spuštěny na jakémkoliv ze dvou jader CPU, a je nutné řešit možné souběhy v programu.

2.1.1.1 Stav modulu čtečky před úpravami

V modulu čtečky je spuštěno několik důležitých úloh, které jsou spuštěny po celou dobu běhu programu. Inicializace programu modulu čtečky proběhne vytvořením těchto úloh. Poté již čeká na vnitřní nebo vnější impulzy k činnosti a reaguje na ně. Stále běžící úlohy jsou tyto:

- Úloha pro čtení zpráv z modulu základny:

  - Zajišťuje vykonávání příkazů, nebo použití dat přicházejících z modulu základny
  - Může se jednat o příkazy pro zvukovou nebo světelnou signalizaci, OTA aktualizace, vykreslení textu nebo obrázků na E-INK displej
- Úloha pro čtení RFID karet pomocí modulu RMD6300
  - Dále nepodporovaná úloha pro zajištění kompatibility čtení předchozího typu karet RFID
  - Zajišťuje čtení dat z karet RFID na frekvenci 125 kHz
  - Posílá přečtená data modulu základny
- Úloha zajišťující obsluhu E-INK displeje
  - Zajišťuje inicializaci a komunikaci E-INK displeje
  - Provádí periodické dotazy na nový obrázek k vykreslení
- Úloha pro čtení karet s technologií NFC pomocí modulu PN532
  - Komunikuje s modulem PN532
  - Načítá ID přiložené karty, který posílá na modul základny

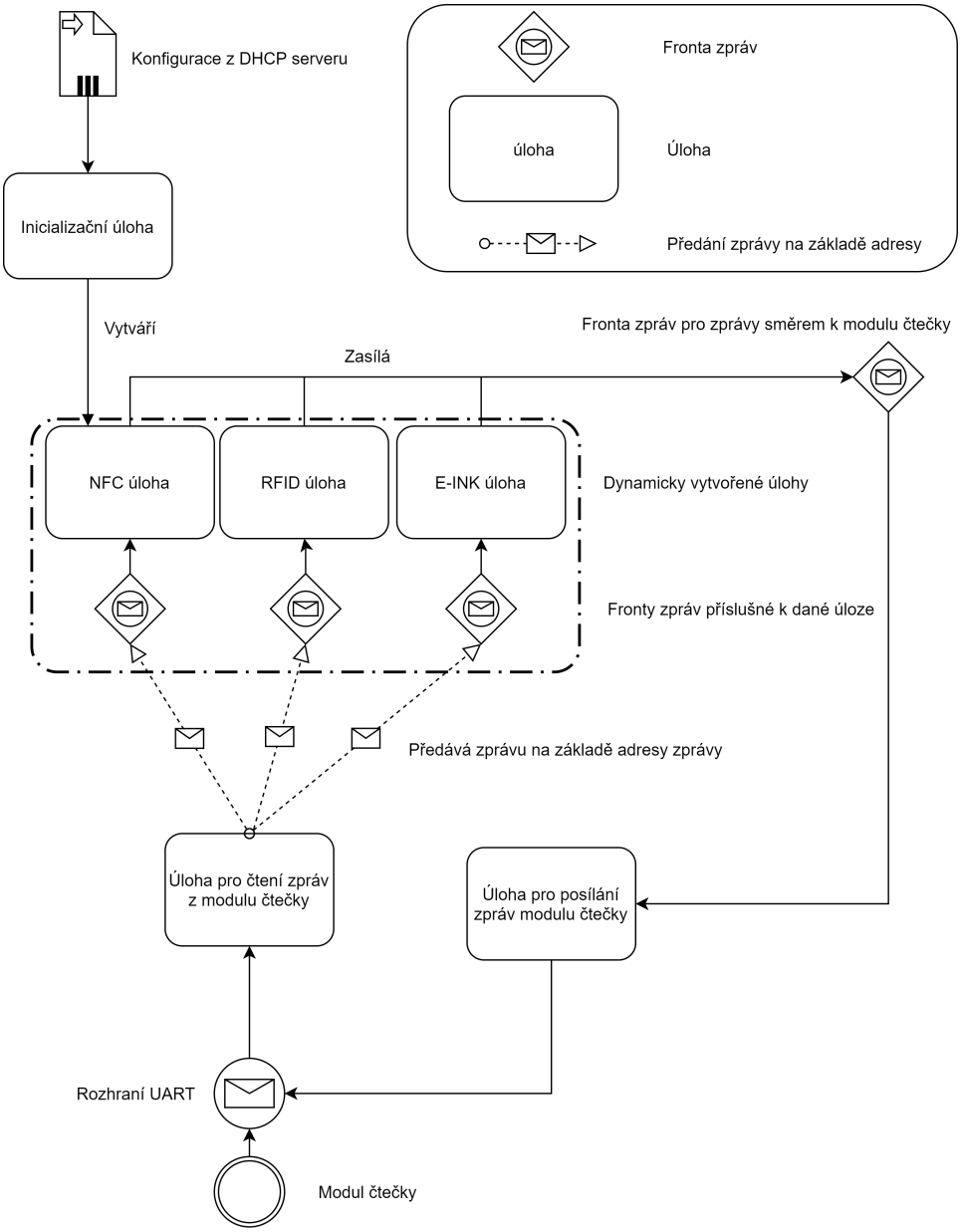
2.1.1.2 Stav modulu základny před úpravami

Úlohy jsou vytvářeny dynamicky na základě konfigurace načtené z přístupového serveru ihned po startu. Aby bylo možné rozdělovat zprávy odpovídajícím úlohám a úlohy měly synchronizovanou možnost odesílat zprávy modulu čtečky, tak jsou použity fronty zpráv. Každá dynamicky vytvořená úloha má svou frontu zpráv, kterou přijímá data z modulu čtečky. Pro obsluhu těchto dvou front zpráv má modul základny dvě stále běžící úlohy. Jedna z nich je pro posílání zpráv modulu čtečky, které vybírá z fronty zpráv určené pro zasílání dat modulu čtečky. Druhá úloha, jež čte zprávy z modulu čtečky, je po načtení roztřiďuje dle adresy zprávy do fronty zpráv dané úlohy, která je registrována, že tento typ zprávy obsluhuje. Použití front zpráv zobrazuje schéma na obrázku 2.3.

Modul základny při inicializaci získá z DHCP kromě IP adresy také informaci o tom, kde se nachází konfigurační soubor. Tato informace je přenášena v polích next-file a next-server, které se obvykle využívají k vzdálenému bootování ze sítě. Konfigurace ve formátu JSON obsahuje informace o verzi, lokaci OTA aktualizací, SIP přihlašovacích údajích a důležitou informaci o tom, jaké úlohy k jakým periferiím mají být spuštěny. Před změnami obsaženými v této práci mohly být pouze následujících typů:

- TCP obecná úloha
- TCP E-INK úloha, která podporovala fragmentaci zpráv
- HTTP obecná úloha

- HTTP úloha obdoba TCP E-INK úlohy
- SIP úloha



Obrázek 2.3: Přehled dynamicky vytvořených úloh a jejich front

Po načtení konfigurace se modul základny pokusí o OTA aktualizaci modulu čtečky, pokud to indikuje číslo verze v konfiguraci. Podobná strategie se využívá při verzování zónových souborů v systému DNS. Poté se dle potřeby provede OTA aktualizace modulu základny. Následně se na základě konfigurace vytvoří definované úlohy. Každá úloha poté má svou frontu zpráv, do které

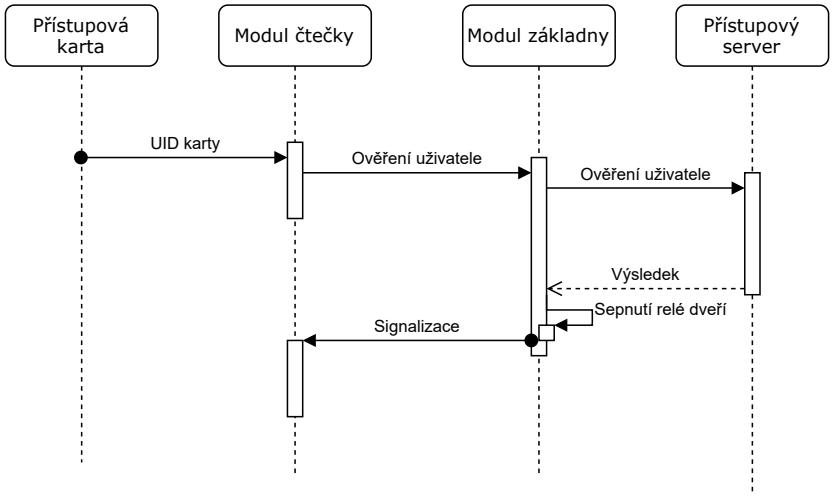
přicházejí zprávy na základě adresy z modulu čtečky. Výjimkou je úloha pro SIP, která nedostává zprávy z modulu čtečky, ale z komponenty projektu modulu základny obsluhující SIP protokol. TCP úlohy se již nepoužívají a jsou nahrazeny HTTP požadavky.

2.1.2 Server přístupového systému

Původní přístupový TCP server je nahrazen novým webovým přístupovým serverem napsaným v jazyce PHP z důvodu jednoduchosti nasazení a možnosti úprav skriptů. Přístupový server přijímá HTTP post požadavky, kdy jsou v URL adrese požadavku obsaženy informace o tom, jaká přístupová jednotka posílá požadavek ke komunikaci. Tento požadavek obsahuje IP adresu, MAC adresu a jméno. V těle požadavku je ID karty, nebo informace o volajícím v případě použití SIP protokolu. Na základě těchto informací PHP skript vyhodnocuje, zda má mít uživatel přístup povolen, nebo odmítnut. Přístupový server vytvoří z ID karty hash, nebo využije telefonní číslo, kterým následně vyhledá uživatele a jeho informace v LDAP. Informace o povoleném nebo zakázaném přístupu uživatele nebo skupin uživatelů do místnosti je obsažena v XML souboru pro danou místnost. Na základě nalezené shody v tomto souboru s informacemi vyhledanými v LDAP poté přístupový server pošle odpovídající odpověď ve formátu JSON.

2.1.3 Proces ověření uživatele

Po přiložení karty k modulu čtečky se přečte unikátní identifikátor (UID) dané karty a zašle se na modul základny. Modul základny poté předá zprávu úloze odpovídající pro danou periferii. Úloha základny odešle UID přístupovému serveru, který vyhodnotí, zda uživatel má povolený přístup. Přístupový server požadavek zpracuje a odešle odpověď. Modul základny na základě odpovědi sepne relé ovládající zámek dveří a pošle modulu základny informaci o tom, jaká signalizace na modulu čtečky se má provést. Tento proces je zachycen na obrázku 2.4.



Obrázek 2.4: Sekvenční diagram původního ověření

2.2 Smart karty

Základním požadavkem rozšíření původního systému bylo použití bezkontaktních smart karet pro ověření uživatele. Smart karta je fyzické elektronické autorizační zařízení určené ke kontrole zdrojů nebo přístupu. Obvykle má tvar plastové karty a obsahuje integrovaný obvod. Smart karty mohou být jak bezkontaktní, tak kontaktní, případně obojí jako například platební karty. Běžně používané SIM karty v mobilních telefonech jsou také smart karty, v tomto případě určené pro identifikaci uživatele mobilním operátorem. Dalším příkladem smart karet jsou Java card, které umožňují běh aplikací napsaných v jazyce Java přímo na smart kartě. [3] [4]

Základní parametry smart karet definují mezinárodní standardy ISO/IEC 7810 a ISO/IEC 7816. Prvně jmenovaný standard ISO/IEC 7810 definuje základní fyzické charakteristiky identifikačních karet. Určuje například velikost smart karty typu ID-1 (85,6 x 53,98 mm), kterou používají platební karty a mnoho ostatních karet. [5]

2.2.1 Kontaktní smart karty

Standard ISO/IEC 7816 je obsáhlejší (15 částí) a definuje důležité aspekty smart karet, které jsou využívány v pozdějších standardech a technologiích. Tento standard byl původně určen pro kontaktní smart karty. Důležitou částí standardu karet je definice protokolu APDU (Aplication Protocol Data Unit) v části standardu ISO/IEC 7816-4, jež definuje strukturu příkazů a odpovědí ke komunikaci se smart kartou. [6]

2.2.2 Bezkontaktní smart karty

Bezkontaktní smart karty jsou definovány standardem ISO/IEC 14443. Jeho obsahem je definice bezkontaktních smart karet na blízkou vzdálenost. Karty obsahují RF anténu a standard popisuje bezdrátovou komunikaci na frekvenci 13,56 MHz. Standard ISO/IEC 14443 však nedefinuje vyšší úrovně komunikačního protokolu jako jsou příkazy a odpovědi smart karty. Z tohoto důvodu bezkontaktní smart karty převážně využívají standard ISO 7816-4, případně vlastní proprietární definice. [7]

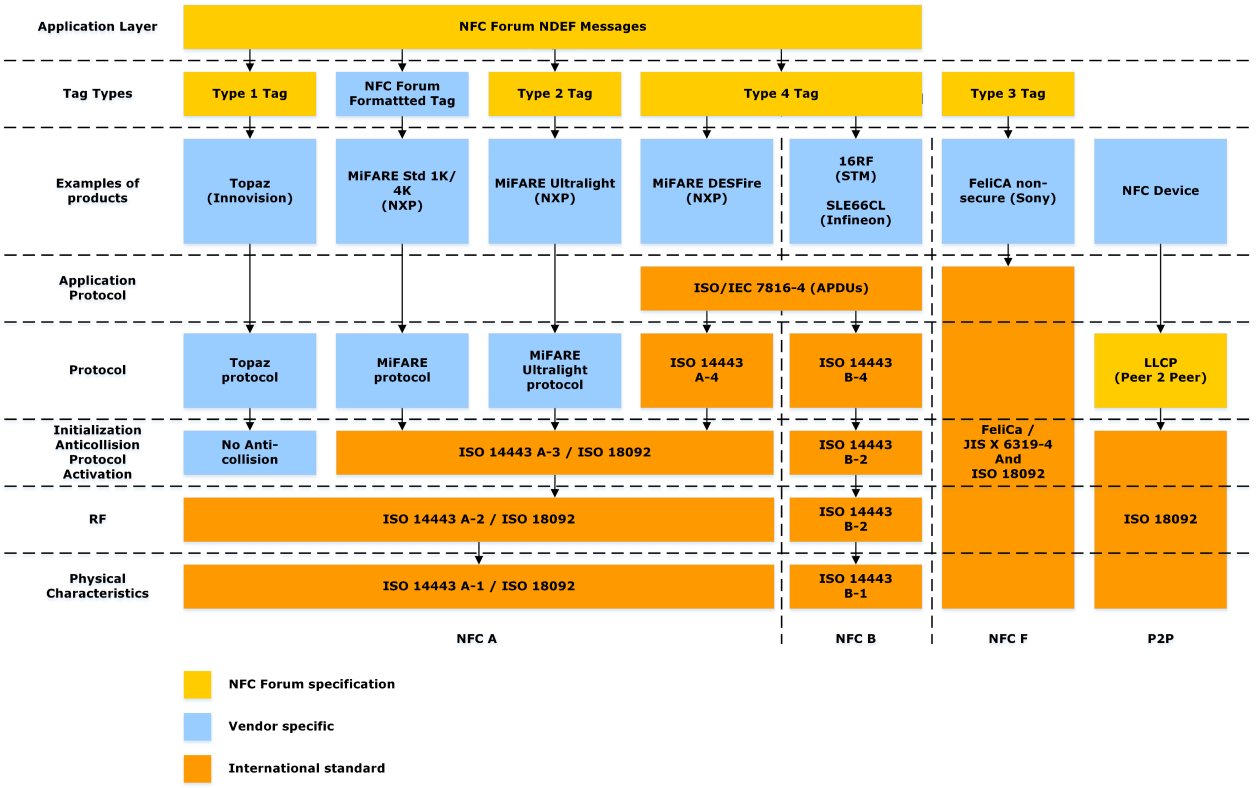
2.2.3 Napájení bezkontaktní karty

Smart bezkontaktní karty jsou obvykle pasivní. Nemají tedy vlastní zdroj elektrické energie. Karta má vestavěnou anténu, pomocí které se, pokud se dostane do blízkosti čtecího zařízení, začne napájet elektrickým proudem díky indukci elektromagnetického pole čtecího zařízení. Dle standardu ISO/IEC 14443 je blízkost definována okolo 10 cm. Napájení karet i komunikace probíhá na frekvenci 13,56 MHz. [8]

2.2.4 Technologie NFC [9]

Technologie NFC je sada technologií pro přenos dat na velmi krátkou vzdálenost (obvykle do 10 cm) s obvyklými přenosovými rychlostmi od 106 kbit/s do 424 kbit/s. Dělí se na pasivní a aktivní podle druhu napájení. Zatímco aktivní mají vlastní zdroj napájení, pasivní jsou napájeny aktivními zařízeními.

Technologie NFC definuje komunikační módy pro Near Field Communication Interface and Protocol (NFCIP 1) za použití zařízení komunikujícího na frekvenci 13,56 MHz. Standard technologie NFC vychází ze standardu ISO/IEC 14443 (definujícího bezkontaktní smart karty pro identifikaci a jejich komunikaci) a je tedy se standardem ISO/IEC 14443 částečně kompatibilní. Standard ISO/IEC 18092 definuje jiný přenosový protokol a tím nahrazuje část 4 standardu ISO/IEC 14443. [10]



Obrázek 2.5: Sada technologií NFC [11]

Technologie NFC má čtyři operační režimy:

- Tag reader/writer – umožňuje číst a zapisovat z informace z tagů
- Peer to peer – umožňuje komunikaci mezi dvěma aktivními NFC zařízeními
- Card emulation – zařízení emulující chování jako smart bezkontaktní karty podle standardu ISO/IEC 14443

- Wireless Charging mode – umožňuje bezdrátové nabíjet s maximálním výkonem 1 Watt

2.3 Standard ISO/IEC 14443

Standard ISO/IEC 14443 obsahuje 4 části [12]:

1. Fyzické charakteristiky
2. Radiofrekvenční výkonové rozhraní a signálové rozhraní
3. Inicializaci a řešení kolizí
4. Přenosový protokol

Bezkontaktní karty (ve standardu nazývané PICC - proximity integrated circuit card) se dělí na typ A a B. Ty se liší metodou modulace, kódovým schématem a inicializačním procesem. Přenosový protokol je naopak pro oba typy stejný. Smart karty použité pro rozšíření vlastností přístupového systému typu MIFARE DESfire jsou typu bezkontaktní karty A, proto bude dále popsán převážně tento typ karet. [13]

První část definuje rozměry karet a to, že by měly splňovat standard ISO/IEC 7810, ISO/IEC 15457-1 nebo jakoukoliv jinou velikost. [14] Druhá část definuje charakteristiku elektromagnetických polí pro napájení karty a komunikaci na frekvenci 13,56 MHz, modulační metody a kódová schémata. Typ A používá směrem k čtečce ASK modulaci se 100 % hloubkou dat zakódovanými pomocí upraveného Millerova kódování. Opačný směr využívá OOK modulaci s kódováním typu Manchester. [8]

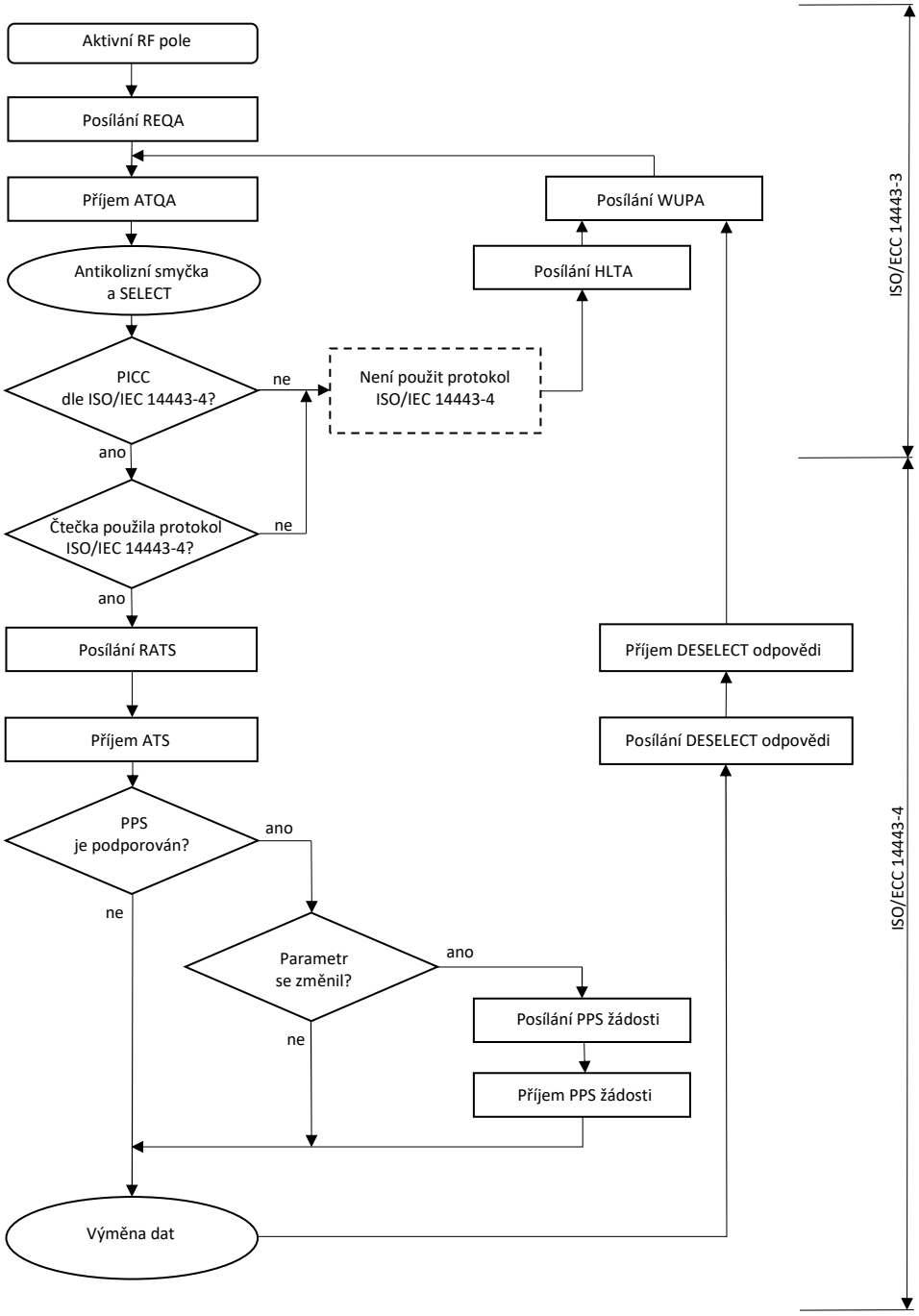
Třetí část definuje inicializaci a antikolizní smyčku. Obsahuje také definice formátu rámce. [15]

Čtvrtá část standardu ISO/IEC 14443-4 definuje poloduplexní blokový přenosový protokol pro komunikaci v bezdrátovém prostředí. Například komunikaci s bezkontaktními kartami nebo jinými PICC (bezkontaktní karta nebo objekt) a aktivační a deaktivační sekvenci pro tento protokol. Protokol definovaný touto částí standardu umožňuje použití protokolu APDU a výběr aplikací, tak jak jsou definovány v standardu ISO/IEC 7816-4. [16]

2.3.1 Inicializační a aktivační sekvence pro bezkontaktní smart karty typu A

Komunikace s bezkontaktní smart kartou začíná příkazem REQA, kterým se bezkontaktní čtečka dotazuje, zda jsou v blízkosti bezkontaktní smart karty. Bezkontaktní smart karta odpovídá sekvencí ATQA (Answer To Request). Sekvence ATQA obsahuje antikolizní bit a velikost UID (unique identifier, česky jedinečný identifikátor). Poté probíhá antikolizní smyčka a detekce, zda bezkontaktní smart karta podporuje standard ISO 14443-4 pomocí odpovědi SAK (Select Acknowledge). V této chvíli má čtečka znalost UID dané bezkontaktní smart karty. Při aktivaci bezkontaktní smart karty se zasílá požadavek na odpověď ke zvolení bezkontaktní smart karty (RATS). Odpovědí bezkontaktní

smart karty je potvrzení výběru (ATS analogie ATR u kontaktních karet standardu ISO/IEC 7816). Dále může proběhnout volitelná část s požadavkem na navýšení bitového toku. Výchozí bitový tok pro bezkontaktní smart karty typu A je 106 kb/s. Tento proces je zachycen na obrázku 2.6. [13] [17]



Obrázek 2.6: Inicializační a aktivační sekvence pro bezkontaktní smart karty typu A [17]

Při inicializaci lze detekovat typ bezkontaktní smart karty, a to pomocí sekvence ATQA. Zde však může nastat kolize s více bezkontaktními smart kartami a na ATQA se tedy nelze zcela spolehnout. Typ bezkontaktní smart karty lze detekovat také pomocí odpovědi SAK. [16]

2.4 Komunikace mezi smart kartou a čtečkou smart karet [18]

Standard ISO/IEC 7816-4 definuje strukturu, zabezpečení a příkazy pro komunikaci se smart kartou. Čtečka karet a smart karta spolu komunikují pomocí zpráv, které jsou v kontextu standardu ISO/IEC 7816-4 nazývané Application Protocol Data Unit, zkráceně APDU. Komunikace probíhá vždy pomocí páru APDU, jeden je příkaz a druhý je odpověď na daný příkaz. Odpověď musí vždy přijít před novým příkazem.

Příkaz má definovanou třídu (CLA), typ instrukce (INS) a dva parametry (P1, P2). Struktura APDU příkazu je zobrazena v tabulce 2.1. Každá z uvedených položek je délky jednoho bajtu. Parametry P1 a P2 indikují způsob a možnosti zpracování příkazu. Poté mohou následovat další data. Kde jeden, případně tři bajty, což je prodloužená varianta, kterou musí smart karta podporovat a informuje o ní v ATR (Answer To Reset), definují délku dat příkazu. Pokud příkaz nemá žádná další data, tak délku nevyplňuje. Dále příkaz obsahuje maximální délku odpovědi (Nr).

CLA	INS	P1	P2	Lc	Data Field	Le

Tabulka 2.1: APDU příkaz

- CLA – Class – třída příkazu, 1 bajt
- INS – Instruktion – instrukce příkazu, 1 bajt
- P1 – Parameter 1 – první parametr příkazu, 1 bajt
- P1 – Parameter 1 – druhý parametr příkazu, 1 bajt
- Lc – Lenght command – délka příkazu, 0 až 3 bajty
- Data Field – Data – data příkazu (APDU dotaz), Lc bajtů
- Le – Lenght expected – očekávaná délka dat odpovědi (APDU odpověď), 0 až 3 bajty

Odpověď obsahuje maximálně tolik bajtů dat, kolik stanoví příkaz, a dva povinné stavové bajty na konci odpovědi. Stavové bajty (SW1, SW2) mají obecnou konvenci definovanou ve standardu. Struktura APDU odpovědi je zobrazena v tabulce 2.2. Obsahuje čtyři třídy stavů:

- Normální stav

- Varování
- Chyba vykonání
- Chyba příkazu

Response	SW1	SW2

Tabulka 2.2: APDU odpověď

- Response – Data odpovědi – 0 až 3 bajty, data s odpovědí s délkou Le, mohou být nulová
- SW1 – Status Word 1 – stavový příznak 1, 1 bajt
- SW2 – Status Word 2 – stavový příznak 2, 1 bajt

Normální stav je definován prvním bajtem s hodnotou 9000 nebo 61XX, který značí, že je možné přečíst další data, jež neobsahovala daná odpověď z důvodu omezení maximální velikosti dat odpovědi Nr. Tyto odpovědi lze vyčíst příkazem GET-RESPONSE a jejich velikost obsahuje druhý bajt stavové dvojice.

Existují dva druhy tříd příkazů (CLA), proprietární a všeobecná. Třídy příkazů lze také použít jako referenci logického kanálu. Logický kanál má své vlastní bezpečností stavy a je možné je použít k běhu více aplikací zároveň.

Instrukční bajt informuje smart kartu o tom, co se má vykonat. Jsou definované všeobecné instrukce jako například pro tvorbu, smazání, úpravy souboru na smart kartě či pro autentizace.

Dále standard ISO/IEC 7816-4 specifikuje strukturu pro aplikace a data. Standard dělí struktury na dvě kategorie: jednoúčelové (dedicated) soubory a základní soubory. Jednoúčelové mohou být aplikace nebo skupiny souborů či objektů. Jednoúčelové soubory mohou být rodiči jiných souborů. Vzniká tím hierarchická struktura. Základní soubory obsahují data a nemůžou být rodičem jiných dat.

Struktury lze vybírat podle jména dedicated souboru. Tento název se může skládat z řetězce až 16 bajtů. Aplikace jsou adresovány pomocí aplikačního identifikátoru (AID), který může být také použit jako název dedicated souboru. Dále je možné soubor zvolit pomocí unikátní reference, jež se skládá ze dvou bajtů. Jinými možnostmi je volba pomocí cesty, která obsahuje postupně jednotlivé reference (názvy nebo identifikátory) souborů, případně krátkou referenci základních souborů pomocí jména, která však nemusí být podporována.

Aplikace jsou identifikovány, jak už bylo uvedeno výše, pomocí až 16 bajtového klíče. V prvním bajtu je zakódovaná kategorie aplikačního klíče. Je definována mezinárodní, národní, standardizovaná a proprietární kategorie. Proprietární aplikace začíná bajtem ‚FX‘.

2.5 Použití modulu čtečky PN532 pro komunikaci se smart kartou [19]

PN532 je modul typu transceiver (vysílač i přijímač) pro bezdrátovou komunikaci pracující na frekvenci 13,56 MHz, který je vyráběn společností NXP Semiconductors. PN532 implementuje demodulaci a dekódovaní signálů mnoha standardů jako jsou ISO/IEC 14443A, MIFARE, ISO/IEC 18092 NFCIP-1 a další. V modulu PN532[1] je použit mikrokontrolér Philips 80C51, který zajišťuje práci s čtecím obvodem PN512, a tím velmi usnadňuje použití tohoto obvodu. Použitím tohoto modulu je vyřešena práce s registry obvodu PN512, přechody mezi stavy, periodické kontroly příznaků a podobně. Modul PN532 umožňuje emulační režim, kdy se modul PN532 chová jako čtecí obvod PN512.

Modul PN532 podporuje 6 různých operačních modů:

- ISO/IEC 14443A/MIFARE Reader/Writer
- FeliCa Reader/Writer
- ISO/IEC 14443B Reader/Writer
- ISO/IEC 14443A/MIFARE Card MIFARE Classic 1K or MIFARE Classic 4K card
- Emulation mode
- FeliCa Card emulation
- ISO/IEC 18092, ECMA 340 Peer-to-Peer

Modul PN532 je kompatibilní se standardem ISO/IEC 14443 a to jak s typem A, tak B. Úplně řeší rámce definované standardem ISO/IEC 14443A a detekci chyb. Splňuje standard ISO/IEC 18092 NFCIP-1 pro aktivní i pasivní komunikační režimy s ostatními zařízeními splňujícími tento standard. Umožňuje datové přenosy do rychlost 424 kbit/s.

Komunikace s modulem PN532 je možná pomocí rozhraní SPI, I2C a vysokorychlostní UART.

2.5.1 Komunikace přes rozhraní SPI

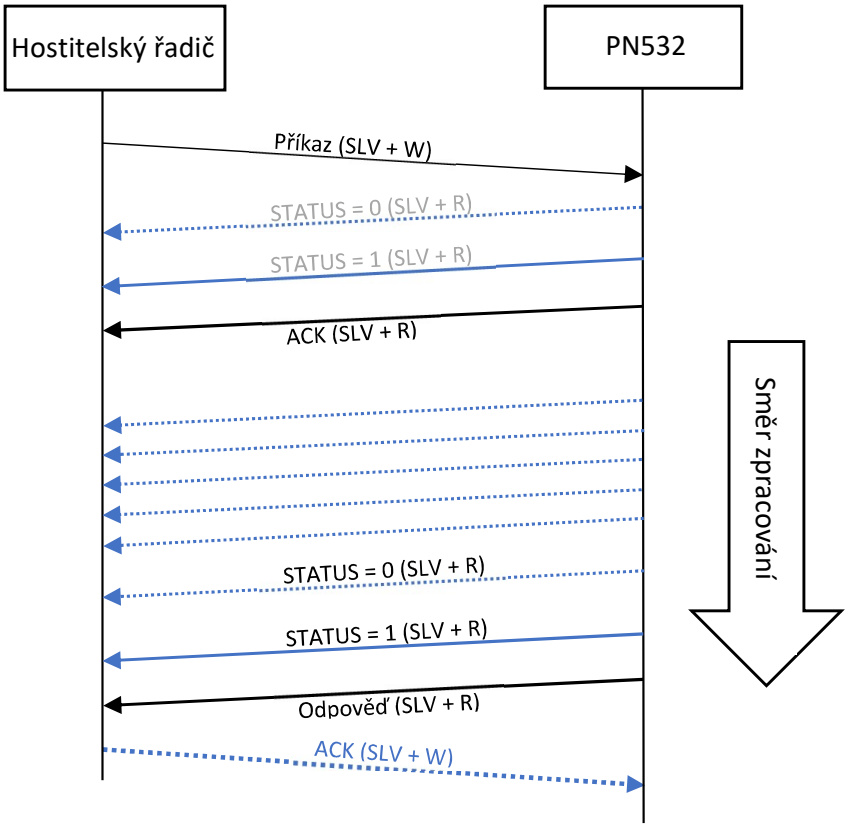
Pro komunikaci s modulem PN532, která je při použití rozhraní SPI poloduplexní, existuje pět typů rámců:

- Normální informační rámec
- Rozšířený informační rámec – pro rámce větší jak 255 bajtů
- Potvrzovací (ACK) rámec – informuje o úspěšném přečtení zprávy

[1] Modul PN532 vznikl spojením obvodu PN512 a mikrokontroléru 80C51 do jedné součástky

- Negativně potvrzovací (NACK) rámec – hostitel používá pro znovu poslání odpovědi z PN532
- Rámec o chybě (error frame) – informuje o chybě na aplikační úrovni

Zásadní rozdíl oproti jiným možnostem komunikace s modulem PN532 je to, že hostitel při komunikaci zasílá jeden bajt informující o typu operace. Touto operací může být zápis dat, čtení dat nebo zjištění stavu (zda má modul PN532 data k přečtení). Posledně jmenovaná operace je pak nutná pouze tehdy, kdy se nevyužívá pin IRQ, který plní stejnou funkci jako čtení stavu. Pin IRQ indikuje logickou jedničkou stav, že modul PN532 má data ke zpracování. Komunikace bez použití IRQ pinu za pomoci rozhraní SPI je zobrazena na obrázku 2.7. [20]



Obrázek 2.7: Komunikace s rozhraním SPI [20]

2.6 Šifrování pomocí algoritmu AES

Jedná se o mezinárodně uznávaný standardizovaný algoritmus, který využívá symetrickou blokovou šifru. Nahrazuje starší algoritmus DES nebo jeho bezpečnější variantu 3DES, která je však mnohem

pomalejší. Je to symetrická šifra, což znamená, že používá stejný klíč pro šifrování i dešifrování. Využívá bloky pevné délky 128 bitů a délka klíče může být v třech velikostech: 128, 192 či 256 bitů. Pro šifrovací standard AES byla zvolena specifická část algoritmu Rijndael, vyvinutého dvěma belgickými autory. [21][22]

V roce 2002 se algoritmus AES stal federálním standardem USA. [22] Využívá se například pro zabezpečení komunikace přes Wi-Fi v režimu WPA2 a WPA3. [23] Zpráva německého Spolkového úřadu pro bezpečnost informační techniky z dubna roku 2020 uvádí, že AES 128 je doporučován pro nové aplikace a není znám žádný závažný útok na tento algoritmus. [24]

2.6.1 Zjednodušený princip algoritmu AES

Algoritmus AES pracuje v několika iteracích. Počet iterací je závislý na délce klíče (10 pro 128 bitů, 12 pro 192 bitů a 14 iterací pro 256 bitů klíče).

1. Data jsou rozdělena do bloků o velikosti 128 bitů
2. Provedení expanze klíčů
    - Z prvotního klíče se se vytvoří nové podklíče pro jednotlivé iterace
3. Klíč se pomocí operace XOR přidá do bloku dat
4. Substituce bajtů (zajišťuje nelinearitu)
    - Bajty jsou nahrazeny pomocí předdefinované tabulky
5. Posunutí řádků
6. Kombinace sloupců
    - Zkombinuje čtyři bajty v sloupci
    - Spolu s krokem 5. zajistí náhodnost v šifře
7. Zahájení nové iterace. Přidání podklíče pomocí XOR operace
    - Pokračuje se bodem 4. dokud se neprovedou všechny iterace
    - Poslední iterace neprovede bod 6., kombinaci sloupců

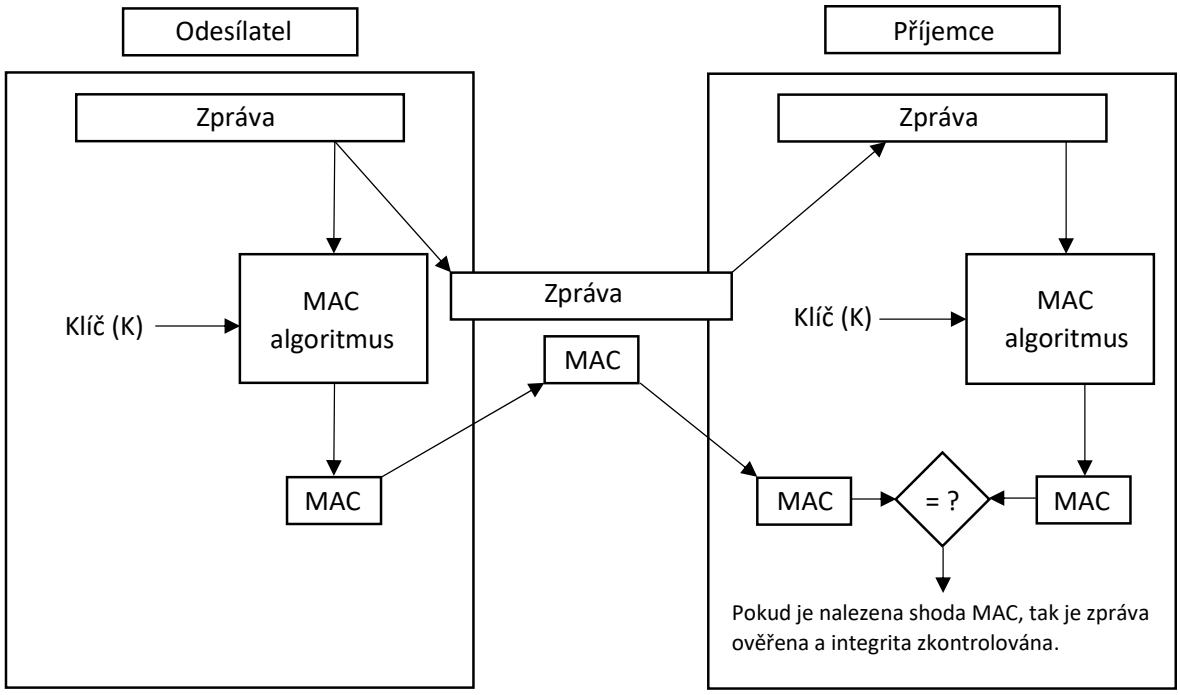
Pro dešifrování se provedou všechny kroky v opačném sledu. [25]

Důležitou součástí algoritmu je také inicializační vektor (zkráceně IV), který se používá k inicializaci prvotního stavu. Inicializační vektor je většinou náhodná, nebo pseudonáhodná sekvence, někdy je však potřeba mít inicializační vektor předem určený. Motivací pro použití inicializačního vektoru je zabránit útočníkovi odvodit vazby mezi bloky zpráv a jejich podobnost, skrývá tedy vzory

v datech. Inicializační vektor tak poskytuje náhodnost, aby stejný blok dat nebyl zašifrován do stejné podoby vícekrát. Pro stejný výstup algoritmu je tedy nutný nejen klíč, ale také inicializační vektor, tím vzniká pár klíč a inicializační vektor. Inicializační vektor se v komunikaci musí vyměňovat, aby mohla komunikace pokračovat. [26][27]

2.6.2 Ověřovací kód MAC

MAC (message authentication code - ověřovacího kód zprávy) je krátká sekvence dat, která zajišťuje autenticitu a integritu zprávy. Vzniká pomocí kryptografické funkce nad daty za pomocí klíče. Využívá symetrického šifrování. Příkladem těchto je například HMAC (použitý pro IPsec, TLS, SSL, SSH, ...), CBC-MAC, Poly1305 nebo OMAC. Využití ověřovacího kódu MAC je zobrazeno na obrázku 2.8. [28] [29]



Obrázek 2.8: Použití kódu MAC [30]

2.6.3 Algoritmus OMAC

Algoritmus One-key MAC je ověřovací kód zprávy zkonstruovaný z blokové šifry a lze použít pro blokovou šifru AES. Autoři OMAC jsou Tetsu Iwata a Kaoru Kurosawa z japonské univerzity

Ibaraki. Existují dvě varianty OMAC, a to OMAC1 neboli CMAC a OMAC2. [31] CMAC je od roku 2005 doporučován americkým úřadem NIST. [27]

2.7 Útoky na bezkontaktní smart karty a jejich dělení

Útoky lze dělit na invazivní, částečně invazivní a neinvazivní.

Invazivní útoky vyžadují vyjmutí mikroprocesoru ze smart karty. Tento typ útoku je sice nejsilnější, ale obvykle vyžaduje velice náročné zařízení a zabere mnoho času. Příkladem takovéhoto útoku je čtení dat ze sběrnice pomocí sond, kdy musí být narušena vnější pasivní vrstva. Zařízení je tedy poškozeno a nelze již použít obvyklým způsobem nebo jen s obtížemi.

Neinvazivní útoky jsou takové, které lze provést, aniž by karta byla jakkoliv modifikována, a tedy fyzicky je karta nepoškozená. Například neinvazivní útok postranním kanálem se provádí pomocí sledování nepřímých informací jako je teplota, časová prodleva a podobně. Tyto informace mohou statisticky odpovídat odpovídajícím výpočtům nebo šifrovacím klíčům.

Částečně invazivní útoky vyžadují, aby byl dosažitelný povrch čipu. Není však nutné provést narušení čipu jako takového. Příkladem je sledování elektromagnetického záření a vkládání chyb do čipu smart karty pomocí laseru. [32] [33]

Dělení útoků na smart karet dle S. Kumara [34]:

- Invazivní
  - Reverzní inženýrství
  - Pomocí sondy (probe attack)
- Neinvazivní
  - Časová analýza
  - Analýza spotřeby
    - Jednoduchá analýza spotřeby
    - Diferenční analýza spotřeby
- Částečně invazivní také známe pod názvem vkládaní chyb (fault injection)
  - Narušení operací
  - Odepření služby (DoS)
  - Útok pomocí elektrického proudu
  - Útok pomocí vysoké frekvence hodin
  - Útok pomocí světelných paprsků
  - Útok pomocí elektromagnetického pole

Útoky lze také dělit na útoky na důvěrnost (získání klíčů, možnost klonovat kartu) nebo integritu (změna dat v smart kartě nebo změna chování smart karty). [32]

2.7.1 Zranitelnosti bezkontaktní smart karty MIFARE Classic

Karty rodiny typu MIFARE classic z roku 1994 jsou částečně založeny na standardu ISO 14443. Využívají proprietární šifrování pomocí CRYPTO1 algoritmu. CRYPTO1 je proudová šifra založená na LSFR registru. Proudová šifra je taková šifra, která vstupní datový tok kombinuje s pseudonáhodným proudem bitů. Pseudonáhodný proud bitů je nazvaný keystream a je vytvořen na základě šifrovacího klíče. LSFR (linear feedback shift register) je posuvný registr, kde jeho vstupní bit je lineární funkcí závislou na předchozím stavu. [35]

Návrh řešení částečně spoléhal na to, že algoritmus CRYPTO1 nebyl veřejně znám, protože už v roce 2000 byl napadnutelný algoritmus DES, a tedy pokud by byl CRYPTO1 veřejný, tak by byl podobně napadnutelný, protože používal pouze klíč délky 48 bitů. Postupně se objevovaly další problémy, kdy se povedlo pomocí reverzního inženýrství, spočívajícího v rekonstrukci obvodů čipu a odposlechu komunikace, zjistit kryptografický algoritmus. Po odhalení všech problémů tohoto algoritmu stačí zachytit komunikaci čtečky se smart kartou a následně lze z této komunikace vytvořit klon použité karty. [36] Zranitelnosti smart karty MIFARE Classic a algoritmu CRYPTO1 dle výskytu:

1. Problémy generátoru pseudonáhodných čísel
2. Problémy šifrovací funkce
3. Chyby v komunikačním protokolu
4. Chyby v nasazení MIFARE Classic

2.7.1.1 Problémy generátoru pseudonáhodných čísel

Generátor pseudonáhodných čísel pro smart karty MIFARE Classic má dva zásadní problémy. Prvním z nich je malá entropie generovaných čísel a to taková, že je použito pouze 16 pseudonáhodných bitů. LSFR registr tak generuje sekvence, které mají periodu 65 535 ($2^{16} - 1$). K vygenerování všech možností poté stačí jen 0,6 sekundy.

Druhým problémem je, že při každém novém napájení je LSFR registr nastaven na pevně danou hodnotu. [37] [36]

2.7.1.2 Problémy šifrovací funkce

Algoritmus Crypto-1 využívá pouze určité liché bity stavu, který se nachází v LSFR registru. Tímto se zmenšuje prostor možných stavů nutných k prohledání.

Nejsou použity všechny stavové liché bity a je tedy určitým způsobem možné získat předchozí stavy LSFR registru pomocí vytvoření funkce, jež zpětně vytváří předchozí stav registru LSFR na základě jeho aktuálního stavu.

Šifrovací algoritmus je také statisticky vychýlen (v angličtině nazýván jako bias) a liší se tedy od pravé náhodné sekvence. Některé stavové bity mají tedy větší vliv na proud bitů keystream než jiné. Například poslední tři bity stavu ovlivňují proud bitů keystream s pravděpodobností pouze 25 procent. V ideálním případě by stavové bity ovlivňovaly proudový klíč v polovině případů. [37] [35] [36] [38]

2.7.1.3 Chyby v komunikačním protokolu

Jedním z problémů smart karet MIFARE Classic je opětovné použití paritních bitů. Protože paritní bit se počítá pro nezašifrovaný obsah, tak pro paritní bit a následující bit nezašifrovaného obsahu se používá stejný proud bitů keystream. To by u proudových šifer nastávat nemělo.

Při chybě komunikace, jež nastane při ověřovacím procesu, dochází k úniku informací o bitech proudu bitů keystream.

Po úspěšném ověření se nemění stav LSFR registru při ověření pro jiný sektor. To způsobuje, že při znalosti jednoho klíče je možné přečíst informace z ostatních sektorů, ke kterým by normálně nebyl možný přístup. [37] [36]

2.7.1.4 Chyby v nasazení smart karet MIFARE Classic

Při výrobě smart karet MIFARE DESfire jsou nastavené výchozí klíče, které jsou známé. Pokud provozovatel těchto karet tento výchozí klíč nezmění, je velice snadné kartu napadnout, protože výchozí klíče jsou veřejně dostupné.

Ve výchozím stavu jsou také nastavena práva k datovým blokům a provozovatel karet by je měl změnit. [37] [36]

2.7.2 Útok na bezkontaktní smart karty MIFARE DESfire MF3ICD40

V roce 2011 se povedlo týmu z německé Rýnské univerzity úspěšně napadnout karty typu MIFARE DESfire verze MF3ICD40, která je představitelem první řady DESfire. Útok trvající cca sedm hodin dokáže získat bezpečnostní klíč karty. Útok je neinvazivní a nezanechává tedy žádné stopy o tom, že byla karta prolomena. Takový útok šlo v době vydání článku provést se zařízením v ceně do 3000 $, a proto používání těchto karet není zcela bezpečné. Nové verze karet typu MIFARE DESfire tuto chybu neobsahují. [39]

Princip útoku na MIFARE DESfire MF3ICD40 spočívá v analýze postranních kanálů (side channel analysis) a šablonovém útoku (template attack). Toto je metoda zaznamenávající elektromagnetické záření a jiné fyzikální charakteristiky, čímž lze odvodit důležité detaily o procesech uvnitř karty. Pomocí této analýzy byli schopni napadnout šifrovací algoritmy AES nebo 3DES,

které zatím nelze efektivně napadnout útoky typu brute-force a AES ani analytickými útoky. [40] Pro algoritmus 3DES existuje útok sweat32. [41]

Sledováním diferenční analýzy výkonu za pomoci šablon vzniklých na základě předchozích měření lze postupně najít bezpečnostní klíč ke kartě. Šablony vznikají na základě sledování dat, kdy útočník vytvoří šablony chování zabezpečeného zařízení, jež poté použije k rychlému získání zabezpečených dat. K tomu, aby takovéto šablony mohl vytvořit, útočník musí mít přístup k jiné kopii zabezpečeného zařízení, které může plně ovládat. [42]

2.7.3 Útoky na bezkontaktní smart karty MIFARE DESfire EV1

Smart karty typu MIFARE DESfire EV1 a vyšší verze už jsou proti výše uvedenému typu útoku odolné a prozatím není zaznamenán žádný úspěšný pokus o prolomení této karty. Objevují se ukázky pokusů napadnutí této karty. Ty ale spočívají v použití zpětně kompatibilních režimů nebo zneužití transakčního mechanismu, kdy se nesprávně využívá tento mechanismus systému. Algoritmická složitost a návrh karty by měl zamezit algoritmickým útokům. Útoky postranním kanálem by také neměly být možné. [43]

2.7.4 Ochrany před útoky postranním kanálem

I na úrovni obvodů se výrobce karet snaží zabránit, aby bylo možné provádět analýzy spotřeby elektrického proudu. Toho lze docílit například použitím obvodů, které vyrovnávají odběr elektrického proudu karty. Toto v praxi není vůbec jednoduché vyřešit. Musí se vyrobit doplňkové obvody s velmi podobnou charakteristikou a ty nejsou jednoduše masově vyrobitelné z důvodu nutnosti velké výrobní přesnosti. Další možností řešení je přidání šumu, například náhodné zvyšování spotřeby při kryptografickém výpočtu. Obě tyto možnosti schovávají data, změny spotřeby, které postranní útok sleduje. [44]

Na úrovni obvodů je také možnost zamezit těmto útokům pomocí změn času, tím útočníkovi zamezíme časovou synchronizaci mezi jednotlivými měřeními. Můžeme například náhodně měnit hodinový signál okruhu.

Na úrovni algoritmické je také několik možností, jak znesnadnit nebo zamezit těmto typům útoku. Jednou z nich je náhodná změna pořadí vykonávání šifrovacího algoritmu. Dále je možné vkládat instrukce nebo iterace, které neovlivňují výsledek.

Na úrovni protokolu je vhodné omezit čas mezi vytvořením jednotlivých klíčů/jednorázových hesel. U chytřejších karet je možné toto omezení provést až po několika neúspěšných pokusech. Tento princip je podobný tomu, který se objevuje u přihlášení do internetového bankovnictví a jiných citlivých aplikací. [45]

2.8 Bezkontaktní smart karty řady MIFARE

Typy karet řady MIFARE:

- MIFARE Classic
- MIFARE Plus
- MIFARE Ultralight
- MIFARE DESfire

MIFARE Classic jsou první smart bezkontaktní karty z řady MIFARE, které začaly být vyráběny v roce 1994. V roce 1996 byly poprvé nasazeny ve veřejné hromadné dopravě v jihokorejském hlavním městě Soulu. V roce 2015 proběhlo zavedení nového vylepšení karet v podobě Classic EV1. Zde došlo ke zdokonalení ochrany před elektrostatickou elektřinou pomocí nového výrobního procesu a byla přidána známka originality. Tyto karty podporují pouze první tři části standardu 14443 typu A. Obsahují paměť až 4 kB. Nejsou bezpečné, protože proprietární algoritmus CRYPTO1 byl prolomen. [46] [47]

Smart karty MIFARE Plus vznikly na základě karet Classic, se kterými si zachovávají zpětnou kompatibilitu, a mají být jejich náhradou. Jejich výhodou je plná podpora standardu ISO/IEC 14443 A 1-4 a ISO 7816-4. Důležité také je, že doposud nejsou žádné známé úspěšné útoky na tyto karty. [46] [48]

Řada smart karet MIFARE Ultralight je určena jako levné řešení pro vysoko-objemové aplikace (koncerty, hromadná doprava). Jejich cílem je nahrazení čárových kódů nebo magnetických proužků, není tedy kladen zásadní důraz na bezpečnost a existují články o napadnutelnosti těchto karet i novějších řad (Ultralight C). [46] [49]

Řada smart karet MIFARE DESfire dává důraz na bezpečnost a funkce. Podporuje šifrování tripleDES a AES. Tato řada karet má několik evolucí, které postupně zlepšují kompatibilitu se standardy, bezpečnost a možnost využití v prostředí s více poskytovateli služeb na jedné kartě. Existuje také odlehčený typ DESfire Light. [46] [50] Pro smart karty DESfire kromě zastaralé MF3ICD40 není znám žádný útok. [43]

2.9 Bezkontaktní smart karty MIFARE DESFire

První bezkontaktní smart karty rodiny MIFARE DESfire byly uvedeny v roce 2002 s důrazem na vysokou hardwarovou a softwarovou bezpečnost. Jsou určené pro aplikace jako jsou platební systémy, hromadná přeprava osob, přístupové systémy apod. V těchto situacích je možné, aby více různých provozovatelů služeb sdílelo jednu kartu, aniž by byla jakkoliv kompromitována bezpečnost. [51]

Existují tři evoluce MIFARE DESfire a jedna původní zastaralá a napadnutelná Mifare DESfire D40. [40] Všechny verze jsou zpětně kompatibilní.

Karty obsahují předprogramovaný operační systém, který má jednoduchou složkovou strukturu se soubory. Karty MIFARE DESfire splňují plně standard ISO/IEC 14443 typ A a postupně se každá další evoluce stávala více kompatibilní se standardem ISO/IEC 7816-4. [50]

Karty mají možnost mít několik různých aplikací a každá aplikace má své soubory. Obsahuje anti-tearing mechanismus, který zaručuje transakční integritu dat. Všechny evoluce karet MIFARE DESfire podporují přenosové rychlosti 106 kb/s (výchozí pro bezkontaktní karty), 212 kb/s, 424 kb/s a 848 kb/s. Délka unikátního identifikátoru karet je 7 bajtů a je možné zapnout možnost náhodného ID, podobně jako používají současné mobilní telefony s podporou technologie NFC. Karta obsahuje 16bitový procesor a šifrovací koprocesor pro AES/Triple-DES. Má ROM paměť o velikosti 48 Kb. Zabudován je zde také generátor opravdových náhodných čísel. [52] [46]

Karty MIFARE DESfire podporují strukturu zpráv APDU, definovanou v standardu ISO/IEC 7816-3. Všechny evoluce (EV1 – EV3) podporují tyto ISO/IEC 7816-4 příkazy:

- SELECT FILE
- READ BINARY
- UPDATE BINARY
- READ RECORDS
- APPEND RECORD
- GET CHALLENGE
- INTERNAL AUTHENTICATE
- EXTERNAL AUTHENTICATE

2.9.1 Organizace paměti smart karet MIFARE DESfire [50] [53] [54]

Velikosti paměti bezkontaktních smart karet MIFARE DESfire jsou v těchto variantách - 2 kB, 4 kB a 8 kB. Data uložená na kartě mají garantovanou životnost 10 let. Maximální počet aplikací na jedné smart kartě je 28 a jsou identifikovány pomocí 3 bajtů AID. Velikost souboru, případně i kapacita, je určena při vytvoření souboru.

Smart karty MIFARE DESfire mohou obsahovat různé typy souborů:

- Standardní datový soubor
- Zálohovací datový soubor
- Hodnotu se zálohou

- Lineární záznam
- Cyklický záznam
- Transakční MAC soubor (od verze EV2)

Standardní datový soubor je běžný neformátovaný soubor, do kterého lze zapisovat libovolná data a poté je číst. Zálohovací datový soubor je podobný standardnímu, ale má navíc transakční mechanismus, který umožňuje zapsat všechna data (COMMIT), nebo žádná (ROLLBACK).

Soubor hodnoty se zálohou obsahuje 32bitovou celočíselnou hodnotu, již lze měnit operacemi credit a debit, kdy credit přidává a debit odečítá z hodnoty. Tento typ souboru podporuje transakce. Je například použitelný u provozovatelů hromadné dopravy pro elektronickou peněženku nebo evidenci počtu jízd.

Lineární záznam je soubor pro zápis mnoho podobně strukturovaných dat. Má omezený počet záznamů, které se postupně plní. Po dosažení maximální kapacity nelze dále zapisovat a pro další zápis se musí celý soubor vyčistit.

Cyklický záznam je podobný lineárnímu, ale při naplnění maximální kapacity je poslední soubor automaticky smazán.

Transakční MAC soubor se používá k ověření pravosti provedené transakce mezi čtecím zařízením a bezkontaktní kartou. Využívá se k vytvoření důvěry v případě, že je více poskytovatelů služby, nebo k odhalení podvodných transakcí. [55]

2.9.2 Certifikace EAL [56] [57] [58]

Mezinárodní standard Certifikace Evaluation Assurance Level (EAL) je číselné ohodnocení systému podle Common Criteria for Information Technology Security Evaluation (Standard ISO/IEC 15408) platné od roku 1999. Tato certifikace ohodnocuje systém podle úrovně jistoty správně implementovaných bezpečnostních funkcí a vlastností. Dále ohodnocuje, jakým způsobem bylo zajištěno testování systému. Velmi ale záleží, jaké funkční bezpečnostní cíle si systém stanovuje. Proto je nutné sledovat bezpečnostní cíle (security target), které definují bezpečnostní vlastnosti systému.

Ve standardu je definováno sedm úrovní jistoty ohodnocení:

1. EAL1 funkcionálně testováno
    - Ohodnocení produktu uživatelem, tak jak bylo dodáno bez dalších informaci
2. EAL2 strukturálně testováno
    - Vývojáři dodají informace o řešení systému, ale nic nad rámec běžného vývoje
3. EAL3 metodicky testováno a kontrolováno
    - Kontrola a základní vedení v průběhu vývoje

4. EAL4 Metodicky navrženo, testováno a revidováno
    - Použití přísných komerčních postupů vývoje
5. EAL5 Semi-formálně navrženo a testováno
6. EAL6 Semi-formálně ověřený návrh a testováno
7. EAL7 Formálně ověřený návrh a testováno

2.9.3 Evoluce smart karet MIFARE DESfire

Smart karty MIFARE DESfire byly vydávány v několika řadách, výrobcem karet nazývaných evolucemi.

Karty řady EV1 splňují certifikaci CC EAL4+. Karty podporují až 28 aplikací a 32 souborů v jedné aplikaci. Mohou mít zapnuté náhodné ID při aktivaci. [50]

Karty řady EV2 dosahují na certifikaci CC EAL5+. Nemají omezený počet aplikací a zároveň přidávají nový typ souboru transakční MAC. Zlepšuje se podpora standardu ISO 7816-4. Je dosaženo zlepšení správy klíčů a jejich sdílení. Umožňuje kontrolu vzdálenosti od čtecího zařízení. [53] [59]

Karty řady EV3 přidávají sdílenou správu aplikací a další bezpečnostní funkce, jako je kontrola originality karty pomocí ECC podpisu. [54]

2.9.4 Zabezpečení smart karet MIFARE DESfire

Jak název lehce napovídá, tak prvotní verze karet MIFARE DESfire využívaly šifrovací algoritmus DES, avšak ten se ukázal neefektivní proti útokům typu brute-force [60], a proto byl v novějších evolucích karet nahrazen novějším 3DES (také už není zcela bezpečné [61]) a poté AES šifrováním, které obsahovalo navíc i CMAC a další bezpečnostní funkcionality.

2.9.4.1 Mechanismus anti-tearing

Bezkontaktní smart karty jsou bez vlastního napájecího zdroje. Jsou napájeny vzdáleně pomocí elektromagnetické indukce v RF poli čtecího zařízení. Situace, kdy uživatel pohne kartou od čtečky a karta kvůli tomu ztratí napájení, se nazývá tearing. To může způsobit neúplný zápis nebo narušení paměti. [62]

2.9.4.2 Relay útok

Karty řady EV2 a výše umožňují kontrolu vzdálenosti od čtecího zařízení. Tato kontrola má zamezit relay útokům, které jsou obdobou útoku typu man in the middle. Příkladem takového útoku je to, když k osobě s přístupovou kartou přistoupí útočník, jenž pomocí svého zařízení vytvoří spojení mezi zabezpečeným objektem (např. vozidlem) a kartou na větší vzdálenost. Inicializuje se tak

sekvence mezi kartou a vozidlem k otevření, aniž by osoba s bezpečnostní kartou byla fyzicky blízko u vozidla, čímž útočník získá přístup k vozidlu. [53]

2.9.4.3 Diverzifikace klíče [63]

Diverzifikace klíče je způsob, jak lze odvodit klíče z určitého unikátního vstupu, například z hlavního klíče. Každá karta pomocí této diverzifikace získá svůj unikátní klíč. Smyslem této diverzifikace je zabezpečení toho, že i v případě získání klíče ke kartě bude tento klíč použitelný jen pro tuto napadenou kartu a nebude napaden celý systém.

Klíč je vygenerován například na základě pevného UID karty nebo pomocí jiného typu personalizace. Všechny klíče je možné vygenerovat na základě jediného hlavního klíče, ale je také možné takovýchto hlavních klíčů použít více.

2.9.4.4 Algoritmus pro vygenerování diverzifikovaného klíče pomocí AES-128 [64]

Vstup algoritmu jsou unikátní data pro diverzifikaci a hlavní klíč. Například UID bezkontaktní karty a 16 bajtový hlavní (master) klíč. Výstupem algoritmu je diverzifikovaný 16 bajtový klíč, který je odvozen z hlavního klíče a daného unikátního vstupu o maximální velikosti 31 bajtů. [63] Postup pro generování diverzifikovaného klíče:

1. Vypočítá se CMAC za pomocí hlavního klíče a diverzifikačních dat
2. Před vypočtený CMAC se vloží bajt 0x01 a případně se data zarovnají tak, aby délka dat byla 32 bajtů.
3. Provede se diverzifikace na základě hlavního klíče pomocí algoritmu AES s inicializačním vektorem nastaveném na samé nuly

2.9.5 Aplikace, klíče, zabezpečení [53] [53] [50]

Každá aplikace je identifkována pomocí 3 bajtového AID, také je možné využít názvu dedikovaného souboru, jak je definováno v standardu ISO/IEC 7816-4.

Bezkontaktní smart karty MIFARE DESfire podporují propracovaný systém kontroly přístupu k datům. Souborový systém umožnuje sdružovat data do jednotlivých aplikací a souborů, přičemž přístup a práva k jednotlivým souborům lze konfigurovat pro každý soubor zvlášť. Správa jednotlivých aplikací je možná rozdělit pro více poskytovatelů. Tímto způsobem má pak každý poskytovatel přístup a správná práva jen ke svým aplikacím nebo k aplikacím, které vyžaduje pro svou práci.

Karta obsahuje dvě úrovně klíčů. Jedna úroveň jsou klíče, či pouze jeden hlavní klíč (master key) pro správu aplikací, kterými je možné vytvářet, upravovat a nastavovat jednotlivé aplikace. Druhá úroveň jsou klíče na úrovni aplikací. Těch je možných pro jednu aplikaci až 14 a umožňují pro každý klíč, který může odpovídat poskytovateli (správci) aplikace, určitá práva k dané aplikaci.

Lze také delegovat dané aplikace bez toho, aby bylo nutné poskytovat hlavní klíč pro správu aplikací (MIsmartApp). [51]

2.9.6 Nativní DESFIRE APDU

Pro komunikaci s bezkontaktními kartami MIFARE DESfire je možné využít dvou různých typů protokolu APDU. Buď standardního, jak je definováno v standardu ISO/IEC 7816-4, nebo je možné použít nativní APDU pro DESFIRE. Podpora pro standard ISO/IEC 7816-4 APDU byla postupně vylepšována, zatímco pro řadu EV1 byla pouze možnost vložit nativní protokol APDU do protokolu APDU standardu ISO/IEC 7816-4. [50] V dalších řadách je podpora standardu ISO/IEC 7816-4 vylepšena a používá příkazy, které jsou definované jako obecné pro všechny smart karty. [53] [54]

2.9.7 SAM moduly

Secure access module (dále SAM) je modul pro zabezpečený přístup. SAM modul obsahuje zabezpečenou paměť a podporuje kryptografické operace. Typickým SAM modulem je SIM karta. Smyslem použití SAM modulu je, že všechny kryptografické klíče jsou zde uloženy a operace s nimi se provádí pouze v SAM modulu a nelze je extrahovat, jsou tedy bezpečeně uloženy. SAM modul může také fungovat jako proxy mezi smart kartou a čtecí aplikací, kdy SAM modul všechny data posílaná na kartu šifruje a data ze smart karty dešifruje. Z tohoto důvodu čtecí aplikace nemusí řešit žádné operace ohledně kryptografie. [65]

2.9.8 MIFARE SAM AV3

MIFARE SAM od výrobce NXP Semiconductors je SAM modul, který nativně podporuje karty MIFARE. Podporuje algoritmy DES, TDEA, AES a RSA. Kromě klasického použití má navíc režim X-mode, který umožňuje SAM modulu přímou komunikaci s obvodem pro čtení karet. Režim X-mode podporuje čtecí obvody PN512, RC663 a RC52x. Má také možnost spouštět uživatelský kód v zabezpečené paměti a tím umožnit vlastní zabezpečenou logiku. Cena tohoto modulu se v době vzniku této práce pohybovala orientačně okolo 500 Kč za kus. [66] [67] [68]

2.10 Vzdálené aktualizace OTA (Over the air update)

OTA aktualizace je metoda, která umožňuje vzdáleně distribuovat aktualizace, nové verze software, nastavení či jiné bez nutnosti být fyzicky u daného zařízení. Usnadňuje rozšíření aktualizace v případech, kdy je složité se fyzicky k zařízení dostat nebo když je zařízení příliš mnoho, a technik by je ručně aktualizoval velmi dlouho. Ve frameworku ESP-IDF pro mikrokontrolér ESP32 je tato možnost nativně podporována a jsou implementovány základní i pokročilé možnosti, jak takovéto aktualizace provést. [69]

2.10.1 Oddíly a oddíly paměti ESP32 [70]

Mikrokontrolér ESP32 má svou flash paměť rozdělenou pomocí tabulky oddílů (partition table). Oddíly mohou obsahovat několik aplikací anebo jiná data, jako například souborový systém, certifikáty nebo obrázky, kalibrační data a jiné. Začátek tabulky oddílů je ve výchozím nastavení na adrese 0x8000. Tato adresa lze však změnit. V takovém případě musí být zavaděč zkompilován s novou adresou. Tabulka oddílů může obsahovat nejvýše 95 oddílů. Integrita dat je zajištěna kontrolním součtem pomocí algoritmu MD5 na konci tabulky oddílů. Framework ESP-IDF má dvě předdefinované tabulky oddílů. Jednu tabulku oddílů pro jednu tovární aplikaci, která je nahrána a nelze měnit ji za běhu, a druhou tabulku pro dva OTA oddíly a jednu tovární aplikaci. Uživatel si může vytvořit vlastní definici tabulky oddílů, a tedy vlastní rozdělení paměti. Vlastní definice se tvoří jako dokument ve formátu csv, jenž je potom zpracován programem, který z něj vytvoří binární tabulku, kterou lze nahrát.

Každá položka v tabulce oddílů má název, typ (aplikace, typ nebo jiné), podtyp a adresu, kde oddíl začíná v paměti. Podtypy pro typ aplikace jsou:

- Tovární (0x00) – nelze změnit pomocí OTA aktualizace a nemůže být odstraněna
- OTA (0x10) – pokud se chce použít OTA aktualizace, je nutné mít alespoň 2 oddíly podtypu OTA. Maximální počet OTA oddílů je 15
- Test (0x20) – je podtyp rezervovaný pro testovací tovární procedury. Je také použit, pokud se nenalezne žádný validní oddíl aplikace. Lze také nakonfigurovat tak, aby byl tento oddíl spuštěn pomocí GPIO pinu při startu.

Důležitým oddílem je oddíl typu data, který má podtyp OTA. Tento oddíl obsahuje informaci, který OTA oddíl se má zavádět, a proto musí být vždy přítomný, pokud chceme využít OTA aktualizaci. Tento oddíl se skládá ze dvou flash sektorů z důvodu, aby bylo zabráněno poškození dat, a každý sektor má čítač, aby bylo možné zjistit, který oddíl je novější a má být použit.

2.10.2 Bootloader (zavaděč)

Zavaděč mikrokontroléru ESP32 provádí tyto kroky:

1. Provede minimální inicializaci vnitřních modulů
2. Vybere oddíl, který má být zaveden
3. Nahraje tento oddíl do RAM pamětí a spustí

Zavaděč má pevně danou adresu 0x1000 v paměti. Programátor aplikací má také možnost si napsat svůj vlastní zavaděč a nevyužít ten, který je dodáván pro framework ESP-IDF.

Další z možností bootování je použití zabezpečeného zavedení (secure boot), kdy je zajištěno, že jenom podepsaný kód lze spustit. Tato kontrola se provádí při každém startu. Pokud je tato možnost zvolena, nelze již bootloader aktualizovat. Jestliže se tato možnost použije zároveň s šifrováním flash paměti, mělo by být nemožné spustit jiný než podepsaný kód. Bohužel tato skutečnost není zcela pravdivá a existují útoky, které mohou obejít toto zabezpečení. V novějších revizích čipu jsou však známé útoky vyřešeny. [71]

2.10.2.1 Secure boot

Při prvním zavedení zavaděč vygeneruje jedinečný zaváděcí klíč, který uloží do eFUSE paměti. Klíč používá šifrovací algoritmus AES-256 a je použit k výpočtu otisku zavaděče. Při dalších spouštěních bootovací ROM program ověří otisk oproti aktuálnímu zavaděči. K tomuto ověření aplikace zavaděčem je použit algoritmus Elliptic Curve Digital Signature (ECDSA). Zavaděč obsahuje pouze veřejný klíč, zatímco privátní klíč, kterým se mohou podepisovat další verze aplikace, má vývojář aplikace. Bez znalosti privátního klíče nelze podepsat novou aplikaci. Pokud by útočník zjistil veřejný klíč, neměl by z něj žádný užitek. [71]

2.10.3 Šifrování flash paměti

Pomocí šifrování paměti můžeme zabezpečit, aby data uložená v paměti flash nebyla čitelná například při přímém přečtení flash paměti. Firmware je nahrán v nešifrované podobě a je zašifrován až při prvním zavedení, kdy je vygenerován unikátní klíč vázaný na zařízení, případně je možno použít vlastní předem vygenerovaný klíč. Vždy se šifruje bootloader, oddíly aplikací a tabulka oddílů. Ostatní oddíly mohou být volitelně také zašifrované. K šifrování se používá algoritmus AES-256. [72]

2.10.4 Útok na Secure Boot a flash encryption

Útok je založen na principu fault injecton (injekce chyb). Takový útok je založen na narušení chování systému vkládáním chyb pomocí fyzických prostředků. Metody, jak provést tento útok, mohou spočívat například ve velmi přesném měnění napětí, změnách teplot nebo pomocí silného magnetického pole či laserového záření. Útočník způsobí malé chyby externím komponentům tak, aby narušil normální činnost čipu. [73] Pokud je útok veden na bezpečnostní podsystém, může vyústit v získání kryptografických klíčů nebo k přeskočení bezpečnostních procedur. Kryptografické klíče lze dále zneužít k přečtení a modifikaci dat, jako je firmware nebo data v paměti zařízení.

Útok zveřejněný LimitedResults je založen na tomto principu. Jeho výsledkem je odhalení klíče pro Secure Boot, zneplatnění Secure Boot a zavedení vlastního neověřeného firmware. [74] [75] Dokonce je také možné získat klíč k šifrování flash paměti, a tak přečíst data, která jsou uložena v paměti. [76] [77]

Pro tento typ útoku však musí být umožněn fyzický přístup k čipu ESP32, musí být odstraněny některé části a modifikovány elektronické obvody. Útočník také musí mít přístup k napájecím pinům čipu, aby mohl opakovaně provádět pokusy o přesné změny napětí. [76] [74]

Kvůli možnosti přečíst citlivá data v mikrokontroléru ESP32 je tedy nevhodné mít uložená jakákoliv data na čipu ESP32 staršího, než je revize ESP32-D0WD-V3. Případně je vhodné použít čip ESP32-S2, který je zaměřen na bezpečnost a měl by být odolnější proti podobným útokům. Na druhou stranu tento útok není možné realizovat na velkém počtu zařízení, protože klíče jsou vázané na zařízení, fyzický útok se tedy musí provést na každém zařízení zvlášť. [76]

2.10.5 Vzdálená OTA aktualizace

V prostředí ESP-IDF jsou data potřebná pro provedení OTA aktualizace přijímána, zatímco běžný firmware běží, tudíž se nemusí přerušit funkčnost zařízení a je minimalizován čas výpadku. Nový firmware se zapisuje do OTA oddílu, který se zrovna nepoužívá, a proto je nutné mít alespoň dva OTA oddíly. Po stažení dat proběhne validace nově staženého oddílu, a pokud je validace úspěšná, označí se jako oddíl k zavedení při příštím spuštěním. Označení se provede zápisem do datového OTA oddílu.

Hlavička OTA dat obsahuje informace o verzi aplikace, datu a čase kompilace a verzi frameworku ESP-IDF. Lze také zjistit verzi aplikace, která zrovna běží. Takto se může porovnat, zda existuje novější verze a zda má OTA aktualizace proběhnout. Verze aplikace je řetězec, jenž může být nastaven v CMakeLists.txt, makrem v kódu aplikace, případně se použije text z příkazu git describe. Pokud ani jedno není k dispozici, tak je výchozí hodnotou 1. [69]

Framework ESP-IDF podporuje funkci App Rollback, která umožnuje běh firmware s kritickou chybou. Po stažení a zavedení nového firmware musí program zavolat funkci, jež označí nový firmware jako funkční. Pokud to neudělá nebo jestli označí firmware jako za chybný, tak se při příštím spuštění zavede předchozí verze firmwaru. Další možností je použít s App-Rollback možnost Anti-Rollback, kdy je možné definovat minimální verzi firmware, která může být spuštěna. [69]

Framework ESP-IDF má dva způsoby jak provést OTA update v kódu:

- Jednoduchá OTA aktualizace – jednoduchá funkce, které se předá certifikát a URL adresa, kde se nachází nová verze firmwaru. Data se stáhnou pomocí HTTPS

- Sekvence volání, které umožňují lepší kontrolu nad procesem a není nutné stahovat data pomocí HTTPS, ale lze použít cokoliv jiného, co je k dispozici (UART, SPI, dokonce i BLE)

Doporučuje se stahovat data šifrovaně a zabezpečeně. V případě jednoduché OTA aktualizace je HTTPS vynucené a lze ho vypnout pouze za účelem hledání chyb při vývoji.

2.11 Aktuální stav OTA aktualizací

Jak modul základny, tak modul čtečky využívá předdefinované oddíly pro OTA aktualizace. Důležitým předpokladem je, že aktualizace modulu čtečky musí proběhnout dříve než aktualizace modulu základny. Pokud dojde k nedodržení tohoto pořadí, nelze zaručit funkčnost systému, jelikož by po aktualizaci modulu základny nemusela fungovat komunikace s modulem čtečky a nebylo by jak aktualizovat modul čtečky pomocí OTA aktualizace prostřednictvím modulu základny (změněná struktura zprávy, jiná rychlost komunikace - baud rate a podobně).

2.11.1 Modul základny

OTA aktualizace pro modul základny přístupového systému je poměrně jednoduchá. V nevolatilní paměti (NVS) má modul uloženu informaci o aktuálním čísle aktualizace. V konfiguračním řetězci ve formátu JSON je uloženo číslo verze, která má aktuálně běžet na modulu základny.

Pokud se při stažení konfigurace, jež se děje při startu modulu, číslo v NVS paměti a v konfiguraci liší, zahájí se OTA aktualizace. Ta se provádí pomocí zjednodušeného procesu OTA aktualizace. Informace o tom, kde lze stáhnout binární soubor s novou verzí, je uvedena v přijaté konfiguraci. Po úspěšném dokončení OTA aktualizace se přepíše číslo verze v NVS paměti číslem z konfigurace a tak se zajistí, že se již nebude znovu aktualizovat na tuto verzi. Poté se provede restart mikrokontroléru ESP32 a zavede se nová verze firmware.

2.11.2 Modul čtečky

OTA aktualizace pro modul čtečky je komplikovanější, jelikož modul není přímo připojen k síti LAN a musí komunikovat přes sběrnici UART s modulem základny.

Proces OTA aktualizace probíhající na modulu čtečky začne, když přijde z modulu základny zpráva typu UPDATE START, která indikuje, že základna zahájila OTA aktualizaci pro modul čtečky. Čtečka tento začátek procesu aktualizace indikuje pomocí blikání LED diod. Poté na modul čtečky přicházejí zprávy typu UPDATE DATA, které jsou zapisovány přímo na OTA oddíl. Při příchodu zprávy UPDATE END se označí nově zapsaný oddíl k zavedení a zpráva o úspěchu aktualizace se pošle modulu základny. Po krátkém časovém úseku (2 sekundy) se mikrokontrolér ESP32 restartuje a zavede se nová verze.

Proces OTA aktualizace modulu čtečky z pohledu modulu základny začíná podobně jako aktualizace základny. Informace o verzi modulu čtečky je uložena také v paměti NVS modulu základny. Je tomu tak proto, aby modul základny, který čte konfiguraci, mohl rozhodnout, zda má provést aktualizaci čtečky, neboť modul základny zahajuje stahování OTA aktualizace. Pokud se čísla aktualizací liší, modul základny zahájí OTA aktualizaci používající následujícím postupem:

1. Inicializuje HTTPS klienta a pošle požadavek pro stáhnutí nové verze modulu čtečky, URL adresa nové verze je také v konfiguraci

2. Po zkontrolování hlaviček HTTP požadavku zašle modulu čtečky UPDATE START

3. Přečtená data se posílají postupně modulu čtečky zprávou s typem UPDATE DATA. Zpráva je limitována maximální velikostí zprávy mezi moduly (protokol pro komunikaci mezi moduly je navrhnut přímo pro přístupový systém)

4. Po přečtení a poslání všech dat zašle modul UPDATE END a čeká, zda čtečka potvrdí, že přenos proběhl úspěšně. Pokud tak do časového limitu čtečka neučiní, aktualizace se označí jako neúspěšná, i když update proběhl úspěšně. Jestliže přijde potvrzení, modul základny si uloží do NVS paměti nové číslo verze modulu základny a proces OTA aktualizace končí.

2.11.3 Problémy předchozí implementace

Jedním z problémů předchozí implementace je to, že modul základny má informaci o verzi modulu čtečky, aniž by věděl, že aktualizace opravdu proběhla v pořádku, anebo že proběhla v pořádku, ale pouze nedošla potvrzovací zpráva v časovém limitu. Může se tedy stát, že čtečka zapíše chybně data do OTA oddílu a nebude schopna tento oddíl zavést. O tom však modul základny nebude mít informaci a nový update neproběhne.

Dalším problémem je samotné využívání čísla verze v konfiguraci, které samo o sobě neříká, jaká verze firmware na modulech doopravdy aktuálně běží. To může v extrémním případě vyústit v to, že čísla aktualizaci se úspěšně mění, ale firmware zůstává pořád stejný.

Protože tato implementace nevyužívá rollback funkcionalitu a ani tovární oddíl, není možné v případě chyby, jež zabraňuje nové OTA aktualizaci, zavést oddíl, který by mohl tento problém vyřešit.

Kapitola 3

Návrh optimalizací vzdálených aktualizací přístupových jednotek (OTA)

3.1 Nahrazení čísla běžící verze firmwaru

Nově navrženou změnou, která zkvalitní práci s aktualizacemi modulů, je nahrazení čísel verzí použitím přímo řetězce specifikujícího danou verzi, jenž je do binárního souboru přidán v době kompilace. Tento řetězec lze také zjistit za běhu v aktuální verzi běžící na zařízení. Tímto krokem se vyřeší hned několik problémů. Jako příklad může sloužit to, že nebude docházet k zbytečné aktualizaci, pokud ji není třeba provést. Přidáním zpráv na kontrolu verze modulu čtečky se vyřeší situace, která může nastat, když přenos OTA dat proběhne korektně, ale modul čtečky není schopen zavést novou verzi a zůstane na minulé verzi. Výhodou je, že tento řetězec verze se může zasílat na server, což umožňuje získat přehled, jaká verze na kterém zařízení aktuálně běží. Další velkou výhodou je možnost zkontrolovat, zda je vůbec stahovaný firmware určen pro daný modul. Toto může zabránit situaci, kdy se omylem zamění binární soubor pro modul základy s modulem čtečky nebo naopak.

3.1.1 Návrh nového procesu aktualizací OTA

V nově navrženém způsobu OTA aktualizací modul základny nečte z konfigurace číslo verze a neporovná jej se svým v paměti NVS, ale ihned započne stahování z dané webové adresy, kde se nachází nový binární soubor. Z hlaviček binárního souboru poté modul zjistí data o tom, zda je nová aktualizace určená pro modul základny a jaká verze tato aktualizace je. Pokud není aktualizace určená pro modul základny nebo verze aktualizace je stejná jako ta, kterou už na modulu základny má, dojde k ukončení požadavku. Pokud je verze rozdílná pokračuje se v OTA aktualizaci.

Verze aktualizace se čte z Git tagu commitu a je tedy možné lehce odvodit, jaká verze běží na daném modulu a jaký kód ve verzovacím systému této verzi odpovídá. Textová reprezentace verze odpovídá příkazu „git describe“. [69]

Jednotlivé verze k nasazení pro jednotlivé moduly se označují pomocí Git tagů, které obsahují datum ve formátu `modul_yyyy_MM_dd[rX]`. Kde modul je názvu „main“ pro modul základny nebo „reader“ pro modul čtečky. Znaky yyyy odpovídají roku, MM měsíci 1 až 12 a dd dnu v měsíci. Dále je volitelná přípona začínající r a číslem, jež označuje jednotlivé verze s menšími opravami, které nejsou tak velké, aby byly považovány za novou verzi. Informaci o běžící verzi a posledním commitu zobrazuje modul čtečky na E-INK displeji.

3.2 Nový způsob OTA aktualizace čtečky

Proces aktualizace modulu čtečky probíhá obdobně jako v předchozím případě a pro modul čtečky se při příjmu aktualizačních dat nic nemění. Protože modul základny nemá informaci o verzi, jež běží na modulu čtečky, byla přidána nová zpráva, kterou si modul základny vyžádá verzi modulu čtečky. Na základě této odpovědi modul základny rozhodne, zda má pokračovat v aktualizaci modulu čtečky, nebo ne. Modul základny musí také manuálně parsovat hlavičku binárního souboru a vyčíst z ní informaci o verzi a informaci, pro jaký projekt je binární soubor určen. U aktualizace pro modul základny se toto dělá automaticky a není tak nutné ručně parsovat binární soubor.

3.3 Přidání rollback režimu

Při přidání rollback možnosti by modul po aktualizaci provedl inicializaci a test základních funkcí, jako je komunikace s druhým modulem a stažení inicializační konfigurace. Pokud by vše proběhlo v pořádku, nová verze by se označila jako úspěšně zavedená. Jestliže by se však vyskytl nějaký problém, modul by označil OTA oddíl s novou aktualizací za chybný, vrátil by se do původní verze a zahlásil by chybu serveru. Toto by zabránilo stažení a běhu verze firmware, která by nefungovala správně. Rollback se musí povolit při kompilaci zavaděče, je tedy nutné tuto změnu provést ručně a nelze ji bezpečně provést pomocí OTA aktualizace. Aktualizovat zavaděč pomocí OTA aktualizace je sice teoreticky možné, ale v případě výpadku proudu nebo chyby je nutné zařízení ručně znovu nahrát. Je také nutné předem povolit zápis do míst (CONFIG_SPI_FLASH_DANGEROUS_WRITE), která jsou nebezpečná pro zápis. Jedná se o oddíly běžící aplikace, tabulky oddílů, zavaděče. [78] Proto tato možnost nebyla přidána, i když by byla přínosná v procesu OTA aktualizací modulů.

3.3.1 Přidání továrního oddílu a jeho funkce

Další přínosnou možností by bylo přidání továrního oddílu, který by byl zaveden v případě neúspěšné aktualizace nebo při indikaci v konfiguraci, a v němž by byl záchranný jednoduchý OTA aktualizační

proces, který by byl ověřen a vždy by fungoval. Tento proces by měl být co nejvíce jednoduchý tak, aby byla minimalizována možnost chyby. V ideálním případě by po zavedení továrního procesu byla pouze inicializována síťová periferie a následně se provede vzdálená aktualizace například z pevně dané adresy.

Certifikát by však měl být stále zkontrolován, aby útočník nebyl schopen zavést nebezpečný kód. V případě podepisování binárních souborů by nutnost ověřovat certifikát nebyla tak důležitá, protože by útočník neměl možnost spustit vlastní kód na zařízení. Tato možnost také není v nasazeném přístupovém systému jednoduše použitelná do praktického provozu, protože tovární oddíl lze nahrát pouze ručně a nelze provést nahrání továrního oddílu pomocí OTA aktualizace.[69] [70]

3.4 Aktualizace certifikátu a doba platnosti certifikátu

Jelikož se certifikát pro SSL doporučuje a prakticky se jeho použití ve frameworku ESP-IDF vyžaduje [69], je nutné se zamyslet nad jeho vytvořením a případnou aktualizací v důsledku jeho vyzrazení.

Certifikáty mají svou vlastní platnost nebo používají platnost kořenové certifikační autority. Ve výchozím nastavení se validita certifikátu nekontroluje. Důvodem je to, že mikrokontrolér nemusí mít vždy dostupný reálný čas. Sami vývojáři to doporučují jako jednu z možností, nejedná se však o nutnost. [79] Kontrola času a expirace certifikátu v embedded zařízeních přináší komplexnost, která nemá u provozování přístupového systému, kdy je certifikát centrálně aktualizován a je možné certifikát vyměnit, velký užitek. [80]

Když jsou všechny zařízení pod jednotnou správou, je také možností vytvořit a podepsat si vlastní certifikát s dostatečně dlouhou validitou (například na 20 let). Pokud se požaduje aktualizovat certifikát pomocí OTA aktualizace, jsou tyto možnosti:

1. Vyhradit vlastní oddíl jen pro certifikáty
2. Oddíl se souborovou strukturou SPIFFS, kde budou uložené certifikáty
3. Přiložit certifikáty při kompilaci do binárního souboru dané verze

Poslední možnost je nejjednodušší, ale musí se provést OTA aktualizace celého firmware modulu. Toto u prvních dvou variant není nutné a je možné provést aktualizaci certifikátu za běhu bez nutnosti restartu.

3.5 Souhrn změn vzdálené OTA aktualizace

Změny, které by mohly být přínosné, ale lze je implementovat do stávajícího systému jen s obtížemi, a proto nebyly implementovány:

- Přidání továrního oddílu, který by prováděl záchrannou aktualizaci

- Přidání rollback režimu
- Podepisování aktualizovaného SW

Implementované změny procesu vzdálených OTA aktualizací, jimiž byl rozšířen přístupový systém:

- Nahrazení čísla verzí řetězcem verze z Git pro modul základny
- Nahrazení čísla verzí řetězcem verze z Git pro modul čtečky
  - Vytvoření zpráv pro dotaz na verzi běžící na modulu čtečky
  - Kontrola hlavičky stahované verze pro modul čtečky
- Kontrola, že aktualizace odpovídá danému modulu
- Aktualizace certifikátu vloženého v binárním souboru

Kapitola 4

Implementace rozšíření přístupového systému o možnost použití smart karet

Tato část práce se bude zabývat nutnou implementací změn pro rozšíření stávajícího přístupového systému o novou funkcionalitu vyčítat zašifrovaný obsah ze smart karet MIFARE DESfire a těmito daty bezpečně ověřit uživatele. Implementace je dělána tak, aby zásahy do stávajícího kódu byly minimální.

4.1 Princip autentizace uživatele

Na VŠB - TUO se začaly používat nové smart karty MIFARE DESfire EV1 a EV2. Starší karty typu EV1 jsou používány v případě, že když chce uživatel používat v jedné kartě i služby INKARTA nebo ODISka kvůli zajištění kompatibility služeb ostatních poskytovatelů, kteří podporují pouze verzi karet EV1. Z důvodu, že karty EV1 využívá více poskytovatelů, je na kartách použito více aplikací s vlastními klíči, jimž jsou přiřazena odpovídající práva k souborům a správě aplikací, což karty MIFARE DESfire plně podporují. [81]

Protože UID karet je primárně určeno k rozlišení více karet při antikolizní smyčce a nemělo by se používat k autentizaci uživatele, je na smart kartě MIFARE DESfire uložen zašifrovaný soubor se speciálními daty identifikujícími uživatele. Pro účely autentizace přístupovým systémem má poskytnutý klíč pouze práva k čtení tohoto souboru a žádná další práva nemá (výpis aplikací, souborů, změny ...).

4.2 Návrh čtení šifrovaných dat ze smart karet MIFARE DESfire

Po první analýze návrhu zabezpečeného systému vyplynulo, že je důležité zvolit, kde budou probíhat kryptografické operace:

1. Zabezpečený SAM modul na modulu základny
2. Virtuální SAM modul na serveru
3. Operace se budou provádět přímo na některém modulu přístupové jednotky

První možnost je použít SAM modulu je nejrychlejší a nejbezpečnější možnost návrhu systému. Modul základny obsahuje slot pro použití tohoto SAM modulu v podobě micro SIM (3FF). Přiložená karta by komunikovala s aplikací pomocí tohoto modulu. Modul čtečky by pouze přeposílal data na modul základny, který by odesílal dešifrovaná data z karty na server. Tato možnost by byla nejjednodušší na implementaci, protože by stačilo použít předpřipravené operace výrobce, a také nejvíce zabezpečená.

Přímo pro karty MIFARE DESfire dodává výrobce NXP Semiconductors SAM modul AV3. SAM modul AV3 má certifikaci EAL6+. [66] Využití této varianty však skýtá velkou nevýhodu, kterou je cena za SAM moduly AV3. Odhadované náklady při použití SAM modulu AV3 pro všechny instalované přístupové jednotky se pohybují v částce přes sto tisíc korun. V aplikaci jaké je tento přístupový systém na univerzitě nasazen, je tato částka oproti jiným variantám obtížně obhajitelná. [67] [66] [68]

Druhá možnost představuje na první pohled velmi levnou alternativu k použití SAM modulů. Místo SAM modulu na každé přístupové jednotce by existoval pouze jeden virtuální SAM modul, který by běžel na serveru pro přístupový systém. Modul čtečky by tedy pouze přeposílal přes modul základny zašifrovanou komunikaci karty na přístupový server, kde by probíhala veškerá logika i dešifrování zpráv. Nevýhodou tohoto řešení je nutnost více komunikovat přes síťové rozhraní. Toto řešení bohužel přináší značné zpoždění, které by bylo znatelné, protože při komunikaci s kartou by muselo provést nejméně 5 požadavků na přístupový server.

Seznam požadavků nutných k ověření uživatele za předpokladu, že by se kryptografické operace prováděly na přístupovém serveru:

- 1x požadavek pro generování diverzifikovaného klíče
- 2x požadavek pro vyjednání session (dočasného pro jednu interakci) klíče
- 1x požadavek pro zvolení aplikace
- 1x požadavek pro načtení dat

Třetí možností je provádět kryptografické operace na modulu čtečky. Po přiložení přístupové karty by se přečetlo UID karty, to by bylo zasláno na přístupový server, který by vygeneroval diverzifikovaný klíč na základě UID přiložené smart karty, jenž by byl poslán modulu čtečky přes modul základny. Ostatní operace by se prováděly na modulu čtečky a pouze výsledek operací by byl zaslán přístupovému serveru k vyhodnocení. Komunikace s přístupovým serverem by byla po celou dobu zašifrovaná. Tato možnost zajistí rychlost velmi podobnou jako při použití modulu SAM, aniž by je

bylo nutné pořizovat. Nevýhodou tohoto řešení je, že se s diverzifikovaným klíčem pracuje v modulu čtečky v nezašifrované podobě. Je tedy hypoteticky možné tento diverzifikovaný klíč získat. Aby tato možnost nastala, útočník by musel získat fyzicky přístup k modulu čtečky a určitým způsobem zajistit přečtení diverzifikovaného klíče. I kdyby se mu tohoto obtížně získatelného přístup podařilo dosáhnout, dostal by pouze jeden diverzifikovaný klíč karty, který by umožnil napadnout pouze jednoho uživatele.

Pro použití v této práci s přihlédnutím na dané možnosti a možnost implementace do stávajícího přístupového systému byla po konzultaci zvolena třetí možnost řešení čtení bezkontaktních karet MIFARE DESfire. První předkládaná varianta byla vyloučena z důvodu finanční nákladnosti a druhá na základě předběžných testů, které ukázaly na pomalost takového řešení. Při použití druhé možnosti bylo předběžně spočítáno, že jeden dotaz na přístupový server by trval okolo 250 ms, uživatel by tedy musel čekat minimálně dvě sekundy na ověření.

Pro rozšíření přístupového systému o funkci čtení šifrovaných dat z karet MIFARE DESfire musely být provedeny tyto kroky:

1. Implementace podpory komunikace mezi modulem čtečky a kartami MIFARE DESfire pomocí PN532.
2. Implementace příkazů a šifrování pro MIFARE DESfire
3. Vícekroková komunikace s kartou, kdy jsou data prvně načtena a diverzifikovaný klíč se vypočítá na základě těchto dat na přístupovém serveru
4. Úpravy pro vícekrokovou autentizaci na přístupovém serveru
5. Vytvoření programu pro vypočtení diverzifikovaného klíče

4.3 Návrh komunikace pro podporu výměny dat mezi modulem PN532 a smart kartami MIFARE DESfire

Pro zajištění komunikace s bezkontaktní smart kartou bylo nutné rozšířit funkce modulu čtečky pro kompletní komunikaci s modulem PN532. Protože je komunikace s kartou MIFARE DESfire proprietární a je nutné podepsat dohodu o mlčenlivosti (NDA), bylo nutné využit mnohé neoficiální zdroje. [82][83][84][85][86][87][88] V původním kódu se používal modul PN532 pouze k čtení UID přiložené smart karty. Ke zprovoznění komunikace s kartou, která se provádí v modulu PN532 pomocí příkazu nazvaného InDataExchange, musely být provedeny tyto kroky:

1. Doplnění komunikace s modulem PN532 o čtení stavových bitů
2. Potvrzování a negativní potvrzování rámců

3. Plné parsování informací o kartě – ATQA, SAK, ATS

4. Implementace příkazu InDataExchange

Původní program modulu čtečky neimplementoval plnou komunikaci s modulem PN532 přes rozhraní SPI. Modul PN532 má pět typů rámců, ze kterých bylo nutné implementovat následující tři:

1. Normální informační rámec

2. Potvrzovací rámec – ACK

3. Chybový rámec

Rámce modulu PN532 při použití rozhraní SPI mají modifikovanou strukturu, kdy se pomocí prvního bajtu indikuje operace, která má být provedena. Operacemi mohou být čtení stavu modulu, čtení dat a zápis dat. Před operací čtení dat je nutné zkontrolovat stav modulu PN532. Modul PN532 odpoví bajtem indikujícím, zda je možné přečíst data, nebo se musí čtení stavu opakovat. Protože je délka odpovědí variabilní, je nutné čtení dat z modulu PN532 rozdělit na dva kroky, kdy se nejprve přečte část s délkou odpovědi a až poté se dočte zbytek dat.

Přidáno bylo také potvrzování rámců, které dříve nebylo potřeba, protože se z modulu PN532 pouze četlo UID přiložené smart karty. Doplněny byly též informace o ATQA, SAK, ATS, jež poskytují informace o kartě.

Nakonec byl pro modul PN532 implementován příkaz InDataExchange, který umožňuje vyměňovat data s kartou pomocí modulu PN532. S touto funkčností lze přejít k dalšímu kroku, a to ke komunikaci s kartou MIFARE DESfire. Tyto změny jsou implementovány v komponentě s názvem „pn532“ ve zdrojových kódech modulu čtečky (\reader_module \components \pn532).

4.3.1 Implementace komunikace se smart kartou DESfire

Komunikace s kartou MIFARE DESfire je možná buď pomocí proprietárního protokolu MIFARE APDU nebo dle protokolu APDU definovaného ISO standardem 7816-4. Primárně použitý protokol APDU je MIFARE nejen z důvodu částečně lepší kompatibility s kartami typu EV1, ale i protože ISO 7816-4 nedefinuje příkaz GetCardUID, který je možné volitelně použít (ve verzi EV1 není implementována struktura souboru definovaná ve standardu ISO 7816-4 a také se používá pro ISO autentifikaci jiný algoritmus, a to 3DES, jenž nesplňuje požadovanou úroveň zabezpečení [89]).

Pro karty MIFARE DESfire byly implementovány tyto příkazy:

- DESFIRE_GET_APPLICATIONIDS (0x6A) – výpis AID aplikací
- DESFIRE_AUTHENTICATE_AES (0xAA) – autentifikace pomocí algoritmu AES

- DESFIRE_SELECT_APPLICATION (0x5A) – výběr aplikace pomocí AID
- DESFIRE_GET_FILE_IDS (0x6F) – výpis souborů vybrané aplikace
- DESFIRE_READ_DATA (0xBD) - přečtení dat souboru
- DESFIRE_CARD_UID (0x51) – výpis reálné UID karty po autentifikaci
- DESFIRE_ADDITIONAL_FRAME (0xAF) – dlouhé příkazy lze posílat po menších částech, tímto příkazem si karta též vyžádá zbývající data

Tyto příkazy jsou implementovány v příloze a to v kódu modulu čtečky v souboru „desfire.c“ komponenty s názvem „pn532“ (\reader_module \components \pn532 \desfire.c). Ostatní příkazy je možné jednoduše doplnit, ale pro potřeby ověření pomocí těchto bezkontaktních karet nejsou aktuálně potřeba. Zásadní pro funkčnost ověření uživatele je číst data ze zašifrovaných souborů, a tedy být schopen si ověřit přístup. Veškerá funkcionalita pracující s technologií NFC, komunikace s modulem PN532 a s kartami MIFARE DESfire byla vytvořena jako komponenta projektu, která není závislá na zbytku projektu.

4.3.2 Proces ověření s bezkontaktní kartou MIFARE DESfire

K ověření se využívá šifrovací algoritmus AES 128 bitů. Pro zahájení ověření se posílá příkaz DESFIRE_AUTHENTICATE_AES a následuje parametr číslo klíče pro danou aplikaci (klíčů může být v EV1 až 14 pro jednu aplikaci). Pokud daný klíč není určen pro šifrovací algoritmus AES, karta odpoví chybou s obsahem 0xAE .

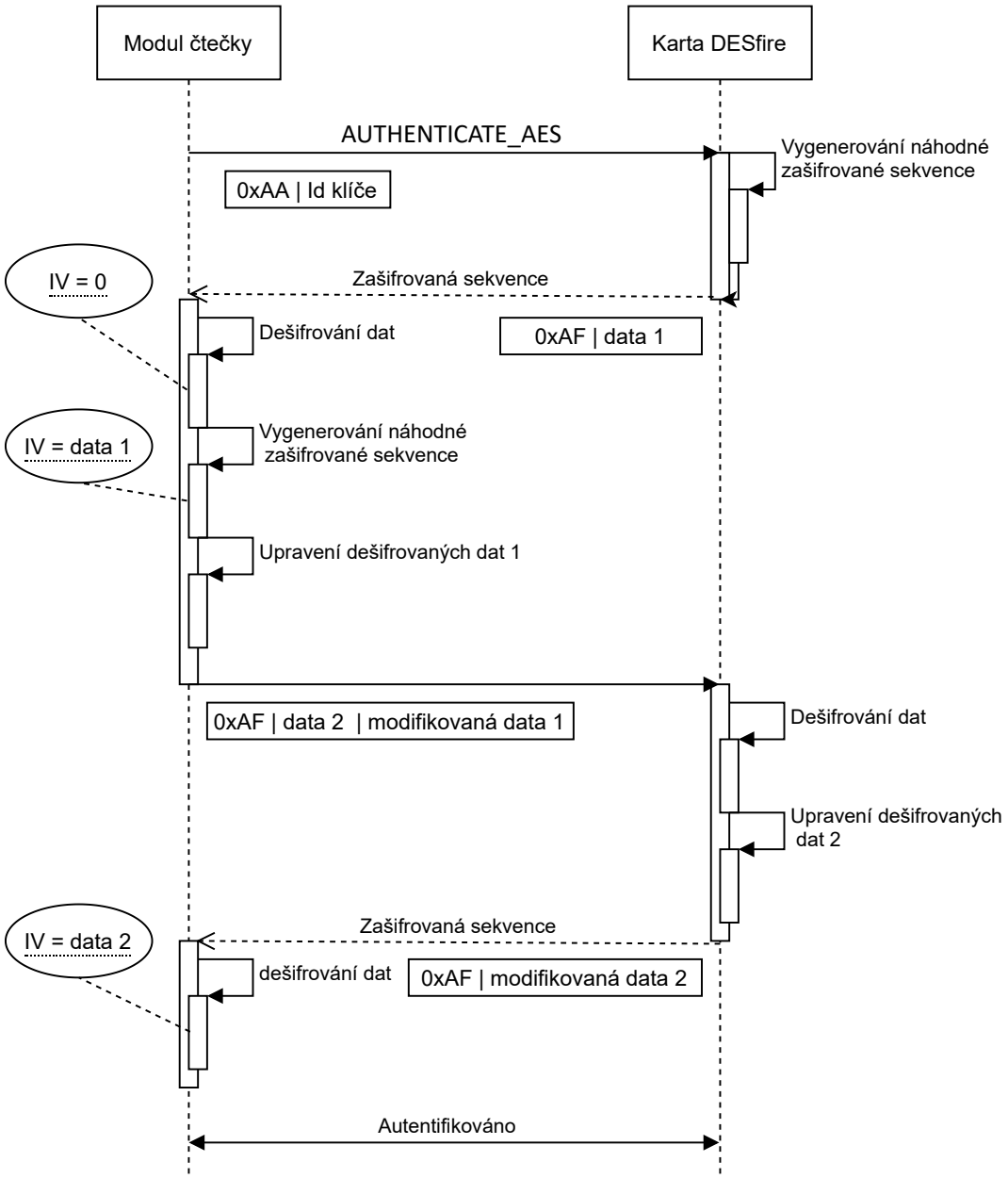
Bezkontaktní karta vytvoří náhodnou 16 bajtovou sekvenci a zašifruje ji pomocí vybraného klíče. Bezkontaktní karta odpoví na příkaz bajtem 0xAF (Additional frame) a zašifrovanou sekvencí.

Modul čtečky přijme zašifrovaná data, pomocí svého diverzifikovaného klíče je rozšifruje a získá původní náhodnou sekvenci. Inicializační vektor pro dešifrování je nastaven na samé nuly. Modul čtečky vytvoří 16 bajtovou náhodnou sekvenci. Tuto náhodnou sekvenci spojí s dešifrovanou přijatou sekvencí, která se otočí o jeden bajt doleva (operace ROL), a vytvoří tak 32 bajtovou sekvenci o dvou částech. Tato sekvence je modulem čtečky zašifrována pomocí algoritmem AES, kde inicializační vektor pro algoritmus AES je přijatá náhodná sekvence karty ještě před šifrováním.

Karta přečte 32 bajtovou sekvenci a dešifruje ji pomocí algoritmu AES. Zkontroluje, zda souhlasí druhá část přijaté sekvence (16 bajtů) se sekvencí, kterou karta poslala na začátku. Pokud sekvence odpovídají, karta získává informaci, že uživatel má správný klíč. Karta otočí o jeden bajt doleva první části, zašifruje pouze tuto část a odešle zpět.

Modul čtečky přijme 16 bajtovou zašifrovanou část. Dešifruje ji s inicializačním vektorem nastaveným na posledních 16 bajtů odeslaných kartě. Modul čtečky zkontroluje, zda je shodná dešifrovaná sekvence se sekvencí, kterou vygeneroval. Pokud vše proběhne úspěšně, obě strany mají informaci, že opačná strana disponuje správným klíčem. Tím je dokončeno ověření, které je také zachyceno

na obrázku 4.1 a pro další komunikaci se vygeneruje session klíč, jenž je sestaven z vygenerovaných náhodných částí. První čtyři bajty náhodné sekvence karty, pak 4 první bajty náhodné sekvence čtečky a poté následují 4 poslední bajty náhodné sekvence čtečky a dále 4 poslední bajty sekvence čtečky. Tento session klíč se používá pouze po dobu trvání ověření a při novém ověření se vždy generuje nový klíč. Proces ověření je detailně popsán v příloze E.



Obrázek 4.1: Příklad procesu autentizace

4.3.3 Šifrování příkazů

MIFARE DESfire má tři režimy posílání příkazů:

1. Nezabezpečená (plain)
2. Nešifrovaná komunikace zabezpečená CMAC
3. Šifrovaná

Nezabezpečená komunikace se používá do doby, než se provede ověření. Po ověření je nutné používat buď CMAC režim nebo plné šifrování. To, zda se musí použít šifrovaná komunikace, nebo postačuje CMAC a nezabezpečený režim, záleží na typu příkazu a u souborů také na nastavení přístupu jednotlivých souborů.

V režimu CMAC příkazy procházejí CMAC výpočtem a aktualizují inicializační vektor. Odpověď karty obsahuje 8 bajtový CMAC. CMAC odpovědi se tvoří z celé odpovědi kromě stavového bajtu. Tímto lze ověřit, že data nebyla změněna v průběhu komunikace a neprobíhá žádný útok typu „man in the middle“.

V šifrovaném režimu se nevyužívá CMAC, místo toho se zprávy blokově šifrují. Zprávy musí být rozděleny na části a případně doplněny o zarovnání ve tvaru 0x80 a zbytek 0x00. Ne vždy se šifruje celý příkaz, některé příkazy mají zašifrovanou pouze citlivou část (například data k zapsání). V případě čtení ze souborů na smart kartě se také využívá kontrolní součet CRC, který se přidává na konec zprávy před šifrováním. Pro CRC algoritmus je využit CRC-32 s polynomem 0xED88320 definovaný například v CRC-32-IEEE802.3 (pro Ethernet) nebo HDLC.

Ověření trvá do doby, než nastane chyba (přijde zpráva o chybě), nebo se změní vybraná aplikace.

4.3.4 Výpočet CMAC

Výpočet CMAC používá session klíč a session inicializační vektor. Před výpočtem CMAC je nutné vytvořit dva podklíče, které se ve výpočtu CMAC používají.

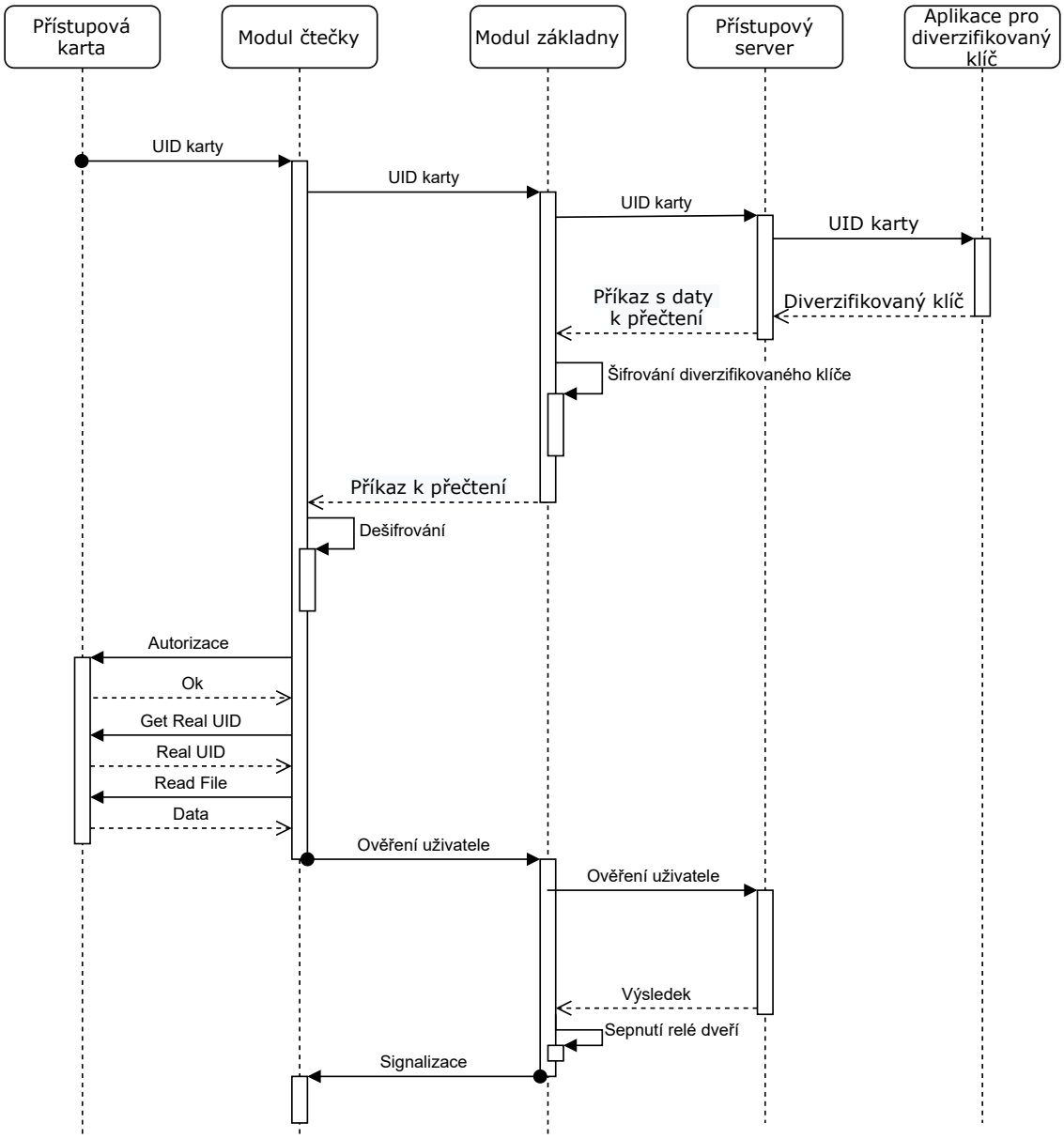
První z nich se vygeneruje prostřednictvím zašifrování bloku samých nul pomocí session klíče a inicializačního vektoru nastaveného na samé nuly. Zašifrovaný výsledek se poté posune o bit doleva. Pokud je při posunu ztracena jednička (jednička na nejvyšším bitu), provede se exkluzivní součin (XOR) posledního bajtu s číslem 0x87.

Druhý podklíč se tvoří z prvního podklíče tak, že se opět posune o jeden bit doleva a při přesunutí jedničky mimo se provede exkluzivní součin s číslem 0x87.

Výpočet CMAC se zahájí zarovnáním na 16 bajtové bloky. Zarovnání je opět první bajt 0x80 a další bajty 0x00. Pokud nebylo zarovnání přidáno, použije se první podklíč, jinak se použije druhý podklíč. Provede se exkluzivní součin posledního bloku dat s podklíčem. Poté se zašifrují data pomocí AES pomocí session klíče a session inicializačního vektoru. Zašifrovaný poslední blok je CMAC kód, který se použije pro aktualizaci session inicializačního vektoru a prvních osm bitů se přidá k odpovědi karty v režimu CMAC.

4.4 Vícekroková ověření uživatele

Modul čtečky nově obsahuje vícekrokovou logiku, kdy je při prvním přiložení vyčteno UID karty a je posláno modulu základny, který zprávu přeposílá přístupovému serveru. Poté se v daném časovém limitu čeká na odpověď přístupového serveru, zda se úspěšně zpracují odeslaná data. Na základě obdržené odpovědi od přístupového serveru se provede požadovaná operace. Kód implementující tuto logiku je v příloze a to v souboru „nfc.c“ zdrojového kódu modulu čtečky (\reader_module\main \nfc.c). Tato logika je zobrazena na sekvenčním diagramu obrázek 4.2.



Obrázek 4.2: Sekvenční diagram nově navrženého procesu ověření

V systému jsou implementovány čtyři typy odpovědí z přístupového serveru:

1. Povolení autentizace na základě UID (nepokračuje se komunikace s kartou)
2. Vyčtení dat ze souboru
3. Vyčtení dat příkazu k získání reálného UID karty
4. Vyčtení dat ze souboru a reálného UID karty

Modul základny nemá informaci o stavu komunikace s kartou. Modul základny sestavuje zprávy serveru ve formátu JSON a překládá odpovědi serveru z JSON na zprávy pro modul čtečky.

Příklad zprávy v JSON je ve výpisu 4.1, kde proměnná result a type indikují typ operace, kterou má přístupová jednotka vykonat. Result obsahuje, zda se má provést další komunikace s kartou, nebo jestli je zamítnut či povolen přístup do místnosti. Proměnná type je v odpovědi pouze v případě, že result označuje další komunikaci se smart kartou, a indikuje, jaký typ komunikace se smart kartou má být proveden. Proměnná divKey obsahuje diverzifikovaný klíč jako řetězec v šestnáctkové číselné soustavě. Proměnná file je identifikátor souboru k přečtení. Proměnná fileLen indikuje, kolik bajtů dat ze souboru se má přečíst, a appId je identifikátor aplikace k vybrání.

```
{
      'result' => 3,
      'divKey' => "e24ce7cfad1dcdc044f10e4a8dbd1b6d",
      'file' => 4,
      'fileLen' => 7,
      'type' => 3,
      'appId' => 5267225
}
```

Výpis 4.1: Příklad zprávy pro vyčtení dat souboru a reálného UID karty

Vyčtená data z modulu čtečky jsou po provedení operací určených v příchozí zprávě opět zaslána k vyhodnocení přístupovému serveru, který na základě těchto dat rozhodne o povolení přístupu. Všechny operace vyžadující další komunikaci s kartou obsahují identifikaci aplikace k výběru (AID) a diverzifikovaný klíč k ověření. Poté, pokud je přístupovým serverem požadována operace přečtení souboru, se odešle číslo souboru a délka dat k přečtení. Průběh komunikace je uživateli indikován jemným modrým podbarvením, které se postupně zvyšuje na základě toho, kolik operací chybí do dokončení ověření.

Při komunikaci základny a čtečky, která je v normálním provozu nešifrovaná, je nutné zašifrovat citlivá data, jakými je hlavně diverzifikovaný klíč. To se provádí pomocí symetrického AES šifrování. Inicializační vektor pro zašifrování dat se tvoří z času přiložení karty, jenž se zasílá přístupovému

serveru. Jeho obsah má tedy k dispozici jak modul čtečky, tak i modul základny. U inicializačního vektoru není nutné, aby byl šifrován, a proto může být inicializační vektor znám i útočníkovi. Klíč k dešifrování dat je uložen v binárních datech firmware modulů a je tedy možné, aby byl vyměněn pomocí OTA aktualizace.

4.5 Úpravy přístupového serveru

Jelikož je hlavní skript přístupového serveru napsaný v jazyce PHP, bylo poměrně komfortní udělat potřebné změny. Bylo nutné přidat logiku pro získání diverzifikovaného klíče a rozšířit odpovědi o požadavek na přečtení souboru a reálného UID. Diverzifikovaný klíč pro danou kartu se získává pomocí HTTP dotazu, který vrací diverzifikovaný klíč na základě dodaného UID.

4.6 Aplikace k odvození diverzifikovaného klíče

Aplikace je napsaná v jazyce C a využívá části open source knihovny libfreefare[1]. Aplikace pomocí klíče a UID odvozuje diverzifikovaný klíč pro danou kartu. Nad ní je PHP skript, který přijímá HTTP požadavky s UID a vrací JSON s diverzifikovaným klíčem. Tuto aplikaci jsem dostal již vypracovanou a nepodílel jsem se zásadně na její tvorbě, avšak se znalostí, na jakém principu aplikace funguje, by pro mě nebyl zásadní problém vytvořit vlastní implementaci této aplikace.

Vhodným vylepšením aplikace by bylo zašifrování hlavního klíče pomocí algoritmu AES-256, jenž by se dešifroval v aplikaci po zadání hesla (klíče k algoritmu AES-256). Dokud by nebylo zadáno heslo, aplikaci by nešlo použít a klíč by byl bezpečně zašifrovaný. Poté by se dešifrovaný klíč nacházel nezabezpečený v paměti RAM, protože by byl aplikací používán a musel by být pro aplikaci čitelný. Tím by se zajistilo, že by nestačilo stáhnout pouze aplikaci v binární podobě, z které by se dalo zjistit hlavní klíč.

Z důvodu zajištění vyšší bezpečnosti systému by bylo vhodnější přesunout tuto aplikaci na speciální zabezpečené USB zařízení (např. token), ze kterého by tento klíč nebylo možné získat. Taková zabezpečená USB zařízení existují v ceně do tisíce korun a neměl by být značný problém přesunout aplikaci na takovéto zařízení. Problém vzniká, pokud je server virtualizován anebo funguje v HA režimu.

4.7 Výsledek prvotní implementace

Výše popsaný návrh rozšíření stávajícího přístupového systému byl naprogramován a byl zcela funkční, ale při prvním použití šifrovaného transportu zpráv přes nově používaný protokol HTTPS se objevil značný problém, jenž spočíval v rychlosti odezvy ověření uživatele. Doba mezi přiložením

[1] https://github.com/nfc-tools/libfreefare

karty a otevřením dveří byla vždy delší než 2 sekundy, což je pro přístupový systém používaný velkým počtem uživatelů i několikrát denně velmi nekomfortní. Proto bylo přistoupeno k podrobnému testování systému a navržení optimalizací tak, aby přístupový systém byl použitelný v reálném provozu a dosahoval stejných parametrů jako původní systém před rozšířením.

Kapitola 5

Testování a optimalizace systému

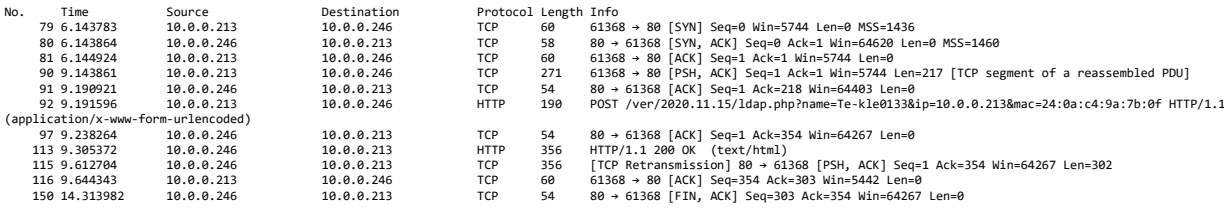
V této kapitole se práce zabývá návrhem možných optimalizací celého systému, tak aby bylo rozšířený přístupový systém možné používat v normálním provozu. Cílem je, aby rychlost před změnami a po nových úpravách byla srovnatelná a nedošlo tedy k žádné degradaci uživatelského komfortu při implementaci vyšší úrovně zabezpečení.

5.1 Základní analýza odezvy systému

Použití smart karet vyžaduje používat zabezpečenou komunikaci všech modulů. Po změně původního protokolu HTTP na zabezpečený protokol HTTPS při komunikaci s přístupovým serverem pro ověřování uživatele bylo zjištěno nejen značné zpomalení systému, ale i náhodné velké zpoždění odezvy pro uživatele, které dosahovalo ve výjimečných případech až deset sekund. Tento problém bylo nutné vyřešit.

V první fázi se provedla analýza obecného využití jednotlivých jader mikrokontroléru. Ta neukázala, že by docházelo k nedostatku výpočetního výkonu v průběhu běhu programů na jednotlivých modulech. Dokonce 95 % času byla jádra procesoru ve stavu nečinnosti (Idle). K měření byly použity nástroje FreeRTOS a to pomocí funkce „vTaskGetRunTimeStats“. Pro její použití musí být při kompilaci povolený sběr dat pro tuto funkci. Zároveň byla také zkontrolovaná volná heap paměť a velikosti zásobníků jednotlivých úloh a jejich využití. U obou sledovaných parametrů se ukázalo, že mají dostatečné rezervy. Výsledky této analýzy neposkytly prostor pro vylepšení SW pro moduly čtečky a základny.

Dále se přistoupilo k prozkoumání síťové komunikace nově implementovaného protokolu HTTPS pomocí aplikace Wireshark, logů z knihovny MbedTLS a knihovny lwIP z modulu základny. Kde je MbedTLS knihovna pro protokol TLS používaná ve frameworku ESP-IDF a lwIP implementuje protokol TCP/IP a je určený pro aplikace s omezenými zdroji. Bylo nalezeno, že velkou část časové prodlevy způsobovala inicializace TSL (handshake) protokolu pro protokol HTTPS, kdy výměna dat k šifrování a poté postupné dešifrování na ESP32 způsobovalo příliš velkou časovou prodlevu.



No.	Time	Source	Destination	Protocol	Length	Info
79	6.143783	10.0.0.213	10.0.0.246	TCP	60	61368 → 80 [SYN] Seq=0 Win=5744 Len=0 MSS=1436
80	6.143864	10.0.0.246	10.0.0.213	TCP	58	80 → 61368 [SYN, ACK] Seq=0 Ack=1 Win=64620 Len=0 MSS=1460
81	6.144924	10.0.0.213	10.0.0.246	TCP	60	61368 → 80 [ACK] Seq=1 Ack=1 Win=5744 Len=0
90	9.143861	10.0.0.213	10.0.0.246	TCP	271	61368 → 80 [PSH, ACK] Seq=1 Ack=1 Win=5744 Len=217 [TCP segment of a reassembled PDU]
91	9.190921	10.0.0.246	10.0.0.213	TCP	54	80 → 61368 [ACK] Seq=1 Ack=218 Win=64403 Len=0
92	9.191596	10.0.0.213	10.0.0.246	HTTP	190	POST /ver/2020.11.15/ldap.php?name=Te-kle0133&ip=10.0.0.213&mac=24:0a:c4:9a:7b:0f HTTP/1.1 (application/x-www-form-urlencoded)
97	9.238264	10.0.0.246	10.0.0.213	TCP	54	80 → 61368 [ACK] Seq=1 Ack=354 Win=64267 Len=0
113	9.305372	10.0.0.246	10.0.0.213	HTTP	356	HTTP/1.1 200 OK (text/html)
115	9.612704	10.0.0.246	10.0.0.213	TCP	356	[TCP Retransmission] 80 → 61368 [PSH, ACK] Seq=1 Ack=354 Win=64267 Len=302
116	9.644343	10.0.0.213	10.0.0.246	TCP	60	61368 → 80 [ACK] Seq=354 Ack=303 Win=5442 Len=0
150	14.313982	10.0.0.246	10.0.0.213	TCP	54	80 → 61368 [FIN, ACK] Seq=303 Ack=354 Win=64267 Len=0

Obrázek 5.1: Záznam logu z programu Wireshark

Vizte výpis z logu Listing 5.1, kde je první část začátek navazování spojení. Další část je ukončení spojení. Mezi začátkem handshake a dokončením je doba 6 sekund. HTTPS požadavek je dokončen za další 1,5 sekundy. Toto zpoždění je také zachyceno na výpisu 5.1. Až dvousekundový handshake TLS však i dle vývojářů ESP32 není výjimečný a dá se v čipech ESP32 očekávat. [90]

```
I (167509) mbedtls: ssl_tls.c:8084 => handshake
I (167509) mbedtls: ssl_cli.c:3510 client state: 0
...
I (173289) mbedtls: ssl_cli.c:3621 handshake: done
I (173299) mbedtls: ssl_cli.c:3510 client state: 15
...
I (174909) eth_module_https: HTTP_EVENT_JSON_ON_DATA, len=102
```

Výpis 5.1: Log při navazování TSL spojení

Dalším nalezeným problémem, který ovlivňoval rychlost odezvy systému, bylo používání standardně zapnutého Nagleova algoritmu pro TCP spojení, který redukuje počet paketů tak, že pakety postupně akumuluje a posílá najednou ve větších celcích. Toto řešení ušetří režii posílání jednotlivých malých paketů, kterou přináší protokolu TCP a IP hlavičky. Nevýhodou je však zpomalení rychlosti odezvy, kdy algoritmus čeká na naplnění buferu s daty, když nejsou potvrzeny všechny předchozí pakety. Ve frameworku ESP-IDF je algoritmus standardně implementován pro velikost 1436 bajtů, což je maximální velikost segmentu. [91] Pro zařízení pracující v režimu reálného času není tento algoritmus vhodný z důvodu zajištění rychlé odezvy. [92]

5.2 Možnosti optimalizace

Možností, jak zrychlit pomalou odezvu systému, bylo nahradit nově použitý protokol HTTPS jiným protokolem. A to takovým protokolem, který by nepotřeboval opakovanou inicializaci šifrování jako bezstavové HTTPS, ale zároveň by byl stále zabezpečený a nejlépe takový, aby nebylo nutné přepisovat podstatnou část kódu pro přístupový server.

Možností řešení jsou klasické datagramy UDP spolu s protokolem DTLS. DTLS je stavový protokol, který poskytuje zabezpečení UDP pomocí TLS. Bohužel v prostředí frameworku ESP-IDF není pro protokol TLS stabilní implementace. Další možností je využít čisté TCP spojení s protokolem TSL pro šifrování. Tato možnost by byla pravděpodobně nejrychlejší, ale znamenala by netriviální změny nejen v kódu přístupového serveru, ale také v kódu modulu základny, a to z důvodu nutnosti řešit implementaci použití protokolu TSL ručně, jelikož neexistovala jednoduchá stabilní podporovaná knihovna pro ESP-IDF. Další důvodem k nepoužití protokolu TCP byly negativní zkušenosti s použitím funkce keepalive a složitějším udržováním spojení v předchozí generaci přístupového systému.

Dalším možným řešením pro šifrovaný komunikační protokol je použití protokolu WebSocket, který umožňuje šifrovaný přenos a používá protokol TCP. Protokol WebSocket používá ohraničené zprávy a ne proudy dat, jak tomu je v protokolu TCP, což umožňuje snadnější implementaci do již existujícího kódu. Nevýhodou nasazení protokolu WebSocket ve frameworku ESP-IDF bylo to, že protokol WebSocket je podporován od verze 4.0, bylo tedy nutné povýšit verzi frameworku ESP-IDF z verze 3.3 na verzi 4.02, která je stabilní verzí s dlouhou podporou (long term). Změnu verze frameworku by však bylo stejně nutné v budoucnu udělat, protože framework ESP-IDF ve verzi 3.3 je podporován pouze do ledna roku 2022. [93]

Velkou výhodou protokolu Websocket je jednoduchost použití v nové verzi frameworku ESP-IDF, srovnatelná rychlost s TCP, velmi snadné nasazení a menší množství změn na straně přístupového serveru. Další dobrou vlastností protokolu WebSocket v prostředí frameworku ESP-IDF je implementace znovu navázání připojení a udržování spojení pomocí výměny kontrolních zpráv typu PING a PONG.

5.2.1 Aktualizace frameworku ESP-IDF na verzi 4

Nově použitý framework ESP-IDF ve verzi 4.02 má s verzí 3.3, která byla původně použita pro vývoj SW, několik zpětně nekompatibilních změn. Jednou ze změn bylo přejmenování hlavičkových souborů, jejich reorganizace a také odstranění implicitně přidávaných hlavičkových souborů. [94]

Byl nahrazen systém kompilace pomocí programu Make novým programem CMake a zcela změněny konfigurační a pomocné skripty napsané v jazyce Python do jednoho skriptu idf.py, který obsluhuje veškeré operace spojené s mikrokontrolérem ESP32. Při použití programu CMake je jednodušší vytvářet vlastní komponenty, čehož bylo využito v projektu pro modul čtečky k oddělení částí kódů obsluhující fonty, práci s E-INK displejem a komunikaci s modulem PN532. Toto řešení usnadňuje orientaci v kódu a snižuje závislosti kódů mezi sebou. Spolu s verzí 4.0 je nově možné jednodušeji použít jiné IDE než Eclipse. Pro IDE Eclipse byl vydán nový doplněk, který značně zjednodušuje práci s ESP32 a frameworkem ESP-IDF. [95]

Dalšími novinkami v nové verzi jsou podpora protokolu WebSocket a také nová komponenta pro Ethernet. Protože staré API pro práci s Ethernetem bylo nahrazenou novou komponentou,

která není zpětně kompatibilní, musela se přepsat část kódu pro práci s Ethernetem. Změny se týkaly inicializace rozhraní Ethernetu, event handleru a získání DHCP informací z nového rozhraní nazvaného NETIF. [94]

5.2.2 Použití protokolu WebSocket

Použití protokolu WebSocket v prostředí frameworku ESP-IDF verze 4.02 je velmi jednoduché. Po inicializaci stačí zadat URL adresu, certifikát a lze zasílat data druhé straně pomocí dvou funkcí, přičemž jedna z nich je pro binární data a druhá pro data textová. Protože se posílají data ve formátu JSON, používá se funkce pro posílání textových dat.

Vše ostatní pro protokol WebSocket řeší takzvaný event handler. Toto je funkce, která obsluhuje události, zde konkrétně události z daného spojení WebSocket. Jedná se o události připojení, odpojení, chyby a přijetí dat. Přijatá data můžou být různého typu. Definované kontrolní rámce jsou: CLOSE (0x8), PING (0x9) a PONG (0xA). Datové rámce se poté předávají úloze obsluhující spojení WebSocket pomocí fronty zpráv. Úloha vybírá jednotlivé datové zprávy ve formátu JSON, zpracovává je, případně zprávy přeposílá v binárním formátu modulu čtečky a na jejich základě spíná relé elektromagnetického zámku dveří.

SW modulu čtečky zůstal při použití nového protokolu WebSocket nezměněn. Toto řešení je možné, protože veškerou logiku komunikace s přístupovým serverem zajišťuje modul základny.

5.2.2.1 Optimalizace pomocí prostředí NodeJS jako serveru pro protokol WebSocket

Pro obsluhu protokolu WebSocket na straně přístupového serveru bylo zvoleno prostředí NodeJS, což je osvědčené a rychlé prostředí nejenom pro webové servery. NodeJS zajišťuje, že i při velkém počtu spojení protokolu WebSocket a velkém počtu požadavků bude systém stabilní a bude mít rychlou odezvu.

Protože hlavní logika přístupového serveru je napsaná v PHP, aplikace v NodeJS posílá požadavky na lokální PHP skripty pomocí HTTPS protokolu. Aplikace má tedy jedinou funkci, a to obsluhovat WebSocket spojení a předávat zprávy dále ke zpracování. V poslední iteraci optimalizací aplikace v NodeJS posílá přímo požadavek aplikaci pro diverzifikaci klíče a sestavuje odpověď modulu základny ve formátu JSON.

5.2.3 Ostatní optimalizace systému

Dále byly řešeny optimalizace SW modulů, a to úpravy priorit úloh běžících na modulech. Úlohy, které jsou nutné pro rychlou odezvu, mají nově zvýšenou prioritu a neprioritní úlohy, které se můžou vykonat kdykoliv, mají naopak prioritu sníženou. Na základě výsledku testování byly upraveny časy čekání blokujících operací a některá zbytečně dlouhá čekání na blokující operace byla odstraněna.

Další část optimalizací se zaměřila na komunikaci mezi moduly čtečky a základny přes rozhraní UART. Byla otestována a následně zvýšena přenosová rychlost přes sběrnici RS422 na 256 000 Bd/s

a také byla optimalizována velikost zpráv, aby byla využita maximální velikost zpráv z důvodu minimalizace fragmentace zpráv.

Dále byly upraveny kompilační konfigurace tak, aby byly povoleny všechny nové optimalizace a vypnuty funkce, které se nepoužívají. Týká se to například kompilace s nastavením „-O2“, ponechání jen nejnutnějších logů, vypnutí kontroly přetečení paměti a podobně.

Jak již bylo uvedeno výše, optimalizace proběhly v několika iteracích. Hlavní změnou byla komplexní implementace protokolu WebSocket. V následujících iteracích byly optimalizovány priority úloh, časy čekání na blokující operace, rychlosti komunikace na různých rozhraních a další.

5.3 Systematické testování

Aby bylo možné různé varianty optimalizace komunikace mezi jednotlivými moduly systému a dalšího nastavení porovnat a následně zjistit, která změna, jak ovlivňuje odezvu systému, bylo nutné přejít k podrobnějšímu měření času všech operací v celém přístupovém systém.

Do systému byly pro účely testování přidány časové značky do určitých části kódů zajišťujících proces ověření uživatele. Modul základny používá reálný čas pomocí SNTP protokolu, modul čtečky však dostává tento čas se zpožděním, protože se přeposílá přes rozhraní UART a není implementována žádná kompenzace pro toto možné zpoždění. Z tohoto důvodu není možné na čas v modulu čtečky plně spoléhat. Měření bylo provedeno s využitím uložení času prvního přečtení karty a od tohoto časového bodu se měřily časy vykonávaných operací modulu čtečky. Pro modul základny se použil čas přijetí zprávy s UID karty. Čas byl zaznamenán v mikrosekundách od startu mikrokontroléru ESP32.

Byly definovány tyto časové značky:

1. pro modul základny:
   (a) Detekce nově přiložené smart karty
   (b) Před zasláním UID karty modulu základny
   (c) Po zaslání UID modulu základny
   (d) Nová NFC zpráva z modulu základny
   (e) Přijatý požadavek ze základny na přečtení dat karty
   (f) Autentifikace DESfire karty
   (g) Přečtení reálného UID karty (volitelné)
   (h) Autentifikace DESfire karty (volitelné)
   (i) Přečtení dat karty
   (j) Před zasláním dat karty modulu základny
   (k) Po zaslání UID karty modulu základny

(l) Signalizace povolení/odmítnutí přístupu

2. pro modul základny:

   (a) NFC UID data z modulu čtečky
   (b) Parsování UID ze zprávy modulu čtečky
   (c) Zahájení poslání UID přístupovému serveru
   (d) Dokončení poslání dat přístupovému serveru
   (e) Data z požadavku na přístupový server přijata
   (f) Data z přístupového serveru zpracována
   (g) Přijatá vyčtená autentizační NFC data z modulu čtečky
   (h) Parsování autentizačních NFC dat ze zprávy modulu čtečky
   (i) Zahájení poslání autentizačních NFC dat přístupovému server
   (j) Dokončení poslání dat přístupovému serveru
   (k) Data z požadavku na přístupový server přijata
   (l) Data z přístupového serveru zpracována

Tyto časové značky zachycují všechny důležité operace při ověřování uživatele a také odhalují místa, která by mohla být pomalá, a tím určují prostor pro optimalizaci. Pro modul základny jsou značky a až f a g až l totožné. Pouze se indikuje, v jaké iteraci se kroky provádějí. V první iteraci se posílají pouze UID karty a v druhém kroku data vyčtená ze zašifrovaného bloku karty.

5.3.1 Způsob sběru dat a jejich zpracování

Data s časovými údaji pro optimalizaci systému byla získána z testovací jednotky, která byla připojená k domácí lokální síti, kde běžel lokální přístupový server, který ale přes VPN připojení do univerzitní sítě posílal požadavky na diverzifikaci klíče univerzitnímu přístupovému serveru. Aplikace pro diverzifikaci klíče běžela na vzdáleném serveru, jenž nebyl ve stejné síti LAN jako přístupová jednotka, proto data nereprezentují odezvu systému zcela přesně. Pro hledání možných optimalizací toto není zásadní problém. Časové značky byly vyčteny z logů posílaných do lokálního počítače přes rozhraní UART, kde byly zapisovány časové body. Dále je zachycen čas jednotlivých bodů relativně k předchozímu bodu a k celkovému času jednoho testu. Jeden celý test obsahuje jedno kompletní ověření uživatele.

Zachycená textová data byla zpracována pomocí nástroje Power BI, kde byla data zpracována a poté i přehledně vizualizována.

Nástroj Power BI je aplikace firmy Microsoft. Je to nástroj pro Business Inteligence umožňující interaktivní vizualizaci dat. Obsahuje také nástroje pro přípravu dat a data mining. [96] Pomocí tohoto nástroje vznikly jednotlivé časové přehledy pro jednotlivé konfigurace s navrženými variantami optimalizací a poté bylo provedeno srovnání různých změn a optimalizací.

Při testování bylo provedeno vždy nejméně 25 měření času odezvy a odlehlá pozorování byla odstraněna.

5.3.2 Úvodní porovnání odezvy systému pro různá nastavení

Pro základní srovnání byly použity tyto varianty nastavení systému:

1. Protokol HTTP se zapnutým Nagleovým algoritmem (původní nastavení)

2. Protokol HTTPS se zapnutým Nagleovým algoritmem

3. Protokol HTTP s vypnutým Nagleovým algoritmem

4. Implementace protokolu WebSocket bez šifrování (WS) s vypnutým Nagleovým algoritmem

5. Implementace protokolu WebSocket se šifrováním (WSS) s vypnutým Nagleovým algoritmem

Při analýze naměřených dat bylo zjištěno, že vypnutí Nagleova algoritmu mělo příznivý efekt na odezvu přístupového systému v podobě zrychlení odezvy ověření uživatele v průměru o 200 ms. Z tohoto důvodu byl Nagleův algoritmus deaktivován. Pro zajištění deaktivace bylo nutné upravit knihovnu frameworku ESP-IDF.

Dále bylo při testování zjištěno, že nejdéle trvaly požadavky na ověření uživatele při komunikaci s přístupovým serverem. S nově použitým protokolem HTTPS trval průměrně požadavek na ověření uživatele okolo 3 sekund. Toto je oproti původní nešifrované variantě HTTP nárůst o 60 % průměrné odezvy přístupového systému. Srovnání průměru a mediánu času operací systému se nachází v tabulce 5.1.

Dále bylo zjištěno, že protokol WebSocket bez šifrování byl lehce pomalejší než původní požadavky protokolu HTTP, avšak toto zpomalení je prakticky neznatelné. Toto je pravděpodobně způsobeno použitím mezikroku, kdy přístupový server pro protokol WebSocket předává požadavek webovému PHP serveru. Použití šifrovaného protokolu WebSocket (WSS) zvedlo průměrnou odezvu o 15 % a to průměrně na 2 sekundy.

Oproti protokolu HTTPS je tedy průměrná odezva o 1 sekundu nižší. To je způsobeno tím, že u šifrovaného protokolu WebSocket není nutné inicializovat šifrované TLS spojení, které se musí opakovaně navazovat, což trvá poměrně dlouho, zatímco spojení u protokolu WebSocket je udržováno stále. Tato analýza naměřených dat ukázala, že protokol WebSocket v šifrované podobě je velmi přínosný a má smysl se dále zabývat jeho optimalizací s dalším podrobným testováním.

Typ protokolu	Průměrná doba odezvy	Medián odezvy
HTTPS s Nagleovým algoritmem	2,983 s	2,211 s
HTTP s Nagleovým algoritmem	1,832 s	1,571 s
HTTP bez Nagleova algoritmu	1,641 s	1,094 s
WS bez Nagleova algoritmu	1,678 s	1,157 s
WSS bez Naglesova algoritmu	1,943 s	1,532 s

Tabulka 5.1: Porovnání variant protokolů

Další testování obsahuje tyto možné optimalizace po implementaci protokolu WSS:

1. Implementace protokolu WSS bez čtení reálného UID

2. Implementace protokolu WSS bez čtení reálného UID a s optimalizacemi kódu

3. Implementace protokolu WSS s dalšími optimalizacemi na straně přístupového serveru a nastavení optimálních nastavení kompilace, zlepšení časů blokací a priorit úloh

Při testování byla dále zjištěna časová náročnost na komunikaci modulu čtečky se smart kartou MIFARE DESfire. Proto je v rámci optimalizace vypuštěna jedna operace - čtení reálného UID přiložené karty. Tato změna nemá zásadní vliv na bezpečnost, protože čtení ze souboru na kartě je také šifrované a má vyšší úroveň zabezpečení než čtení reálného klíče, pro jehož přečtení stačí jakýkoliv platný klíč karty.

V další variantě je optimalizováno čekání na blokující operace (fronty, UART, ...). Zásadní úpravou je změna nastavení priorit aplikace, kdy jsou zcela upřednostněny úlohy důležité pro ověření uživatele jakožto hlavní operace celého přístupového systému. Byly odstraněny dlouhé intervaly blokujících čekání a navýšena rychlost komunikace s modulem PN532 přes rozhraní SPI. Dále byla zvýšena rychlost komunikace přes rozhraní UART mezi moduly základny a čtečky.

Poslední varianta změn spočívá v následujících optimalizacích. Na straně přístupového serveru tkví zdokonalení v tom, že server pro protokol WebSocket v prvním dotazu posílá požadavek na diverzifikovaný klíč přímo serveru pro diverzifikovaný klíč. Je tím ušetřena jedna operace, kterou by musel provést webový PHP server. Při kompilaci byly zapnuty optimalizace typu „-O2“. Dále byly vypnuty kontroly přetečení zásobníku, základní kontroly paměti na haldě (heap) a vypnuty logy kromě úrovně odpovídající chybovému stavu. Byl také aktivován release režim pro kompilaci. Srovnání průměru a mediánu času odezvy se nachází v tabulce 5.2.

Varianta protokolu	Průměrná doba odezvy	Medián odezvy
1. WSS	1,943 s	1,532 s
2. WSS bez čtení reálného UID	1,739 s	1,771 s
3. WSS s prvními optimalizacemi	0,914 s	0,908 s
4. WSS další optimalizace	0,527 s	0,511 s

Tabulka 5.2: Porovnání optimalizací WSS

Jak je zřejmé z tabulky 5.2, vynecháním čtení reálného UID smart karty se podařilo zrychlit proces ověření uživatele průměrně přibližně o 200 ms. První část optimalizací při použití 3. varianty dále zrychlila odezvu systému v průměru přibližně o 800 ms. Následné optimalizace dokázaly snížit odezvu přibližně o dalších 400 ms. Po použití všech aplikovaných optimalizací byla průměrná odezva přístupového systému při ověření uživatele do 530 ms.

5.4 Doby trvání jednotlivých kroků v nejrychlejší variantě

Tabulka 5.3 ukazuje, že nejvíce časově náročnými operacemi přístupového systému jsou komunikace se serverem a provádění operací na přístupovému serveru. Při testování bylo zjištěno že, výpočet diverzifikace klíče má dobu vykonávání od 52 do 64 ms a dobu vykonání skriptu PHP na přístupovém serveru pro ověření uživatele od 130 do 145 ms. Bylo naměřeno, že doba režie protokolu WebSocket na jeden požadavek se pohybuje okolo 45 ms ($185 - 140 = 45$ ms). Je velmi pravděpodobné, že režie protokolu WeboSocket a síťového zpoždění nepůjde zcela jednoduše vyřešit.

Nakonec zbývá ještě prostor k optimalizacím v oblasti ověřování uživatele na přístupovém serveru. Existuje možnost vylepšit proces posílání zpráv přes rozhraní UART tak, aby nepoužíval protokol stop-and-wait, nebo alespoň neblokoval provádění zbytku dané úlohy, pokud to není zapotřebí. Tyto optimalizace nejsou však triviální a znamenaly by podstatný zásah do stávajícího systému.

Krok	Průměrná doba odezvy	Medián odezvy
Ověření uživatele (websocket požadavek na server)	185 ms	169 ms
Požadavek na diverzifikovaný klíč	104 ms	104 ms
Autentizace vůči kartě MIFARE DESfire pomocí AES	82 ms	81 ms
Přečtení zašifrovaného souboru	37 ms	36 ms
Poslání dat z modulu čtečky (2x)	26 ms	26 ms
Ostatní operace	93 ms	95 ms
Celkem	527 ms	511 ms

Tabulka 5.3: Doba trvání jednotlivých kroků v nejrychlejší variantě

5.5 Výsledek optimalizací

Po implementaci všech navržených optimalizací a výměně původně navrženého komunikačního protokolu HTTPS za šifrovaný protokol WebSocket došlo k zrychlení odezvy systému průměrně o 2,5 sekundy na hodnotu přibližně 0,5 sekundy. Což je skoro 80 % zrychlení.

Při závěrečném testování byl generován požadavek na výpočet diverzifikovaného klíče vzdáleně přes VPN připojení do univerzitní sítě. V reálném provozu je přístupový systém připojen k lokální síti LAN, kde výpočet diverzifikovaného klíče trvá pouze 11 ms. To je výsledek o 50 ms rychlejší než při provedeném vzdáleném testování přes připojení pomocí VPN. Tento rozdíl bude mít reálně příznivý vliv na nasazený systém a také zlepší odezvu systému.

Takto optimalizovaný přístupový systém je pocitově srovnatelně rychlý jako původní verze, kde nebylo možné použít zabezpečené čtení dat z přístupových karet a bylo načítáno pouze UID karet.

Kapitola 6

Závěr

6.1 Možnosti pro další vývoj

Zajímavou další možností k rozšíření systému je možnost ověření uživatele pomocí mobilního telefonu s využitím technologie NFC HCE. Implementace se smart bezkontaktními kartami a tvorba protokolu APDU je již kompletně hotová, pro využití mobilního telefonu s podporou NFC tedy schází pouze vytvoření specifických příkazů a odpovědí APDU dle standardu ISO 7816-4. Dále by bylo nutné vytvořit aplikaci na straně chytrého telefonu. Tak by šlo učinit však pouze pro operační systém Android, protože operační systém IOS nemá veřejné HCE NFC API. Na straně mobilního telefonu je nutné navrhnout bezpečné předání a uložení klíče a případně i zabezpečený proces komunikace. Toto například pro operační systém Android umožňuje systém KeyStore, který obsahuje i hardwarové zabezpečení klíčů. Mimo mainstreamové mobilní operační systémy se také nabízí možnost použít například operační systém Tizen, jenž by například umožnil ověření pomocí chytrých hodinek.

Další možností vývoje stávajícího systému je přesunutí aplikace pro diverzifikaci klíče na samostatné zabezpečené zařízení tak, aby klíč nebylo možné jakýmkoliv způsobem získat a byl by tedy bezpečně uložen. Takováto zařízení jsou běžně k prodeji, případně je možné navrhnout vlastní.

Eventualitou pro další vylepšení systému je více využít komunikační protokol WebSocket pro načítání nového obrázku k zobrazení na E-INK displeji. Případně lze přesunout proces zpracovávající protokol SIP na přístupový server a tím zmenšit zátěž modulu základny, což by umožnilo rychlejší vývojové změny okolo zpracovávání komunikace pomocí protokolu SIP.

V blízké budoucnosti bude také nutné odladit software pro další avizovanou aktualizaci vývojového prostředí ESP-IDF ve verzi 4.2, kde je komponenta TCPIP adapter označena za zastaralou a bude nahrazena novou obecnější vrstvou nazvanou „esp-netif“, která bude zajišťovat práci se síťovými rozhraními. Bude tedy nutné provést úpravy tak, aby byla zachována možnost přímo získat informace, které jsou nutné pro konfiguraci po startu modulů, z DHCP.

Nabízí se také možnost sjednotit programovací jazyky na straně přístupového serveru. V současnosti se používá Javascript pro protokol WebSocket a PHP pro ostatní aplikace. Bylo by přínosné

sjednotit všechny kódy do jednoho z těchto jazyků. Pro protokol WebSocket v PHP existuje knihovna Ratchet. U PHP je nutné zajistit rychlost vykonávání skriptů, protože programy napsané pro NodeJS a Javascript mohou být rychlejší.

6.2 Výsledek práce

V úvodu této práce byl představen teoretický základ pro provedení práce. V dalších částech byla řešena problematika požadovaných rozšíření přístupového systému o nové vlastnosti. Byl navržen vylepšený nový proces vzdálených OTA aktualizací pro moduly přístupového systému a následně došlo k úpravě části používaného zdrojového kódu pro moduly přístupového systému tak, aby tento nový proces byl implementován. Tento proces byl také otestován a úspěšně implementován do testovacího provozu na instalovaných přístupových jednotkách.

Hlavním tématem této práce byl návrh a implementace nového bezpečného způsobu ověření uživatelů v přístupovém systému pomocí bezkontaktních karet typu MIFARE DESfire.

Implementována a otestována byla šifrovaná komunikace s kartami MIFARE DESfire. Navržený a otestovaný nový systém umožňuje vyčítat data ze zabezpečených souborů na bezkontaktních kartách pomocí diverzifikovaného klíče. Diverzifikace klíče probíhá na přístupovém serveru a je zajištěno, aby byl klíč vždy zabezpečeně přenášen při komunikaci jednotlivých modulů přístupového systému a přístupového serveru. Byla vyvinuta nová verze SW pro moduly přístupového systému a upraven SW na přístupovém serveru. Kryptografické operace probíhají na modulu čtečky a není tedy třeba SAM modulu.

Implementace nového SW byla rozšířena o použití komunikačního protokolu WebSocket pro zajištění adekvátní rychlosti odezvy. Byla analyzována problematická místa, kde vznikalo zpoždění při zpracování požadavků, a následně byly provedeny důsledné optimalizace na základě měření rychlosti. Výsledná implementace nového SW je srovnatelně rychlá s předchozí verzí přístupového systému a je tedy možné ji nasadit bez degradace uživatelského komfortu. Nově vyvinutý SW bude testován v průběhu roku 2021 a vizí je ostré nasazení na konci roku 2021.

Nově navržené využití komunikačního protokolu WebSocket také umožňuje ověřování uživatelů přes jiné způsoby než již uvedené díky tomu, že je zde možná obousměrná komunikace. V rámci úvah by například bylo možné ověřovat uživatele pomocí QR kódu a přihlášení přes webový prohlížeč v mobilním telefonu.

Výsledná práce umožňuje využít mnohem vyšší stupeň zabezpečení a jistotu ověření než předchozí verze, kdy bylo možné karty klonovat a nebylo použito žádné šifrování při komunikaci. Nový způsob ověření uživatele pomocí karet MIREFARE DESfire je nyní vysoce bezpečný, protože není aktuálně znám žádný útok na tyto karty. Ověřování probíhá šifrovaně a data z karet nelze bez znalosti hlavního klíče, případně diverzifikovaného klíče pro danou kartu klonovat.

Tato práce mi přinesla zkušenosti s kryptografií, zabezpečením dat a autentizací a optimalizacemi již fungujícího systému. Při práci jsem musel využít práci s mnohými standardy k smart

kartám a k jejich zabezpečení. Podrobně jsem seznámil s kartami typu MIREFARE DESfire a jejich principy, které jsou zobecnitelné. Dále jsem se seznámil s měřením rychlosti odezvy na vestavěných systémech a porovnáváním různých optimalizací ve vestavěných systémech. Byla nutná také opatrnost při implementaci nových funkcí, aby nebyly narušeny žádné funkce a chod stávajícího systému.

Literatura

1. ESP32 Series: Datasheet. In: [online]. [B.r.], s. 65 [cit. 2021-04-09]. Dostupné z: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf.
2. *ESP-IDF FreeRTOS SMP Changes* [online] [cit. 2021-04-09]. Dostupné z: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/freertos-smp.html.
3. *Smart Card Technology FAQ* [online] [cit. 2021-04-09]. Dostupné z: https://www.securetechalliance.org/smart-cards-faq.
4. SMITH, Nick. *Smart Card Standards: What Do They All Mean?* [Online] [cit. 2021-04-09]. Dostupné z: https://www.microcosm.com/blog/smart-card-standards-explained.
5. *ISO/IEC 7810:2019: Identification cards — Physical characteristics*. 2019. vyd. ISO/IEC, 2019-01.
6. *ISO/IEC 7816:2019: Identification cards — Integrated circuit cards*. 2019. vyd. ISO/IEC, 2019-01.
7. *ISO/IEC 14443: Contactless integrated circuit cards – Proximity cards*. 2018-07. ISO/IEC, 2018. Dostupné také z: https://www.iso.org.
8. *Cards and security devices for personal identification: Contactless proximity objects — Part 2: Radio frequency power and signal interface*. 2020. vyd. Https://www.iso.org: ISO/IEC, 2020.
9. *ABOUT THE TECHNOLOGY: NFC Technology* [online]. 401 Edgewater Place, Suite 600Wakefield, MA 01880, USA: nfc-forum.org, 2021 [cit. 2021-04-09]. Dostupné z: https://nfc-forum.org/what-is-nfc/about-the-technology/.
10. *ISO/IEC 18092: Telecommunications and information exchange between systems — Near Field Communication — Interface and Protocol (NFCIP-1)*. 2013-03. Https://www.iso.org: ISO/IEC, 2013.
11. HUBERS, Erik. *English: NFC Protocol stack* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2021-04-17]. Dostupné z: https://commons.wikimedia.org/wiki/File:NFC_Protocol_Stack.png.
12. *35.240.15: IDENTIFICATION CARDS. CHIP CARDS. BIOMETRICS* [online] [cit. 2021-04-09]. Dostupné z: https://www.iso.org/ics/35.240.15/x/.

13. NFC Multi Protocol for Interoperability Version 2.0 [online]. [B.r.], s. 13 [cit. 2021-04-09]. Dostupné z: https://www.gsma.com/newsroom/wp-content/uploads/SGP.12-v2.0.pdf.

14. *Cards and security devices for personal identification — Contactless proximity objects: Part 1: Physical characteristics*. 2018. vyd. Https://www.iso.org: ISO/IEC, 2018.

15. *Identification cards — Contactless integrated circuit(s) cards — Proximity cards: Part 3: Initialization and anticollision*. :2016. Https://www.iso.org/standard/70171.html: ISO/IEC, 1999-06-11.

16. *ISO14443A: ISO/IEC 14443 Type A* [online]. -: Http://nfc-tools.org/, 6.12.2018 [cit. 2021-04-09]. Dostupné z: http://nfc-tools.org/index.php/ISO14443A.

17. *ISO/IEC 14443-4: Cards and security devices for personal identification — Contactless proximity objects Transmission protocol*. 2018-07. Https://www.iso.org: ISO/IEC, 2018.

18. *Identification cards — Integrated circuit cards — Part 4: Organization, security and commands for interchange*. 2013. vyd. Https://www.iso.org/: ISO/IEC, 2013. Dostupné také z: https://www.iso.org/standard/54550.html.

19. UM0701-02: PN532 User Manual [online]. [B.r.], s. 200 [cit. 2021-04-09]. Dostupné z: https://www.nxp.com/docs/en/user-guide/141520.pdf.

20. PN532/C1: Near Field Communication (NFC) controller [online]. [B.r.], č. 115436, s. 222 [cit. 2021-04-09]. Dostupné z: https://www.nxp.com/docs/en/nxp/data-sheets/PN532_C1.pdf.

21. *What is AES encryption and how does it work?* [Online]. Suite 3 Falcon Court Business Centre, College Road, Maidstone, Kent, ME15 6TF, United Kingdom [cit. 2021-04-09]. Dostupné z: https://www.comparitech.com/blog/information-security/what-is-aes-encryption/.

22. *Advanced Encryption Standard (AES) (FIPS PUB 197)*. 197. vyd. Department of Commerce, National Institute of Standards a Technology, Information Technology Laboratory (ITL).: National Institute of Standards a Technology (NIST), November 26, 2001.

23. K čemu je dobré WPA3? [Online]. [B.r.] [cit. 2021-04-09]. Dostupné z: https://channelworld.cz/software/k-cemu-je-dobre-wpa3-21474.

24. *BSI – Technical Guideline: Cryptographic Mechanisms: Recommendations and Key Lengths*. 2021-01. Bonn, Germany: Federal Office for Information Security P.O.B. 20 03 63, 53133 Bonn, Germany, March 24, 2021.

25. SMITH, Kaleigh. RIJNDAEL: Advanced Encryption Standard. In: [online]. [B.r.] [cit. 2021-04-09]. Dostupné z: https://www.cs.mcgill.ca/~kaleigh/computers/crypto_rijndael.html.

26. *Information technology — Security techniques: Modes of operation for an n-bit block cipher*. 3. vyd. Https://www.iso.org: ISO/IEC, 2006-02.

27. DWORKIN, Morris. *Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication*. NIST Special Publication 800-38B. Attn: Computer Security Division, Information Technology Laboratory 100 Bureau Drive (Mail Stop 8930) Gaithersburg, MD 20899-8930: National Institute of Standards a Technology, 2005.

28. IWATA, Tetsu. *OMAC: One-Key CBC MAC* [online] [cit. 2021-04-09]. Dostupné z: `https://www.nuee.nagoya-u.ac.jp/labs/tiwata/omac/omac.html`.

29. *message authentication code (MAC)* [online]. https://searchsecurity.techtarget.com/, November 2010 [cit. 2021-04-09]. Dostupné z: `https://searchsecurity.techtarget.com/definition/message-authentication-code-MAC`.

30. TWISP. *Message authentication code, based on MAC.gif* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2021-04-17]. Dostupné z: `https://commons.wikimedia.org/wiki/File:MAC.svg`.

31. IWATA, Tetsu; KUROSAWA, Kaoru. OMAC: One-Key CBC MAC. *Lecture Notes in Computer Science* [online]. [B.r.], roč. 2887, s. 25 [cit. 2021-04-09]. ISSN 978-3-540-39887-5. Dostupné z: `https://doi.org/10.1007/978-3-540-39887-5_11`.

32. FARISSI, Ilhame El; AZIZI, Mostafa; MOUSSAOUI, Mimoun. Classification of smartcard attacks. *2011 International Conference on Multimedia Computing and Systems* [online]. 2011, s. 1–5 [cit. 2021-04-09]. ISBN 978-1-61284-730-6. Dostupné z DOI: `10.1109/ICMCS.2011.5945583`.

33. TUNSTALL, Michael. Smart Card Security. *Smart Cards, Tokens, Security and Applications* [online]. 2008, s. 195–228 [cit. 2021-04-09]. ISBN 978-0-387-72197-2. Dostupné z DOI: `10.1007/978-0-387-72198-9_9`.

34. KUMAR, Sandeep. Classification and detection of computer intrusions [online]. [B.r.], č. UMI Order No. GAX96-01522, s. 180 [cit. 2021-04-09]. Dostupné z DOI: `10.5555/240069`.

35. JOANCOMARTÍ, Jordi Herrera. 5211 - SECURITY IN RFID DEVICES [online]. [B.r.], s. 71 [cit. 2021-04-09]. Dostupné z: `https://core.ac.uk/download/pdf/78518569.pdf`.

36. GARCIA, Flavio D.; KONING GANS, Gerhard de; MUIJRERS, Ruben; ROSSUM, Peter van; VERDULT, Roel; SCHREUR, Ronny Wichers; JACOBS, Bart. Dismantling MIFARE Classic. *Institute for Computing and Information Sciences, Radboud University Nijmegen, The Netherlands* [online]. [B.r.], s. 18 [cit. 2021-04-09]. Dostupné z: `https://www.cs.bham.ac.uk/~garciaf/publications/Dismantling.Mifare.pdf`.

37. KONING GANS, Gerhard de. *Outsmarting Smart Cards*. 2013-05. 2013. ISBN 978-94-6191-675-4.

38. MAYES, Keith E.; CID, Carlos. The MIFARE Classic story. *Information Security Technical Report* [online]. 2010, roč. 15, č. 1, s. 8–12 [cit. 2021-04-09]. ISSN 13634127. Dostupné z DOI: 10.1016/j.istr.2010.10.009.

39. OSWALD, David. *IMPLEMENTATION ATTACKS: FROM THEORY TO PRACTICE*. Bochum, September 2013, 2013. Disertace. Fakultat f ur Elektrotechnik und Informationstechnik ¨ an der Ruhr-Universitat Bochum.

40. OSWALD, David; PAAR, Christof. Breaking Mifare DESFire MF3ICD40: Power Analysis and Templates in the Real World. In: [online]. Horst Görtz Institute for IT Security Ruhr-University Bochum, Germany, [b.r.], s. 16 [cit. 2021-04-09]. Dostupné z: https://www.iacr.org/archive/ches2011/69170208/69170208.pdf.

41. BHARGAVAN, Karthikeyan; LEURENT, Gaëtan. *Sweet32: Birthday attacks on 64-bit block ciphers in TLS and OpenVPN: CVE-2016-2183, CVE-2016-6329* [online] [cit. 2021-04-09]. Dostupné z: https://sweet32.info/.

42. *Template Attacks* [online] [cit. 2021-04-09]. Dostupné z: https://wiki.newae.com/Template_Attacks.

43. FLYNN, Rory. *An investigation of possible attacks on the MIFARE DESFire EV1 smartcard used in public transportation*. Trinity Collega, July 24, 2019. Bakalářská.

44. AND, Markus Kuhn. Physical Security of Smartcards: Security Research. *Information Security Technical Report,* 1999, č. Vol 4, s. 14.

45. RANKL, Wolfgang. Overview about attacks on smart cards. In: *Information Security Technical Report* [online]. 2003, sv. 8, s. 67–84 [cit. 2021-04-09]. Č. 1. ISSN 13634127. Dostupné z DOI: 10.1016/S1363-4127(03)00107-9.

46. TEAM, NXP MIFARE. NXP MIFARE Webinar: Innovation Road Map: Present Improved- Future Inside: Overview of NXP MIFARE product portfolio. Product families include: MIFARE Classic, MIFARE Plus, MIFARE DESFire and MIFARE Ultralight. Details the evolution and innovation of the various product families are given. In: [online]. [B.r.], s. 63 [cit. 2021-04-09]. Dostupné z: https://www.slideshare.net/NXPMIFARETeam/mifare-webinar-innovation-road-map-present-improved-future-inside-1.

47. *MF1S70YYX$_V$1: MIFARE Classic EV1 4K - Mainstream contactless smart card IC for fast and easy solution development*. Rev. 3f.2. NXP, 23 November 2017.

48. *NXP® MIFARE Plus®: Seamless deployment of security upgrades for contactless services* [online] [cit. 2021-04-09]. Dostupné z: https://www.nxp.com/docs/en/fact-sheet/MFPLUSEV2LF.pdf.

49. *MF0ICU2: MIFARE Ultralight C - Contactless ticket IC*. Rev. 3.3. NXP, 30 July 2019.

50. *MF3ICDx21$_4$1$_8$1: MIFARE DESFire EV1 contactless multi-application IC*. Rev. 3.2. NXP, 9 December 2015.

51. *Enhanced user experience through active application management* [online] [cit. 2021-04-09]. Dostupné z: https://www.nxp.com/video/enhanced-user-experience-through-active-application-management:MIFARE-USER-EXPERIENCE-WEBINAR.

52. *MIFARE DESFire® EV1* [online] [cit. 2021-04-09]. Dostupné z: https://developer.fidesmo.com/documentation/desfire-implementation.

53. *NXP® MIFARE® DESFire® EV2* [online] [cit. 2021-04-09]. Dostupné z: https://www.nxp.com/docs/en/fact-sheet/MIFARE-DESFIRE-EV2-FS.pdf.

54. *MF3D(H)x3: MIFARE DESFire EV3 contactless multi-application IC*. Rev. 3.0. NXP, 15 May 2020.

55. TEAM, NXP MIFARE. *Transaction MAC Feature* [online] [cit. 2021-04-09]. Dostupné z: https://www.slideshare.net/NXPMIFARETeam/transaction-mac-feature.

56. *Information technology — Security techniques — Evaluation criteria for IT security: Introduction and general model*. 2009. vyd. ISO/IEC, 2014-01.

57. SHIMONSKI, Robert J. *Microsoft Windows and the Common Criteria Certification Part I* [online] [cit. 2021-04-09]. Dostupné z: https://techgenix.com/Windows-Common-Criteria-Certification-Part-I/.

58. *Common Criteria Assurance Levels* [online] [cit. 2021-04-09]. Dostupné z: https://web.archive.org/web/20041012181256/http://www.cesg.gov.uk/site/iacs/index.cfm?menuSelected=1%5C&displayPage=13.

59. *MIFARE DESFire EV2 – Security Target Lite*. Rev. 1.5. NXP, 2016-04-29.

60. DIFFIE Whitfield; Hellman, Martin E. Exhaustive Cryptanalysis of the NBS Data Encryption Standard. *Computer Science* [online]. [B.r.], roč. 2412454, s. 74–84 [cit. 2021-04-09]. ISSN 0018-9162. Dostupné z DOI: 10.1109/C-M.1977.217750.

61. *Updates* [online]. 100 Bureau DriveGaithersburg, MD 20899: NIST, July 11, 2017 [cit. 2021-04-09]. Dostupné z: https://csrc.nist.gov/News/2017/Update-to-Current-Use-and-Deprecation-of-TDEA.

62. *The anti-tearing: concept and mechanisms* [online] [cit. 2021-04-09]. Dostupné z: https://www.springcard.com/en/blog/news/the-anti-tearing-concept-and-mechanisms.

63. *AN10969: System level security measures for MIFARE installations*. Rev. 2.1. NXP, 22 April 2020.

64. *AN10922: Symmetric key diversifications*. Rev. 2.2. Https://www.nxp.com/docs/en/application-note/AN10922.pdf: NXP, 2 July 2019.

65. *AN12695: MIFARE SAM AV3 - Quick start up guide*. Rev. 1.4. 15 June 2020. Dostupné také z: Https://www.nxp.com/docs/en/application-note/AN12695.pdf.

66. *MF4SAM3HN/9BA6AUZ: MIFARE SAM AV3 Secure Access Module Upto 32kB EEPROM 1kB RAM 32-Pin HVQFN Surface Mount* [online] [cit. 2021-04-09]. Dostupné z: https://www.avnet.com/shop/emea/products/nxp/mf4sam3hn-9ba6auz-3074457345642286020/%5C%20.

67. *Mifare SAM AV3: 19.80 €* [online] [cit. 2021-04-09]. Dostupné z: https://www.cardomatic.de/mifare-SAM-AV3/en.

68. *MF4SAM3HN/9BA6AUZ* [online] [cit. 2021-04-09]. Dostupné z: https://cz.mouser.com/ProductDetail/NXP-Semiconductors/MF4SAM3HN-9BA6AUZ?qs=%5C%2Fha2pyFaduiUpHNOaA6ipuJ5CBQLZmT4hJeezb8GzBkAt1GGoUxKDxyBdFG3KK3S.

69. *Over The Air Updates (OTA)* [online] [cit. 2021-04-09]. Dostupné z: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/ota.html%5C#ota-process-overview.

70. *Partition Tables* [online] [cit. 2021-04-09]. Dostupné z: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/partition-tables.html.

71. *Bootloader* [online] [cit. 2021-04-09]. Dostupné z: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/bootloader.html.

72. Flash Encryption: API Guides. In: [online]. [B.r.] [cit. 2021-04-09]. Dostupné z: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/security/flash-encryption.html.

73. VOAS, J. Fault injection for the masses. *Computer*. [B.r.], roč. 30, č. 12, s. 129–130. ISSN 00189162. Dostupné z DOI: 10.1109/2.642820.

74. ESP32 Fault Injection Vulnerability: Impact Analysis [online]. [B.r.] [cit. 2021-04-09]. Dostupné z: https://www.espressif.com/en/news/ESP32_FIA_Analysis.

75. Espressif ESP32: Bypassing Encrypted Secure Boot: (CVE-2020-13629). In: [online]. [B.r.] [cit. 2021-04-09]. Dostupné z: https://raelize.com/blog/espressif-esp32-bypassing-encrypted-secure-boot-cve-2020-13629/.

76. Security Advisory concerning fault injection and eFuse protections: CVE-2019-17391 [online]. [B.r.] [cit. 2021-04-09]. Dostupné z: https://www.espressif.com/en/news/Security_Advisory_Concerning_Fault_Injection_and_eFuse_Protections%5C%20.

77. Espressif ESP32: Bypassing Flash Encryption: (CVE-2020-15048). In: [online]. [B.r.] [cit. 2021-04-09]. Dostupné z: https://raelize.com/blog/espressif-systems-esp32-bypassing-flash-encryption/.

78. *ESP32: Partition Table Update through OTA* [online] [cit. 2021-04-09]. Dostupné z: https://esp32.com/viewtopic.php?t=11482.

79. *How should one handle expiring TLS certificates?* [Online] [cit. 2021-04-09]. Dostupné z: https://esp32.com/viewtopic.php?t=7585%5C#p31952.

80. *The device can't connect to the mqtt server after the certificate has been changed on the server* [online] [cit. 2021-04-09]. Dostupné z: https://esp32.com/viewtopic.php?t=18276.

81. *Public contract: Bezkontaktní čipové čtečky.* 12.04.2019. vyd. [B.r.]. Č. P19V00000157. Dostupné také z: https://zakazky.vsb.cz/contract_display_149.html?lang=en.

82. ELMUE. *DIY electronic RFID Door Lock with Battery Backup* [online] [cit. 2021-04-19]. Dostupné z: https://www.codeproject.com/Articles/1096861/DIY-electronic-RFID-Door-Lock-with-Battery-Backup.

83. DARCONEOUS. *Nfc-tools / libfreefare* [online]. GIGA-TMS INC. [cit. 2021-04-19]. Dostupné z: https://github.com/nfc-tools/libfreefare/blob/master/libfreefare/mifare_desfire.c.

84. SOTECHCLLC. *Mifare® Application Programming Guide for DESFire®: REV E* [online]. GIGA-TMS INC. [cit. 2021-04-19]. Dostupné z: https://scancode.ru/upload/iblock/de4/mifare_application_programming_guide_for_desfire_rev.e.pdf.

85. SOTECHCLLC. *RFDoorLock* [online] [cit. 2021-04-19]. Dostupné z: https://github.com/sotechcllc/RFDoorLock.

86. ELMÜ. *DESFire EV1 Communication Examples* [online] [cit. 2021-04-19]. Dostupné z: https://hack.cert.pl/files/desfire-9f122c71e0057d4f747d2ee295b0f5f6eef8ac32.html.

87. RIDRIX. *Mifare Desfire communication example* [online] [cit. 2021-04-19]. Dostupné z: https://ridrix.wordpress.com/tag/desfire-protocol/.

88. RALPH, Jacobi; WYATT, Josh. *MIFARE DESFire EV1 AES Authentication With TRF7970: Safety and Security (S2) NFC/RFID Applications* [online] [cit. 2021-04-19]. Dostupné z: https://www.ti.com/lit/an/sloa213/sloa213.pdf.

89. SECURITY, Townsend. *AES vs. DES Encryption: Why Advanced Encryption Standard (AES) has replaced DES, 3DES and TDE* [online] [cit. 2021-04-09]. Dostupné z: https://www.precisely.com/blog/data-security/aes-vs-des-encryption-standard-3des-tdea.

90. *TLS handshake speeds: RSA is faster than ECC!?* [Online] [cit. 2021-04-09]. Dostupné z: https://esp32.com/viewtopic.php?f=13%5C&t=929.

91. *Project Configuration* [online] [cit. 2021-04-09]. Dostupné z: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/kconfig.html.

92. DATAR, Kshama. *Best Practices for TCP Optimization in 2019* [online] [cit. 2021-04-09]. Dostupné z: https://www.extrahop.com/company/blog/2016/tcp-nodelay-nagle-quickack-best-practices/.

93. *ESP-IDF Versions* [online] [cit. 2021-04-09]. Dostupné z: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/versions.html.

94. PROJECTGUS. *ESP-IDF Release v4.0* [online] [cit. 2021-04-09]. Dostupné z: https://github.com/espressif/esp-idf/releases/tag/v4.0.

95. GROKHKOTOV, Ivan. Developing and Debugging ESP32 IoT Applications using Espressif Eclipse Plugin. In: [online]. Eclipse Foundation, [b.r.] [cit. 2021-04-09]. Dostupné z: https://www.youtube.com/watch?v=CbPX3q7LeBc.

96. *Power BI* [online] [cit. 2021-04-09]. Dostupné z: https://powerbi.microsoft.com/cs-cz/.

Příloha A

Struktura zdrojových kódů

A.1 Společné zdrojové kódy pro modul základny a čtečky:

- common.h
  - Obsahuje společné konstanty pro komunikaci a adresy jednotlivých funkcionalit
  - Obsahuje definice společných struktur a výčtů
  - Obsahuje hlavičky funkcí pro vytváření zpráv
- processing_uart.h
  - Zajišťuje zpracování zprávy z UART
  - Zajišťuje zpracování chyb a potvrzování z UART
  - Obsahuje buffer pro příchozí data
- sound.h
  - Zajišťuje funkce pro použití piezoměniče
  - Chráněno mutexem, aby nedocházelo k souběhu a hrála pouze jedna znělka v jednu chvíli
  - Obsahuje funkci pro hraní volitelné melodie
- uart.h
  - Zajišťuje posílání zpráv přes UART
  - Obsahuje funkci pro sestavení (parsování) zprávy z příchozích dat
- common_esp_headers.h
  - Obsahuje obvyklé hlavičky v prostředí frameworku ESP-IDF
  - Z důvodu lepší orientace ve změnách a úpravách po aktualizaci na verzi frameworku ESP-IDF 4.01 vloženy hlavičky do jediného souboru

A.2 Zdrojové kódy pro modul základny:

- emergency_light.h
  - Obsahuje úlohu pro doplňkovou funkci výstražných světel při požáru nebo jiné události
- enviroment.h
  - Definuje makra pro definici prostředí (konstanty, URL adresy, . . .)
  - Je definováno prostředí pro testování a pro ostrý provoz
- ethernet_module.c
  - Vstupní bod programu
  - Zajišťuje inicializaci základních funkcí
  - Obsahuje úlohu pro příjem a poslání zpráv modulu čtečky přes rozhraní RS422
- https.h
  - Obsahuje úlohu pro obsluhu komunikace s přístupovým serverem pomocí protokolu HTTPS při ověřování pomocí smart karet
  - Obsahuje úlohu pro obsluhu komunikace s přístupovým serverem pomocí protokolu HTTPS při ověřování pomocí protokolu SIP
  - Obsahuje úlohu pro obsluhu komunikace s E-INK serverem
- websockets.h
  - Obsahuje úlohu pro obsluhu komunikace s přístupovým serverem pomocí protokolu WebSocket při ověřování pomocí smart karet
- ota.h
  - Definuje funkci pro aktualizaci čtečky
  - Definuje funkci pro aktualizaci základny
  - Definuje funkci pro naplnění informace o verzi a názvu projektu modulu čtečky
- initialization.h
  - Zajišťuje inicializaci modulu na základě informací z DHCP serveru ve formátu JSON
  - Parsuje JSON konfiguraci
  - Spouští jednotlivé úlohy na základě konfigurace
  - Spouští aktualizace v případě potřeby

  - Zajišťuje překlad adres zpráv na odpovídající fronty

- perf.h
  - Obsahuje funkce pro vypsání časových razítek při měření rychlosti odezvy systému

- processing_server_response.h
  - Společné funkce pro vyhodnocení odpovědí z přístupového serveru
  - Dekodování a překlad zpráv z binárního formátu přicházejícího z modulu čtečky do formátu JSON, který se zasílá přístupovému serveru
  - Používá se pro protokol WebSocket i protokol HTTP

- sntp_time.h
  - Obsahuje funkce pro získání reálného času pomocí protokolu SNTP a přenos tohoto času do modulu čtečky

- syslog.h
  - Obsahuje definici funkce pro logování pomocí Syslogu

- utils.h
  - Obsahuje inicializaci periferie Ethernet
  - Pomocné funkce
  - Definice některých pinů specifických pro modul základny
  - Funkce pro přečtení dat o lokaci konfigurace získaných z DHCP serveru a spuštění inicializace, která zpracuje tuto informaci

A.3 Zdrojové kódy pro modul čtečky:

- „eink“ komponenta
  - default.h
    - Obsahuje výchozí obrázek pro E-INK displej
  - epd.h
    - Obsahuje ovládání E-INK displeje, zajišťuje komunikaci s E-INK displejem pomocí rozhraní SPI
    - Upraveno z ukázkového kódu výrobce displeje Waveshare pro mikrokontroléry STM
  - epdpaint.h

        * Obsahuje funkce pro vykreslování informací na E-INK displej

- „pn532“ komponenta
  - helpers.h - Pomocné funkce pro práci se smart kartami MIFARE DESfire
  - desfire.h - Funkce pro komunikaci se smart kartami MIFARE DESfire
  - pn532_functions.h - Kódy pro práci s modulem PN532 a základní komunikace se smart kartami
  - pn532_protocol.h - Obsahuje funkce pro výměnu rámců s modulem PN532
  - pn532_spi.h - Nízkoúrovňová část zajištující přenos rámců při komunikaci s modulem PN532 pomocí rozhraní SPI
- lights.h
  - Obsahuje hlavičky funkcí pro ovládání LED diod a zajišťuje indikaci chybového stavu systému
- eink_api.h
  - Obsahuje hlavičky funkcí pro práci s E-INK displejem na vyšší úrovni
  - Obsahuje úlohu obsluhující vykreslování nových obrázků z E-INK serveru
- nfc.h
  - Obsahuje úlohu pro obsluhu karet NFC
- utils.h
  - Obsahuje definici pinů použitých pro komunikaci UART přes rozhraní RS422
- reader_module_main.c
  - Vstupní bod programu
  - Obsahuje úlohu pro příjem zpráv a jejich zpracování
  - Inicializace periferií a úloh
  - Zpracovávání aktualizací
  - Zpracovávání melodie a RGB LED signalizace
  - Úloha pro čtení karet RFID

A.4 Použité cizí zdrojové kódy:

- Modul základny:
  - components/sip/*
    - Komponenta pro protokol SIP napsaná jiným autorem
- Modul čtečky:
  - components/fonts/*
    - Komponenta obsahující fonty od tvůrců „MCD Application Team“
- epdpaint.h - Zdrojový kód pro vykreslování na E-INK displej od autora Yehui pracujícího pro výrobce E-INK displeje společnosti Waveshare [1]

[1] https://github.com/soonuse/epd-library-stm32

Příloha B

Struktura přiložených souborů

Přiložené soubory obsahují skript pro server NodeJS s protokolem WebSocket a dva projekty pro mikrokontrolér ESP32. Dále zdrojový kód pro přístupový server v jazyce PHP. Jeden je pro modul čtečky a druhý pro modul základny. Dále také obsahují patch pro prostředí ESP-IDF a to z důvodu, že v frameworku ESP-IDF upraveno výchozí nastavení.

Popis složek a souborů:

- pristupovy–system – Obsahuje data pro modul základny
  - main – Obsahuje zdrojové kódy pro modul základny
  - server_certs – Certifikáty pro protokol SSL
  - build – Do této složky se kompilují zdrojové kódy projektu
  - components – Složka pro komponenty
- reader_module – Obsahuje data pro modul čtečky
  - main – Obsahuje zdrojové kódy pro modul čtečky
  - build – Do této složky se kompilují zdrojové kódy projektu
  - components – Složka pro komponenty
- patches – Složka obsahující patch pro opravu předávání parametrů pomocí DHCP protokolu a nastavení NO_DELAY pro TCP socket pro protokoly WebSocket a HTTP/HTTPS
- websocket–server – Složka obsahující JS skript pro obsluhu protokolu WebSocket
- asn–server – Složka obsahující PHP skripty pro přístupový server
- analysis – Složka obsahující naměřená data a soubor PowerBI použitý při analýzách doby času prováděných operací

Příloha C

Kompilace v prostředí ESP-IDF v operačním systému Ubuntu a Debian

C.1 Instalace frameworku ESP-IDF

V prvním kroku je nutné si stáhnout následující balíčky a nástroje takto:

C.1.1 Stažení nástrojů

```
sudo apt-get install git wget flex bison gperf python3 python3-pip python3-
    setuptools cmake ninja-build ccache libffi-dev libssl-dev dfu-util libusb
    -1.0-0
```

Výpis C.1: Stažení nástrojů

C.1.2 Stažení frameworku ESP-IDF

Dále je nutné si stáhnout zdrojové kódy pro framework ESP-IDF takto:

```
mkdir -p ~/esp
cd ~/esp
git clone --recursive https://github.com/espressif/esp-idf/tree/v4.0.1
```

Výpis C.2: Stažení frameworku ESP-IDF ve verzi 4.01 z repozitáře

C.1.3 Konfigurace prostředí

Po stažení frameworku ESP-IDF je nutné nakonfigurovat nástroje tohoto frameworku:

```
cd ~/esp/esp-idf
./install.sh
```

Výpis C.3: Konfigurace nástrojů

C.1.4 Nastavení systémových proměnných

Po konfiguraci nástrojů je nutné nastavit proměnné prostředí pomocí příkazu:

```
. $HOME/esp/esp-idf/export.sh
```

Výpis C.4: Export proměnných

C.2 Použití frameworku ESP-IDF

Nyní je již framework ESP-IDF korektně nainstalován a je možné kompilovat, spouštět a konfigurovat projekty. Pro správné chování projektu základny pro přístupový systém je nutné aplikovat dva patche. Pomocí příkazu „menuconfig“ se nastavuje konfigurace projektu a je důležité nastavit port, na kterém je připojený příslušný mikrokontrolér ESP32.

```
cd ~/esp/pristupovy-system
idf.py menuconfig
```

Výpis C.5: Konfigurace projektu

C.2.1 Kompilace projektu

Kompilace projektu je možná pomocí příkazu:

```
idf.py build
```

Výpis C.6: Kompilace projektu

C.2.2 Nahrání projektu

Nahrání binárního souboru do ESP32 se provádí příkazem:

```
idf.py -p PORT [-b BAUD] flash
```

Výpis C.7: Nahrání projektu do mikrokontroléru ESP32

C.2.3 Výpisy logu

Případné výpisy z logu je možné číst pomocí příkazu:

```
idf.py -p /dev/ttyUSB0 monitor
```

Výpis C.8: Čtení logu z mikrokontroléru ESP32

Příloha D

SW pro přístupový server

D.1 Skript v jazyce PHP

Skript pro ověření uživatele v jazyce PHP je možné spustit v prostředí podporující PHP. Například je vhodné využít webový server Apache s modulem „mod_php“.

D.2 Server pro obsluhu protokolu WebSocket

Pro zprovoznění serveru pro obsluhu protokolu WebSocket v prostředí NodeJS je nutné do systému nainstalovat NodeJS a balíčkový manažer NPM.

```
sudo apt update
sudo apt install nodejs
sudo apt install npm
nodejs -v
```

Výpis D.1: Instalace NodeJS a NPM

Pro spuštění serveru je nejprve nutné nainstalovat balíčky pomocí příkazu:

```
npm install
```

Výpis D.2: Instalace balíčků

Skript lze spustit pomocí příkazu:

```
nodejs server.js
```

Výpis D.3: Instalace balíčků

Pro spouštění a udržovaní serveru v běhu je možné využít správce procesů jako je například PM2.

Příloha E

Ukázka výměny dat při autentizaci smart karty MIFARE DESfire

E.1 Autentizace mezi smart kartou a modulem PN532

Modul PN532 po detekci přiložení smart karty MIFARE DESfire přečte UID karty. Z tohoto UID karty poté vygeneruje přístupový server diverzifikovaný klíč. Pomocí tohoto klíče modul čtečky zahájí proces ověření se smart kartou MIFARE DESfire. Průběh autentifikace:

1. Modul čtečky přijme diverzifikovaný klíč

Sekvence	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15
KEY	93	2e	11	74	2c	d3	0c	79	07	ef	4e	95	0a	94	0b	e4

Tabulka E.1: Diverzifikovaný klíč

2. Modul čtečky zašle příkaz smart kartě

Příkaz	Klíč
AA	00

Tabulka E.2: Příkaz k autentizaci smart kartě

3. Smart karta odpoví odpovědí „AF“

Odpověď	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15
AF	13	04	e4	6f	c7	45	18	1d	ff	28	fc	c5	0d	d9	3e	df

Tabulka E.3: První odpověď smart karty na proces ověření

4. Po dešifrování odpovědi smart karty diverzifikovaným klíčem se získá náhodná sekvence R1

Sekvence	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15
IV	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
KEY	df	4e	78	98	47	48	4e	ee	2f	53	54	8a	3d	42	8d	ec
$R1_{encrypted}$	13	04	e4	6f	c7	45	18	1d	ff	28	fc	c5	0d	d9	3e	df
$R1_{decrypted}$	df	4e	78	98	47	48	4e	ee	2f	53	54	8a	3d	42	8d	ec

Tabulka E.4: Dešifrování sekvence R1

5. Provede se bajtová rotace doleva sekvence R1

Sekvence	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15
$R1_{rotated}$	4e	78	98	47	48	4e	ee	2f	53	54	8a	3d	42	8d	ec	df

Tabulka E.5: Rotace sekvence R1

6. Vytvoření náhodné sekvence R2

Sekvence	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15
$R2$	41	d9	de	83	20	1b	94	b3	0b	a7	6b	59	91	5a	4c	3f

Tabulka E.6: Náhodná sekvence R2

7. Spojení se sekvence R2 a otočení sekvence R1 ($R2 + R1_{rotated}$)

Sekvence	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15
$R2 + R1_{rotated}$	41	d9	de	83	20	1b	94	b3	0b	a7	6b	59	91	5a	4c	3f
	78	98	47	48	4e	ee	2f	53	54	8a	3d	42	8d	ec	4e	00

Tabulka E.7: Spojení sekvencí $R2 + R1_{rotated}$

8. Modul čtečky zašifruje spojenou sekvenci D pomocí diverzifikovaného klíče a jako IV se použije přijatá zašifrovaná sekvence $R1_{encrypted}$

Sekvence	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15
IV	13	04	e4	6f	c7	45	18	1d	ff	28	fc	c5	0d	d9	3e	df
KEY	df	4e	78	98	47	48	4e	ee	2f	53	54	8a	3d	42	8d	ec
$D_{concated}$	41	d9	de	83	20	1b	94	b3	0b	a7	6b	59	91	5a	4c	3f
	78	98	47	48	4e	ee	2f	53	54	8a	3d	42	8d	ec	4e	00
$D_{encrypted}$	55	83	df	31	20	31	00	86	07	8c	0e	20	cb	73	23	8c
	42	52	b7	20	69	d8	59	82	ac	80	41	98	ed	de	93	f1

Tabulka E.8: Zašifrování sekvence $D_{concated}$

9. Zašifrovaná sekvence ($D_{encrypted}$) se zašle smart kartě

Odpověď	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15
AF	55	83	df	31	20	31	00	86	07	8c	0e	20	cb	73	23	8c
	42	52	b7	20	69	d8	59	82	ac	80	41	98	ed	de	93	f1

Tabulka E.9: Zašifrovaný příkaz obsahující sekvencí $D_{encrypted}$

10. Smart karta zašle modulu čtečky odpověď typu „AF“

Odpověď	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15
AF	79	27	54	41	19	62	12	82	82	d7	46	2d	5d	fb	3c	ee

Tabulka E.10: Odpověď smart karty

11. Po dešifrování odpovědi modulem čtečky pomocí diverzifikovaného klíče, se získá posunuté sekvence R2 ($R2_{rotated}$). Jako IV se použije posledních 16 bajtů spojené sekvence D ($D_{encrypted}$)

Sekvence	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15
IV	42	52	b7	20	69	d8	59	82	ac	80	41	98	ed	de	93	f1
KEY	df	4e	78	98	47	48	4e	ee	2f	53	54	8a	3d	42	8d	ec
$R2_{encrypted}$	79	27	54	41	19	62	12	82	82	d7	46	2d	5d	fb	3c	ee
$R2_{rotated}$	3f	41	d9	de	83	20	1b	94	b3	0b	a7	6b	59	91	5a	4c

Tabulka E.11: Dešifrování sekvence $R2_{rotated}$

12. Modul čtečky provede rotaci doprava dešifrované sekvence $R2_{rotated}$

Sekvence	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15
$R2_{rotated}$	3f	41	d9	de	83	20	1b	94	b3	0b	a7	6b	59	91	5a	4c
$R2_{recieved}$	41	d9	de	83	20	1b	94	b3	0b	a7	6b	59	91	5a	4c	3f

Tabulka E.12: Rotace sekvence $R2_{rotated}$

13. Pokud se přijatá a posunutá sekvence($R2_{rotated}$) shoduje s vygenerovanou sekvencí R2, tak proběhla autentizace úspěšně.

$$R2_{recieved} = R2$$

14. Modul čtečky vytvoří session klíč ($KEY_{session}$)

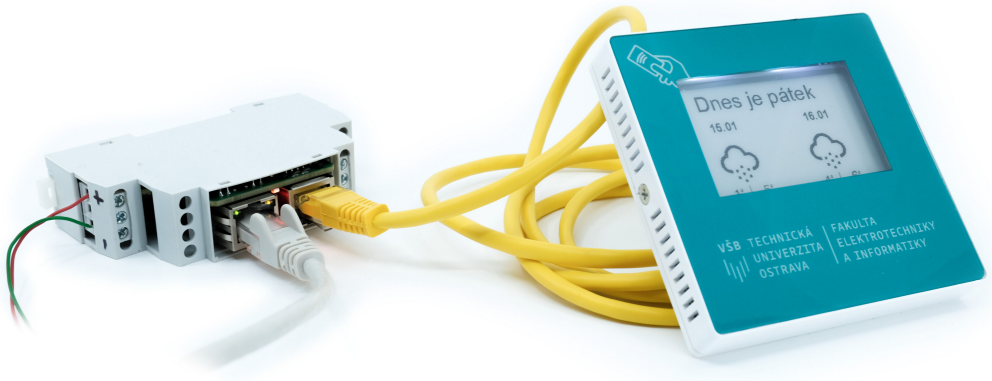
Index	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15
Ze sekvence	R2	R2	R2	R2	R1	R1	R1	R1	R2	R2	R2	R2	R1	R1	R1	R1
Index bajtu	B0	B1	B2	B3	B0	B1	B2	B3	B12	B13	B14	B15	B12	B13	B14	B15
$KEY_{session}$	41	d9	de	83	df	4e	78	98	91	5a	4c	3f	3d	42	8d	ec

Tabulka E.13: Odpověď smart karty

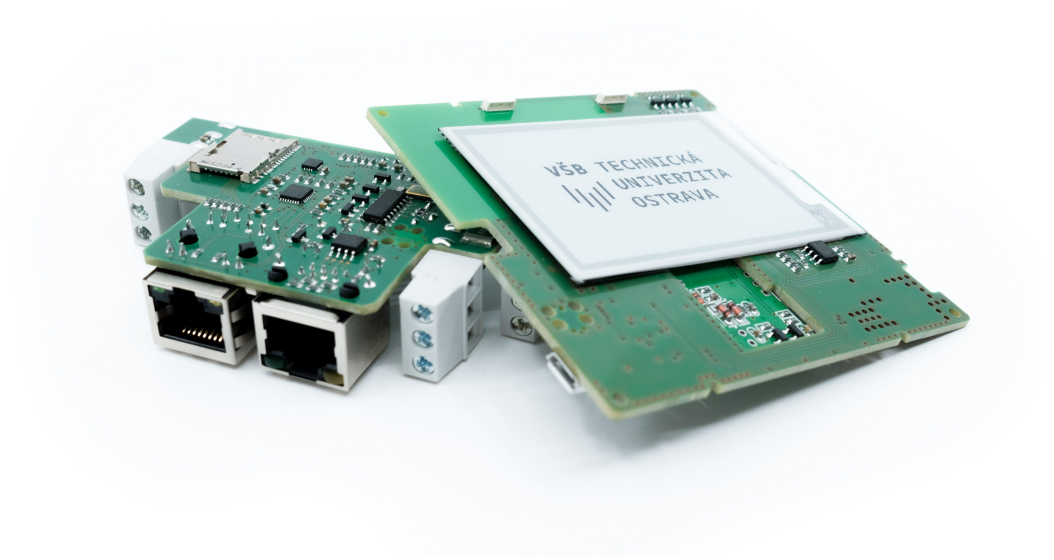
Příloha F

Fotografie přístupového systému

F.1 Ukázka modulů



Obrázek F.1: Sestava modulu čtečky a základny



Obrázek F.2: HW řešení čtečky a základny

Obrázek F.3: Modul čtečky