Masarykova univerzita
Fakulta informatiky

Identifikácia užívateľov pomocou biometrickej autentifikácie integrovanej do Windows 10

Diplomová práca

Bc. Ján Gábela

Brno, jeseň 2016

Namiesto tejto stránky vložte kópiu oficiálneho podpísaného zadania práce a prehlásenie autora školského diela.

Prehlásenie

Prehlasujem, že táto diplomová práca je mojím pôvodným autorským dielom, ktoré som vypracoval samostatne. Všetky zdroje, pramene a literatúru, ktoré som pri vypracovaní používal alebo z nich čerpal, v práci riadne citujem s uvedením úplného odkazu na príslušný zdroj.

Bc. Ján Gábela

Vedúci práce: Ing. Mgr. et Mgr. Zdeněk Říha, Ph.D.

Poďakovanie

Chcel by som poďakovať vedúcemu práce Ing. Mgr. et Mgr. Zdeňkovi Říhovi, Ph.D. za odborné vedenie a prejavenú ochotu. Tiež sa chcem poďakovať konzultantovi Mgr. Ondřejovi Gabrhelíkovi za venovaný čas, hodnotné komentáre a poskytnuté zdroje potrebné pri vývoji. Poďakovanie patrí aj Mgr. Mariánovi Čečundovi za poskytnuté konzultácie ohľadom možností integrácie implementovaného riešenia do existujúcej aplikácie. V neposlednom rade sa chcem poďakovať svojej rodine a priateľom za trpezlivosť a podporu.

Zhrnutie

Diplomová práca popisuje FIDO a jej špecifikácie. Väčší dôraz kladie na analýzu špecifikácie FIDO UAF. Ďalej analyzuje systém pre biometrickú autentifikáciu, ktorý je integrovaný do operačného systému Windows 10 a analyzuje vzťah k FIDO UAF špecifikácii. Na základe analýzy je implementovaná aplikácia, ktorá pridáva podporu pre spracovanie UAF protokolu s využitím prostriedkov integrovanej biometrickej autentifikácie v systéme Windows 10. V poslednom kroku je analyzovaná možnosť integrácie biometrickej autentifikácie do existujúcej aplikácie, ako náhrada za autentifikáciu užívateľa pomocou PINu.

Kľúčové slová

FIDO, Fast IDentity Online, UAF, Universal Authentication Framework, Microsoft Passport, Windows 10, Windows 10 Mobile, Windows Hello, UWP, Universal Windows Platform

Obsah

1 **Úvod** 1
2 **Autentifikácia na internete v súčasnosti** 3
2.1 *Bezpečnosť hesiel* 3
3 **FIDO aliancia a špecifikácie FIDO aliancie** 5
3.1 *Špecifikácie FIDO aliancie* 6
3.1.1 UX – Dvojfaktorová autorizácia (U2F) 6
3.1.2 UX – Autentifikácia bez použitia hesla (UAF) . . 7
3.1.3 Ako FIDO funguje 7
4 **Universal Authentication Framework** 10
4.1 *Prehľad dokumentov špecifikácie UAF* 10
4.2 *Architektúra a komponenty UAF* 11
4.2.1 FIDO Autentizátor 11
4.2.2 Abstraktná vrstva autentizátora (ASM) 12
4.2.3 FIDO klient 13
4.2.4 UAF protokol 13
4.2.5 Server 13
4.2.6 Overenie metadát autentizátora 14
4.3 *Prípady použitia* 14
4.3.1 Registrácia autentizátora 15
4.3.2 Autentifikácia 16
4.3.3 Dodatočná autentifikácia 18
4.3.4 Potvrdenie transakcie 18
4.3.5 Zrušenie registrácie autentizátora 19
4.4 *Súkromie užívateľov* 20
5 **Biometrická autentifikácia zabudovaná do Windows 10** . . 21
5.1 *Windows Hello* 21
5.1.1 Výkonnostné požiadavky 22
5.1.2 Typy Windows Hello autentizátorov 23
5.2 *Microsoft Passport* 26
5.2.1 Ako Microsoft Passport pracuje 27
5.2.2 Microsoft Passport API 29
5.3 *Podpora FIDO vo Windows 10* 30
6 **UWP UAF** 33
6.1 *Návrh* 33
6.2 *Možnosti komunikácie aplikácii vo Windows 10* 36

6.2.1 Komunikácia pomocou spustenia URI protokolu 37
6.2.2 Spustenie URI protokolu určenou aplikáciou . . 38
6.2.3 Spustenie URI protokolu a čakanie na výsledok 38
6.2.4 Zdieľanie údajov medzi aplikáciami 39
6.2.5 Zistenie podpory spracovania URI protokolu v systéme 39
6.2.6 *Application Services* – Služby aplikácie 40
6.3 *UwpUaf Client API a UwpUaf ASM API* 40
6.4 *UwpUaf demo aplikácia* 42
6.5 *UwpUaf klient demo* 44
6.6 *UwpUaf ASM demo* 46
6.6.1 Hello autentizátor 47
6.6.2 Fingerprint simulátor 50
6.7 *Integrácia UwpUaf riešenie do existujúcej aplikácie* 52
7 **Záver** 56
Literatúra 58
A **Možnosti komunikácie aplikácii vo Windows 10 – Ukážky kódov** 63
B **Obsah elektronickej prílohy** 67

Zoznam tabuliek

5.1 Prijateľné požiadavky Windows Hello biometrických autentizátorov *24*

Zoznam obrázkov

4.1 Architektúra FIDO UAF *12*
4.2 Registrácia autentizátora *16*
4.3 Autentizácia *17*
4.4 Potvrdenie transakcie *18*
4.5 Zrušenie registrácie autentizátora *19*
6.1 Detailný pohľad na architektúru FIDO UAF na strane klienta *34*
6.2 UwpUaf architektúra *35*
6.3 UwpUaf demo - Hlavná stránka aplikácie *43*
6.4 UwpUaf demo - Demonštrácia registrácie *43*
6.5 UwpUaf demo - Demonštrácia overenia užívateľa *44*
6.6 UwpUaf demo - Demonštrácia zrušenia registrácie *44*
6.7 UwpUaf demo - Zobrazenie odpovede spracovania registrácie *45*
6.8 UwpUaf demo - Zobrazenie odpovede spracovania overenia užívateľa *45*
6.9 UwpUaf demo - Zobrazenie spracovania zrušenia registrácie *46*
6.10 UwpUaf Client demo - Výber autentizátora *47*
6.11 UwpUaf Client demo - Zrušenie registrácie *47*
6.12 UwpUaf ASM demo - Overenie identity užívateľa pomocou biometriky očnej dúhovky *48*
6.13 UwpUaf ASM demo - Úspešné overenie identity užívateľa pomocou biometriky očnej dúhovky *48*
6.14 UwpUaf ASM demo - Možnosti overenia vo Windows Hello *49*
6.15 UwpUaf ASM demo - Overenie identity užívateľa pomocou PINu *49*
6.16 UwpUaf ASM demo - Zrušenie registrácie Windows Hello *50*
6.17 UwpUaf ASM demo - Fingerprint autentizátor simulátor, Registrácia *50*
6.18 UwpUaf ASM demo - Fingerprint autentizátor simulátor, Autentifikácia *51*

6.19 UwpUaf ASM demo - Zrušenie registrácie Fingerprint autentizátor simulátora *51*
6.20 Zjednodušená architektúra PT aplikácie *52*

1 Úvod

Metódy autentifikácie užívateľa na internete v súčasnosti sú vo väčšine prípadov založené na princípe znalosti niečoho, čo užívateľ pozná, najčastejšie heslo. Tieto metódy máju svoje problémy a prirodzene sa hľadajú nové, bezpečnejšie, metódy autentifikácie užívateľa na internete.

Jednou z nových metód je metóda, ktorá na autentifikáciu užívateľa, na internete, využíva lokálnu biometrickú autentifikáciu užívateľa v zariadení. Zariadenie, samozrejme, musí obsahuje nejaký biometrický senzor, alebo sa biometrický senzor dá k zariadeniu pripojiť. Metóda a jej špecifikáciu vyvíja FIDO aliancia.

Túto metódu autentifikácie pridal Microsoft do svojho operačného systému Windows 10, v ktorom predstavil úplne nový biometrický systém, ktorý je integrovaný priamo do operačného systému.

Úlohou práce je popis FIDO aliancie a jej špecifikácií s dôrazom na špecifikáciu Universal Authentication Framework protokolu. Následne podľa špecifikácia implementovať FIDO klienta a FIDO autentizátor, ktorý bude využívať biometrický systém integrovaný do Windows 10. Riešenie implementovať na mobilné zariadenie s operačným systémom Windows 10 Mobile. A na koniec, implementované riešenie sa pokúsiť integrovať do existujúcej aplikácie ako náhradu za autentifikáciu pomocou PINu.

Kapitola 2 stručne zhrňuje najčastejšie používané metódy autentifikácie v súčasnosti na internete a popisuje bezpečnostné problémy hesiel.

Kapitola 3 predstavuje FIDO alianciu a špecifikácie ktoré vyvíja. Nasledujúca kapitola 4 podrobnejšie rozoberá jednu zo špecifikácii vyvíjanú alianciou. Táto špecifikácia, Universal Authentication Framework, má za úlohu nahradiť autentifikáciu užívateľov pomocou hesiel za autentifikáciu pomocou lokálnej biometrickej autentifikácie užívateľa v zariadení užívateľa.

Kapitola 5 popisuje biometrickú autentifikáciu, ktorá je zabudovaná do Windows 10. S tým súvisia technológie Microsoft Passport a Windows Hello, ktoré sú v kapitole popísané podrobnejšie. V kapitole sa naviac dozvieme informácie o podpore FIDO v systéme Windows 10.

Praktickej časti práce sa venuje kapitola 6. Praktická časť Kapitola na začiatku popisuje návrh implementovaného riešenia. Ďalej predstavuje všetky implementované časti riešenia. Nakoniec kapitoly je popísaná snaha o integrovanie riešenia do existujúcej aplikácie.

2 Autentifikácia na internete v súčasnosti

V súčasnej dobe, metódy autentifikácie založené na schéme identifikátor (ID)/heslo, sú najčastejšie používaným riešením autentifikácie na internete [1]. ID/heslo schéma je jednoduchá na implementáciu, má nízku zavádzaciu cenu, ale existuje množstvo problémov z bezpečnostného hľadiska [2], ktoré si stručne zhrnieme v nasledujúcej kapitole 2.1.

Ďalšou autentifikačnou metódou sú jednorazové heslá (OTP[1]). Jednorazové heslá majú časovo obmedzenú platnosť a môžeme ich získať rôznymi spôsobmi. Napríklad, OTP môžeme získať v SMS zaslaním do telefónu alebo si ho môžeme vygenerovať buď použitím špecifickej aplikácie, ktorá zdieľa so SP určité tajomstvo, alebo pomocou špeciálneho hardvéru, OTP kalkulačky. Avšak, jednorázové heslá sa vo väčšine prípadov používajú, ako druhý faktor autentifikácie k autentifikácii pomocou ID a hesla.

2.1 Bezpečnosť hesiel

Podľa [3], ktorá cituje iné zdroje, mal v roku 2007 priemerný užívateľ 25 účtov, 6,5 hesiel a vykonával 8 prihlásení denne. Dnes je to oveľa horšie. Analýza 6 miliónov účtov ukázala, že 10,000 bežných hesiel sprístupní až 99,8% užívateľských účty. To prakticky znamená, že iba 0,2% užívateľov si zvolilo bezpečné heslo. Pri pohľade na heslá bankových účtov zistíme, že 73% užívateľov zdieľa svoje heslo k internetovému bankovníctvu s aspoň jednou nefinančnej stránkou, čo znamená, že keď nebankové stránky budú napadnuté a okradnuté, tak bankový účet bude ohrozený. Štúdii, ktoré sa zaoberajú bezpečnosťou hesiel, nájdeme veľa. Preto si ich len v krátkosti zhrnieme.

Heslo je tajná informácia, ktorá je zdielaná medzi užívateľom a SP. Ako taká môže byť ukradnutá útočníkom na strane klienta alebo na strane servera. Každé miesto, na ktorom je heslo zadávané, spracovávané alebo uložené je zraniteľné. Heslo môže byť odhalené či už

1. skratka pre One-Time Password, v preklade jednorázové heslo

útokom hrubou silou alebo slovníkovým útokom. S rastúcim počtom prepojených systémov a aplikácii rastú i nároky na pamätanie si rôznych prihlasovacích údajov. Preto si užívatelia pamätanie hesiel zjednodušujú tým, že volia heslá, ktoré sa ľahko pamätajú (i keď splňujú bezpečnostné požiadavky na heslo, nemusí odhalenie hesla trvať dlho, napr. heslo *P@ssw0rd1*), použitím jedného hesla na viacerých miestach (tu nastáva problém, ak je použité s rovnakým užívateľským menom, teda strata alebo krádež prihlasovacích údajov na jednom systéme má za následok získanie prihlasovacích údajov k iným systémom). Exspirácia hesla tiež nemusí viesť k bezpečnejšiemu heslu. Užívatelia si väčšinou volia nové heslo, ktoré sa podobá starému heslu (napr. *P@ssw0rd1* a nové heslo je potom *P@ssw0rd12*) [4].

3 FIDO aliancia a špecifikácie FIDO aliancie

Fast IDentity Online (FIDO) Alliance je nezisková organizácia, ktorá bola založená v Júli 2012 s cieľom riešiť chýbajúcu schopnosť spolupracovať medzi zariadeniami silnej autentifikácie užívateľa, ako aj problémy spojené s vytváraním a zapamätaním si viacerých užívateľských mien a hesiel [5].

Poslanie aliancie je zmeniť povahu online[1] autentifikácie užívateľa pomocou nasledujúcich bodov [6]:

- Vytvorením technických špecifikácií, ktoré budú definovať otvorenú, škálovateľnú množinu mechanizmov schopných spolupráce, ktoré znižujú závislosť na autentifikácii užívateľa pomocou hesla.
- Vytvorením operačného programu, ktorý pomôže zabezpečiť úspešnú akceptáciu vytvorených špecifikácii po celom svete.
- Odoslaním overených technických špecifikácii uznávaním organizáciám, ktoré sa venujú tvorbe štandardov na formálnu štandardizáciu.

Organizácie má za cieľ uskutočniť revolúciu v online autentifikácii užívateľa za pomoci štandardizovaného a otvoreného protokolu, podporovaného prednými firmami na trhu s IT technológiami, ktorý prinesie užívateľom okrem väčšej bezpečnosti autentifikácie aj ľahké a pohodlné použitie. Tento nový štandard[2] pre rôzne bezpečnostné zariadenia alebo pre rozšírenia do internetových prehliadačov dovoľuje rôznym internetovým stránkam alebo aplikáciám komunikovať so širokou škálou existujúcich alebo budúcich zariadení podporujúcich FIDO protokol [7].

FIDO Alliance definuje dve sady špecifikácii pre dva rôzne protokoly, ktoré odrážajú ich odlišnú použiteľnosť (UX). Viac informácii v nasledujúcej kapitole 3.1.

---

1. V online prostredí, teda v prostredí internetu.
2. FIDO Alliance vo svojich cieľoch hovorí štandarde, ale v čase písania práce štandardizácia existujúcej špecifikácie neexistovala.

- *Universal Authentication Framework (UAF)* – nahradenie autentifikácie užívateľa heslom za autentifikáciu užívateľa pomocou biometriky alebo PINu.
- *Universal Second Factor (U2F)* – k autentifikácii pomocou hesla pridáva silnú dvojfázovú autentifikáciu.

Oba protokoly zdieľajú rovnaké FIDO dizajnové princípy týkajúce sa jednoduchosti použitia a súkromia užívateľa [7]:

- Žiadna tretia strana v protokole.
- Žiadne tajné informácie uložené na serveri, iba verejné šifrovacie kľúče.
- Biometrické údaje (ak sú použité) nikdy neopustia zariadenie, na ktorom je autentifikácia za pomoci biometriky prevedená.
- Žiadna spojitosť medzi službami.
- Žiadna spojitosť medzi účtami.

3.1 Špecifikácie FIDO aliancie

Finálna FIDO 1.0 špecifikácia, bola uverejnená v decembri 2014 [8]. Ako bolo spomenuté, obsahuje dve sady špecifikácii pre dva rôzne protokoly, U2F a UAF, ktoré poskytujú odlišnú použiteľnosť.

3.1.1 UX – Dvojfaktorová autorizácia (U2F)

Je podporovaný Universal Second Factor špecifikáciou a protokolom. Dovoľuje online službám vylepšiť bezpečnosť existujúcej infraštruktúry hesiel pridaním dvojfaktorového overenia užívateľa. Užívateľ sa prihlasuje pomocou mena a hesla ako predtým ale služba môže vyzvať užívateľa na predloženie zariadenia na dvojfázové overenie. Silné dvojfaktorové overenie užívateľa dovoľuje zjednodušiť heslo užívateľa bez ohrozenia bezpečnosti. Výhodou implementácie protokolu U2F je v zjednotení komunikácie medzi aplikáciou a zariadeniami, ktoré slúžia ako druhý faktor autentifikácie užívateľa (tokeny). V podstate

sa jedná o zjednotenie komunikačného rozhrania pre tokeny. To znamená, že môžu byť využité rôzne tokeny od rôznych výrobcov bez nutnosti inštalovať špeciálny softvér pre tokeny. Pre viac informácii odkážem čitateľa na stránky špecifikácie *Universal 2nd Factor (U2F) Overview*[3].

3.1.2 UX – Autentifikácia bez použitia hesla (UAF)

Je podporovaný Universal Authentication Framework špecifikáciou a protokolom. Užívateľ najskôr registruje na online službe zariadenie výberom lokálneho autentifikačného mechanizmu, ako napríklad potiahnutie prsta, pozretie do kamery, hovorenie do mikrofónu, zadanie PINu, atď. UAF protokol dovoľuje vybrať mechanizmus, ktorým vyzve užívateľa na overenie. Po jednorazovej registrácii, užívateľ jednoducho opakuje lokálnu autentifikáciu hocikedy sa potrebuje prihlásiť do online služby. Užívateľ viac nemusí zadávať meno a heslo, keď sa autentifikuje z registrovaného zariadenia. UAF taktiež dovoľuje kombinovanie viacerých autentifikačných mechanizmov, ako napríklad autentifikácia pomocou otlačku prsta + zadanie PINu [9]. Viacej informácii sa dozvieme v kapitole 4.

3.1.3 Ako FIDO funguje

Táto kapitola poskytuje vysokoúrovňový pohľad na FIDO protokoly.

Protokoly používajú štandardné techniky asymetrickej kryptografie na poskytnutie bezpečnejšej autentifikácie. Počas procesu registrácie na online službou, klientske zariadenie užívateľa vytvára nový pár kľúčov. Súkromný kľuč si ponecháva a verejný kľúč registruje na službe. Autentifikácia je prevedená overením, že klient je vlastníkom súkromného kľúča, tým že podpíše výzvu (challange) servera. Súkromný kľuč môže byť použitý iba potom, ako je sprístupnení lokálne na zariadení užívateľa. Odomknutie kľúča je dosiahnuté pomocou užívateľsky prívetivou a bezpečnou akciou ako potiahnutím prsta[4],

3. `https://fidoalliance.org/specs/fido-u2f-v1.0-nfc-bt-amendment-20150514/fido-u2f-overview.html`

4. PayPal potvrdzovanie platieb pomocou snímača otlačkov prstov v zariadení Galaxy S5. Pozri `https://www.paypal-pages.com/samsunggalaxys5/us/index.html`.

zadaním PINu, hovorením do mikrofónu, vložením zariadenia na dvojfázové overenie a stlačením tlačidla na zariadení[5].

Protokoly sú navrhnuté chrániť súkromie užívateľov. Neposkytujú žiadne informácie, ktoré by mohli byť použité rôznymi online službami spolupracovať na sledovaní užívateľov naprieč službami. Biometrické údaje, ak sú použité, nikdy neopustia zariadenie užívateľa [9].

Registrácia

- Užívateľ je vyzvaný aby vybral dostupný FIDO autentizátor[6], ktorý splňuje bezpečnostné politiky stanovené online službou.
- Užívateľ odomkne FIDO autentizátor pomocou čítačky otlačku prstov, tlačidla na zariadení pre dvojfázové overenie, zadaním PINu alebo inou metódou.
- Zariadenie užívateľa vytvorí nový pár kľúčov asymetrickej kryptografie, ktorý je unikátny pre lokálne zariadenie (zariadenie užívateľa), službu a účet užívateľa.
- Verejný kľúč je zaslaný do služby a spojený s účtom užívateľa. Súkromný kľuč a ani informácie lokálnej autentifikačnej metódy (ako napríklad biometrické merania alebo šablóny) nikdy neopustia lokálne zariadenie.

Autentifikácia

- Online služba vyzve užívateľa aby sa prihlásil pomocou predtým registrovaného zariadenia, ktoré splňuje bezpečnostné politiky služby.
- Užívateľ odomkne FIDO autentizátor použitím rovnakej metódy ako pri registrácii.

---

5. *YUBICO* je príklad zariadenia, ktoré podporuje FIDO U2F protokol. Pozri `https://www.yubico.com/products/yubikey-hardware/fido-u2f-security-key/`.
6. FIDO Authenticator – popis v kapitole 4.2.1.

- Zariadenie použije identifikátor užívateľa poskytnutého službou na výber správneho súkromného kľúča, ktorým je následne podpísaná výzva služby.
- Klientske zariadenie posiela podpísanú výzvu späť na server, ktorý ju overí pomocou uloženého verejného kľúča a prihlási užívateľa.

4 Universal Authentication Framework

Ako bolo spomenuté vyššie UAF je jeden z dvoch protokolov, ktoré navrhuje špecifikácia FIDO 1.0. V tejto kapitole si najskôr uvedieme prehľad dokumentov, ktoré obsahuje UAF špecifikácia. Prehľad dokumentov je uvedený v kapitole 4.1. Každý dokument popisuje určitú časť alebo komponentu architektúry UAF. Následne si predstavíme architektúru a jednotlivé komponenty architektúry UAF a hlavné prípady použitia protokolu UAF (kapitoly 4.2 a 4.3).

4.1 Prehľad dokumentov špecifikácie UAF

- *FIDO UAF Protocol Specification:* Definuje formát správ a pravidla pre spracovanie všetkých správ UAF protokolu.
- *FIDO UAF Application API and Transport Binding Specification:* Definuje rozhrania a schopnosť spolupráce pre klientsku aplikáciou využiť UAF. Zahrňuje metódy komunikácie webových a mobilných aplikácii s FIDO klient. Metódy komunikácie na mobilných platformách popisuje pre operačné systémy Android a iOS [10].
- *FIDO UAF Authenticator Commands:* Definuje nízkoúrovňové rozhranie, ktoré by mali UAF autentizátory implementovať pre podporu UAF protokolu.
- *FIDO UAF Authenticator-Specific Module (ASM) API:* Definuje softvérové rozhranie nad UAF autentizátorom, čo dáva štandardizovaný spôsob pre FIDO klienta detekovať a sprístupniť funkcionalitu autentizátora, pretože autentizátor môže byť pripojený cez rôzne fyzické rozhrania (USB, Bluetooth, atď.) [11].
- *FIDO UAF Authenticator Metadata Statements:* Definuje metadata informácie, ktoré opisujú tvar, vlastnosti a schopnosti autentizátorov.

- *FIDO UAF Authenticator Metadata Service:* Definuje metódu získania aktuálnych metadata informácii.
- *FIDO Registry of Predefined Values:* Definuje všetky reťazce a konštanty rezervované UAF protokolom.

Dodatočné dokumenty, ktoré poskytujú dôležité informácie relevantné k UAF:

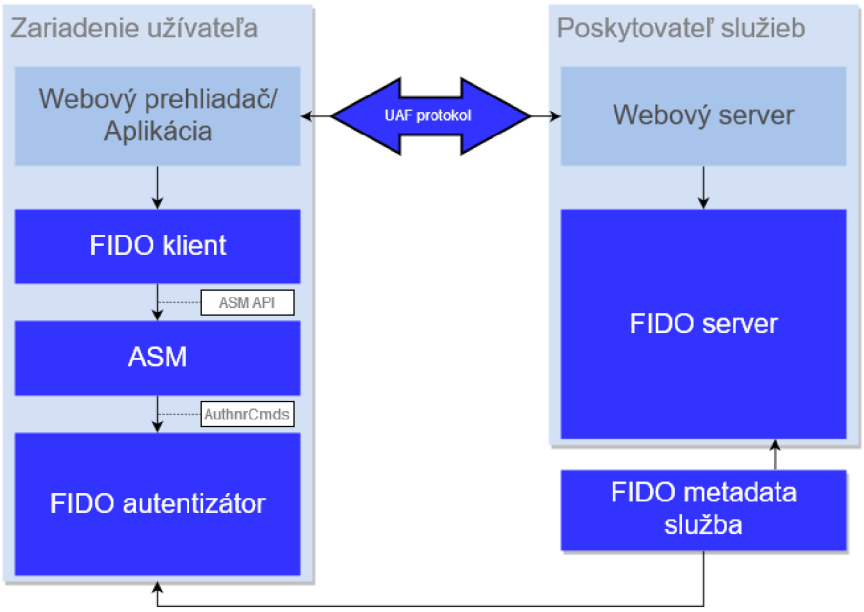
- *FIDO AppID and Facets Specification:* Definuje rozsah užívateľských poverení (user credentials) a ako dôveryhodná výpočtová základňa, ktorá podporuje izoláciu aplikácii, môže rozhodovať o prístupových právach, teda ktoré kľúče môžu byť použité ktorými aplikáciami alebo webovými pôvodcami (web origins).
- *FIDO Security Reference:* Poskytuje analýzu bezpečnosti FIDO protokolov. Obsahuje detailnú analýzu bezpečnostných hrozieb relevantných pre FIDO protokoly.
- *FIDO Glossary:* Definuje technické výrazy a frázy použité v dokumentoch špecifikácie.

4.2 Architektúra a komponenty UAF

Architektúra referenčného riešenia pre FIDO UAF je znázornená na obrázku 4.1. Jednotlivé komponenty architektúry špecifické pre FIDO sú bližšie popísané v nasledujúcich kapitolách.

4.2.1 FIDO Autentizátor

Jadro FIDO protokolov spočíva v koncepte autentizátora. FIDO autentizátor je bezpečná entita, buď pripojená k zariadeniu alebo umiestnená v zariadení užívateľa, ktorá môže vytvoriť pár kľúčov asymetrickej kryptografie, ktorý je spojený s poskytovateľom služieb. Pár kľúčov následne môže byť použitý na UAF silnú autentifikáciu pomocou UAF protokolov. Napríklad FIDO autentizátor môže poskytnúť odpoveď na kryptografickú výzvu použitým súkromného kľúča [13].



Obr. 4.1: Architektúra FIDO UAF. Obrázok vychádza z [12].

Hlavné funkcie, ktoré majú byť poskytnuté UAF autentizátorom sú nasledovné [14]:

- Overenie užívateľa overovacím mechanizmom, ktorý je zabudovaný do autentizátora.
- Vykonávanie kryptografických operácií (generovanie páru kľúčov, digitálny podpis).
- Vytvorenie dátovej štruktúry, ktorá môže byť spracovaná FIDO serverom.
- Poskytnutie overenia na FIDO server, ak je vstavaná podpora.

4.2.2 Abstraktná vrstva autentizátora (ASM)

Poskytuje jednotné API FIDO autentizátorov pre FIDO klientov, ktoré umožňuje používanie kryptografických služieb FIDO autentizátorov na podporu FIDO operácii. Poskytuje jednotné, nízkoúrovňové API na uľahčenie zavedenie FIDO autentizátorov od rôznych dodávateľov, ktoré majú ich vlastné nároky na nevyhnutné ovládače.

4.2.3 FIDO klient

Implementuje spracovanie UAF protokolu na strane klienta a je zodpovedný za následujúce funkcie [15].

- Interakciu s autentizátormi skrz ASM API.
- Interakciu s užívateľským agentom zariadenia (napr. mobilná aplikácia, webový prehliadač), použitím rozhrania, ktoré je špecifické pre agenta. Napríklad, špecifické rozšírenie pre prehliadač by mohlo použiť existujúce rozšírenie prehliadača alebo mobilná aplikácia by mohla použiť špecifické SDK. Užívateľský agent je potom zodpovedný za zasielanie správ UAF protokolu na server poskytovateľa služieb.

Architektúra UAF zaisťuje, že klientsky softvér môže byť implementovaný v celej rade systémov, operačných systémov a webových prehliadačov. Kým klientsky softvér je typicky špecifický pre platformu, interakcia medzi komponentami by mala byť rovnaká z platformy na platformu.

4.2.4 UAF protokol

UAF protokol prenášajú UAF správy medzi zariadením užívateľa a poskytovateľom služieb. Správy protokolu adresujú prípady použitia, ktoré sú popísané v kapitole 4.3.

4.2.5 Server

Implementuje spracovanie UAF protokolu na strane servera a je zodpovedný za následujúce funkcie [15].

- Interakciu s webovým serverom poskytovateľa služieb, ktorý komunikuje správy UAF protokolu s klientom skrz užívateľského agenta v zariadení užívateľa.
- Overovať osvedčenie autentizátora voči metadátam autentizátora, tak aby bolo zabezpečené, že iba dôveryhodné autentizátory sú registrované na serveri.

- Správu osvedčení autentizátorov registrovaných k užívateľským účtom na serveri.
- Vyhodnocovať odpoveď užívateľskej autentifikácie a potvrdenie transakcie k určeniu platnosti odpovede.

UAF server môže byť nasadený na serveri poskytovateľa služieb alebo môže byť poskytovaný ako služba tretej strany.

4.2.6 Overenie metadát autentizátora

V kontexte UAF overenie autentizátora je overenie tvrdení autentizátorov počas registrácie, že kľúče, ktoré generujú a/alebo merania, ktoré reportujú, pochádzajú z nefalšovaného zariadenia s certifikovanými charakteristikami. Podpis osvedčenia prenášaný v správe registračného protokolu je overený serverom. Autentizátory sú vytvorené s osvedčovacím súkromným kľúčom, ktorý je použitý k vytvoreniu podpisu a server overuje podpis použitým verejného kľúča certifikátu autentizátora umiestneného v metadátach autentizátora. Metadáta držiace osvedčovacie certifikáty sú zdieľané medzi servermi[1].

4.3 Prípady použitia

Predpokladá použitia autentizátora je, že užívateľ musí prejsť nejakým procesom nastavenia autentizátora, ktorý je špecifický pre autentizátor a je mimo rozsahu UAF protokolov. Napr. v prípade autentizátora pre snímanie otlačkov prstov, užívateľ musí registrovať svoj otlačok alebo svoje otlačky pre autentizátor, v tomto prípade snímač otlačku prstov. Raz ak je zápis dokončený, tak UAF autentizátor je pripravený na registráciu s poskytovateľom služieb, ktorý podporuje UAF [15].

Nasledujúce kapitoly uvedú vysokoúrovňový pohľad na protokol UAF z pohľadu prípadov použitia protokolu. Informácie o detailoch, ktoré zahŕňajú formát a tok UAF správ spolu s diagramami, sú uvedené v [16].

1. FIDO UAF Authenticator Metadata Service

4.3.1 Registrácia autentizátora

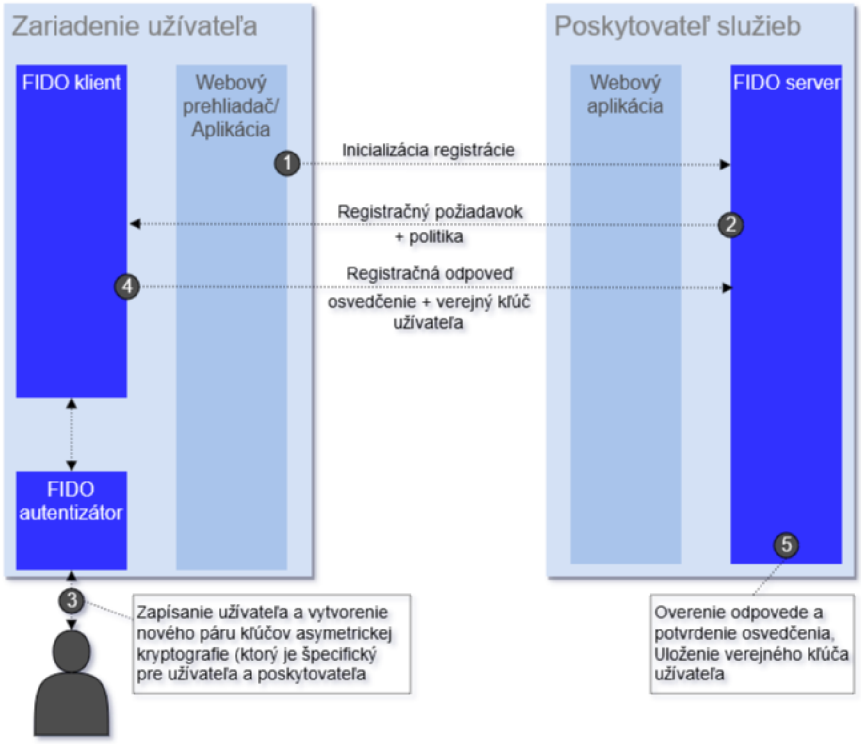
UAF registračný protokol umožňuje:

- Získať zoznam autentizátorov, ktoré sú dostupné zariadení užívateľa. Proces objavovania umožňuje selekciu zoznamu dostupných autentizátorov vzhľadom k bezpečnostnej politike poskytovateľa služieb, teda rozhodne, ktorým autentizátorom povolí alebo nepovolí registráciu.
- Overiť tvrdenia o osvedčení urobené autentizátorom, aby sa zaistilo, že autentizátor je autentický a dôveryhodný. K overeniu dôjde pomocou verejného kľúča certifikátu osvedčenia distribuovaného cez metadáta autentizátora.
- Registrovať autentizátor a spojiť ho s účtom užívateľa serveri. Akonáhle je osvedčenie autentizátora overené, server môže poskytnúť jedinečný identifikátor, ktorý je špecifický pre kombináciu servera a autentizátora. Tento identifikátor môže byť následne použitý pre budúcu interakciu medzi serverom a autentizátorom. Identifikátor nie je známy žiadnemu inému zariadeniu.

Najprv aplikácia požiada FIDO server o registráciu. Ten odpovedá zaslaním registračného požiadavku spolu so zoznamom povolených autentizátorov, ktorý spracováva FIDO klient v zariadení užívateľa. Súčasť registračného požiadavku je i náhodne vygenerovaná výzva. Užívateľ si vyberie s dostupných a povolených autentizátorov jeden, ktorý chce registrovať[2]. Následne sa užívateľ overí voči autentizátoru, a ak je toto overenie úspešné, tak autentizátor generuje pár kľúčov asymetrickej kryptografie. Pomocou vygenerovaného súkromného kľúča, ktorý sa uloží v zariadení užívateľa, vytvorí digitálny podpis serverovej výzvy a spolu príslušným verejným kľúčom a osvedčením autentizátora vytvára registračnú odpoveď, ktorá je zaslaná späť na FIDO server, kde je následne spracovaná. FIDO server overuje registračnú odpoveď, overuje podpis výzvy pomocou verejného kľúča, kontroluje osvedčenie autentizátora a v prípade, že nenastal žiaden

2. Pre registráciu ďalšieho FIDO autentizátora je nutné celý proces registrácie opakovať.

problém, tak si ukladá verejný kľúč, ktorý priraďuje k účtu užívateľa. Celý proces registrácie je zhrnutý na obrázku 4.2.



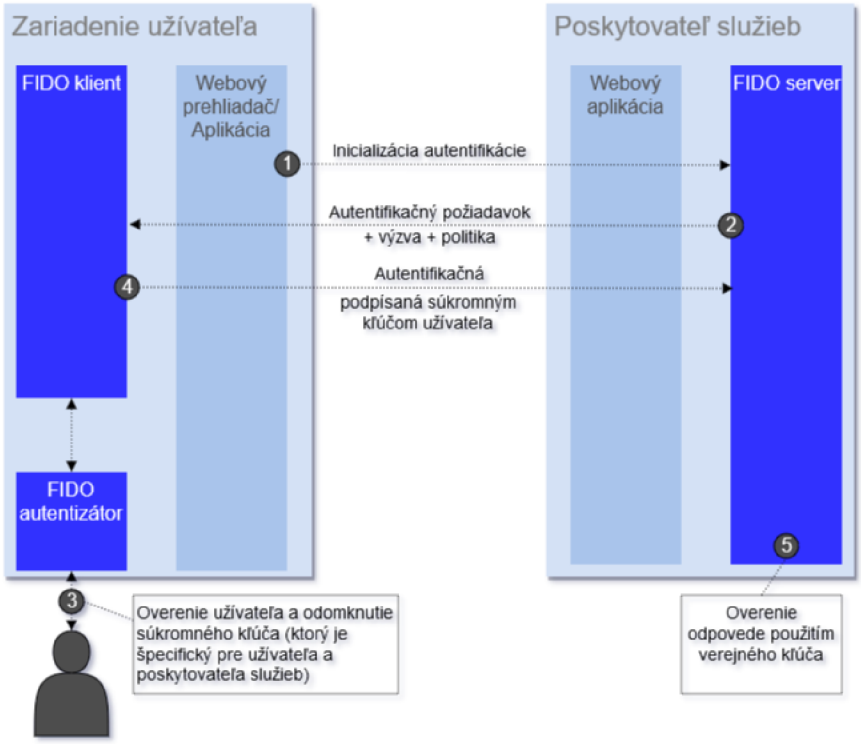
Obr. 4.2: Registrácia autentizátora. Obrázok vychádza z [17].

- Zrušenie registrácie autentizátora. Zrušenie registrácie je zvyčajne vyžadované, keď je účet užívateľa odstránený z RP. RP môže vyvolať zrušenie registrácie ako odpoveď na žiadosť autentizátora na odstránenie z pridruženého užívateľského účtu.

4.3.2 Autentifikácia

Po registrácii autentizátora, bude následne použitý vždy, keď sa užívateľ bude autentizovať voči poskytovateľovi služieb, za predpokladu, že je dostupný. Poskytovateľ služieb môže implementovať rôzne záložné stratégie pre situácie, keď autentizátor nie je dostupný. Tie

by mohli byť v rozpätí od povolenia bežných spôsobov prihlásenia so zníženými právami až k znemožneniu prihlásenia [15].



Obr. 4.3: Autentizácia. Obrázok vychádza z [18].

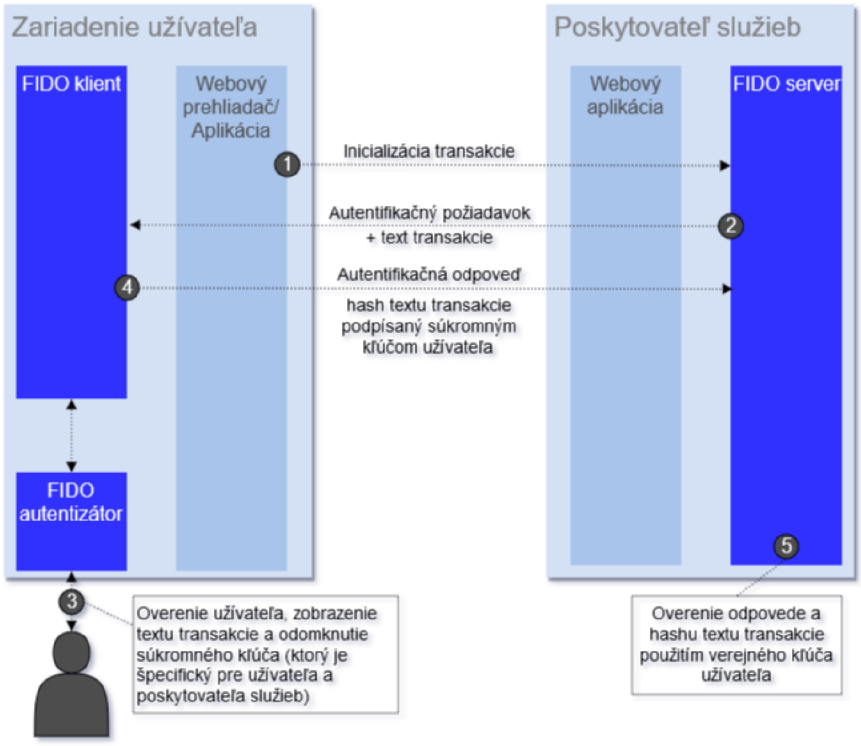
Proces autentifikácie, z užívateľského hľadiska, je podobný procesu registrácie. Užívateľ v tomto prípade zasiela na FIDO server požiadavok na autentifikáciu. Ten odpovedá zaslaním autentifikačného požiadavku so zoznamom registrovaných autentizátorov, ktorý spracováva FIDO klient v zariadení užívateľa. Súčasť autentifikačného požiadavku je takisto náhodne vygenerovaná výzva. Užívateľ si vyberie jeden z registrovaných autentizátorov, ktorým sa chce overiť voči serveru. Následne sa užívateľ overí voči autentizátoru, a ak je toto overenie úspešné, tak autentizátor odomkne príslušný súkromný kľúč, ktorým je vytvorený digitálny podpis serverovej výzvy a vytvára autentifikačnú odpoveď, ktorá je zaslaná späť na FIDO server, kde je následne spracovaná. FIDO server overuje autentifikačnú odpoveď a podpis výzvy kontroluje pomocou príslušného uloženého verejného

kľúča, ktorý bol priradený k účtu užívateľa počas procesu registrácie. Celý proces autentifikácie je zhrnutý na obrázku 4.3.

4.3.3 Dodatočná autentifikácia

V prípade, že užívateľ požiada o cennejšie zdroje poskytovateľa služieb, tak môže požiadať užívateľa o dodatočnú autentifikáciu pre vyššiu dôveryhodnosť. Proces je rovnaký ako v prípade procesu autentifikácie, ktorý je popísaný v predchádzajúcej kapitole, až na to, že nie je začatý užívateľom, ale serverom.

4.3.4 Potvrdenie transakcie



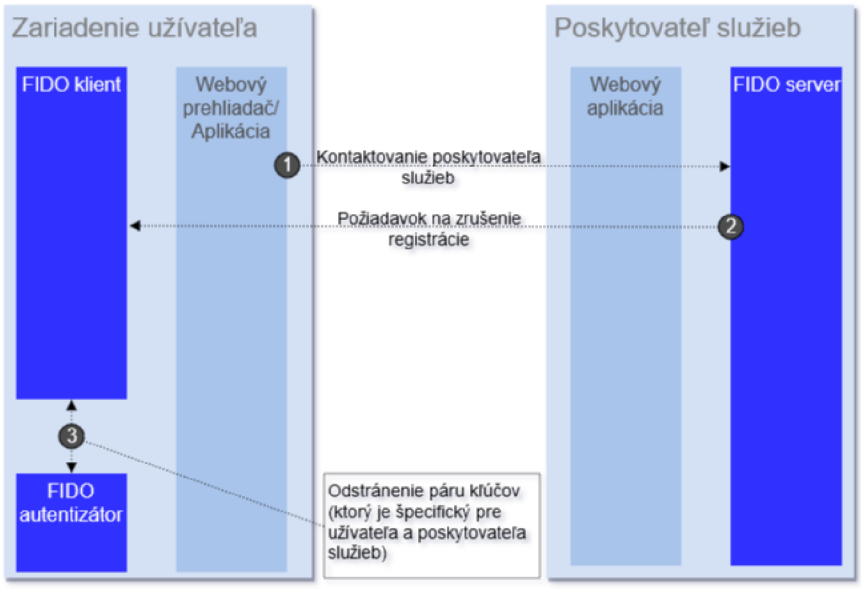
Obr. 4.4: Potvrdenie transakcie. Obrázok vychádza z [19].

Trochu pokročilejší prípad použitia je v bezpečnom spracovaní transakcii. Tento proces je však skoro rovnaký procesu autentifikácie, ktorý je popísaný v kapitole 4.3.2. Celý proces autentifikácie je

zhrnutý na obrázku 4.4. Rozdiel medzi autentifikačným procesom a procesom potvrdenia transakcie je následovný. Zo FIDO serveru sa zasiela autorizačný požiadavok, ktorý sa však líši od autentifikačného požiadavku tým, že miesto výzvy obsahuje transakčné údaje, z čoho následne vyplýva, že v autorizačnej odpovedi nie je podpis serverovej výzvy, ale podpis transakčných údajov. S transakčnými údajmi súvisí ešte jeden dôležitý rozdiel a to ten, že autentizátor musí vedieť tieto údaje zobrazovať užívateľovi, aby užívateľ vedel, čo podpisuje[3].

4.3.5 Zrušenie registrácie autentizátora

Existujú určite situácie, kedy server potrebuje odobrať autentizátor pre určitého užívateľa. Napr. autentizátor užívateľa bol stratený alebo ukradnutý, atď. V týchto situáciách server môže požiadať autentizátor k odstráneniu kľúčov, ktoré sú spojené s užívateľským účtom [15]. Celý proces autentifikácie je zhrnutý na obrázku 4.5.



Obr. 4.5: Zrušenie registrácie autentizátora. Obrázok vychádza z [20].

3. Podpora pre *WYSIWYS – What You See Is What You Sing*, teda v preklade *Čo vidíš je to, čo podpisuješ.*

4.4 Súkromie užívateľov

Súkromie užívateľov má zásadný význam pre FIDO a je podporované dizajnom UAF. Kľúčové prvky súkromia užívateľov:

- Zariadenie nemá globálny identifikátor viditeľný naprieč poskytovateľmi služieb a nemá globálny identifikátor v rámci určitého poskytovateľa. To znamená, že nie je možné spojiť účty užívateľa medzi službami. Podobne ak dvaja užívatelia majú registrované účty na rovnakej službe, tak poskytovateľ služby nemá ako rozoznať, že oba účty zdieľajú jedno zariadenie.
- UAF protokol vytvára pár kľúčov asymetrickej kryptografie jedinečný pre zariadenie, užívateľa a službu.
- Operácie UAF protokolu vyžaduje minimálny súbor osobných údajov (nanajvýš užívateľského meno). Osobné údaje sú použité len pre potreby FIDO prípadov použitia, napríklad registráciu užívateľa, autentifikáciu užívateľa alebo autorizáciu.
- Overenie užívateľa je vykonané lokálne na zariadení užívateľa. UAF protokol neprenáša biometrické údaje na službu.
- Užívatelia explicitne schvaľujú použitie UAF s konkrétnym poskytovateľom služieb. Kľúče sú generované v priebehu registrácie až po súhlase používateľa.

5 Biometrická autentifikácia zabudovaná do Windows 10

S biometrickou autentifikáciou vo Windows 10 súvisia dve technológie, *Windows Hello* a *Microsoft Passport*, ktoré sú bližšie popísané v kapitolách 5.1 a 5.2. Následne, kapitola 5.3 analyzuje vzťah týchto Windows 10 technológii a FIDO.

5.1 Windows Hello

Windows Hello je biometrický autentifikačný systém predstavený s operačným systémom Windows 10. Je vstavaný priamo do operačného systému. Umožňuje autentifikovať užívateľov na základe biometriky otlačku prstov, tváre alebo očnej dúhovky. Windows Hello autentizátor bude ďalej v texte označovaný skrátene ako Hello. Hello je jedinečný pre kombináciu individuálneho zariadenia a konkrétneho užívateľa. To znamená, že nie je zdieľaný medzi zariadeniami, ani so servermi alebo volanými aplikáciami a nie je možné ho jednoducho vytiahnuť zo zariadenia. Ak viac užívateľov zdieľa zariadenie, každý potrebuje nastaviť svoj vlastný účet. Každý účet získa unikátny Hello v zdieľanom zariadení. Predstavme si Hello ako token, ktorý môžeme využiť k odomknutiu (alebo uvoľneniu) uložených prihlasovacích údajov. Hello sám o sebe neautentifikuje užívateľa voči aplikácii ani službe, ale uvoľňuje prihlasovacie údaje, ktoré môžu autentifikovať užívateľa. Inými slovami, Hello nie sú prihlasovacie údaje pre užívateľa, ale je druhý faktor pre Microsoft Passport [21].

V kombinácii s technológiou Microsoft Passport nahrádza tradičnú autentifikáciu pomocou hesla pri procese prihlásenia. Keď Hello overí užívateľa, autorizuje ho k prístupu do operačného systému Windows v zariadení. Čo však užívateľ nevidí je to, že Windows Hello na pozadí uvolní uložené poverenie užívateľa, ktoré sa použije ako druhý faktor autentifikácie systémom Microsoft Passport. Ale nie je to len pre prihlásenie do systému Windows. Windows Hello spolu s Micro-

soft Passport poskytujú bezpečný spôsob pre aplikácie autentifikovať užívateľa na konkrétnom zariadení [22].

Operačný systém Windows 10 predstavil, spolu s jeho predstavením, podporu pre tri typy Hello autentizátorov (PIN, biometrika otlačkov prstov a tváre). S Windows 10 Mobile pribudol ešte jeden typ (biometrika dúhovky). Na jednotlivé Hello typy sa pozrieme v nasledujúcich kapitolách a v prípade biometrických typov Hello sa pozrieme i na výkonnostné požiadavky na hardvér biometrických zariadení. Výkonnostné požiadavky na hardvér biometrických zariadení sú požadované firmou Microsoft pre použitie v systéme Windows Hello [23].

5.1.1 Výkonnostné požiadavky

Hlavným cieľom biometrickej verifikácie je jednoznačné a bezchybné potvrdenie identity osoby, realizované porovnaním jedinečných charakteristík človeka. Porovnávajú sa šablóny práve získanej biometrickej vzorky s referenčnými vzorkami, ktoré boli poskytnuté skorej, a ktoré slúžia ako vzor pre porovnanie. Cieľom je, aby oprávnená osoba bola pri verifikácii správne prijatá, a naopak neoprávnená osoba bola správne odmietnutá, keďže sa môže zámerne vydávať za niekoho iného. V praxi pracujeme s negatívmi oboch spomenutých javov. Teda s pravdepodobnosťou chybného odmietnutia a s pravdepodobnosťou chybného prijatia [24].

False Accept Rate (FAR) Pravdepodobnosť nesprávneho prijatia. Je pomer nesprávne prijatých overení, tj. biometrický autentifikačný systém zle rozpoznal nesprávnu osobu, k celkovému počtu prijatých overení. Napríklad 1:100 000 (*0.001%*) znamená, že 1 zo 100 000 overení bolo nesprávne. Toto meranie je považované za najdôležitejšie s ohľadom na bezpečnosť biometrického algoritmu.

True Accept Rate (TAR) Pravdepodobnosť správneho prijatia. Súčet FRR a TAR sa rovná 1.

False Reject Rate (FRR) Pravdepodobnosť nesprávneho odmietnutia. Je pomer nesprávne zamietnutých overení, tj. biometrický

autentifikačný systém zle rozpoznal správnu osobu, k celkovému počtu zamietnutých overení. Súčet TAR a FRR sa rovná 1.

Miera istoty (Confidence) Miera istoty v tvrdenie o FAR reprezentuje platnosť analýzy uskutočnenej pri preverovaní tvrdenia o FAR. V závislosti na cieľovom alebo deklarovanom FAR a význame cieľového prípadu použitia sa môže úroveň miery istoty meniť.

Biometrická vzorka Jedná sa o kompletnú biometrickú vzorku, ktorá je potrebná na vykonanie operácie overenie (verifikácie), tz. jeden prst je jedna biometrická vzorka. Napríklad, jeden otlačok prsta je potrebný na overenie. Ale v prípade očnej dúhovky môže algoritmus vyžadovať obe oči na overenie a tak sú obe oči považované za jednu vzorku. Avšak ak je na overenie potrebné iba jedno oko, tak každé oko môže byť považované za samostatnú biometrickú vzorku.

Biometrický podvrh Odkazuje na syntetické repliku biometrickej vzorky. Systémy detekcie podvrhov a živosti biometrických vzorkou sú závisle na použitej biometrickej metóde [23].

5.1.2 Typy Windows Hello autentizátorov

V kapitole si predstavíme dostupné typy Windows Hello autentizátorov a v prípade biometrických typov Hello uvedieme výkonnostné požiadavky na hardvér biometrických zariadení. Požiadavky biometrických autentizátorov sú zhrnuté v tabuľke 5.1.

PIN

Predtým ako budeme môcť použiť Windows Hello biometrické autentizátory v zariadení, je nutné nastaviť PIN Hello autentizátor. Po nastavení PINu je možné nastaviť biometrické Hello autentizátory. PIN autentizátor môžeme vždy použiť k odomknutiu prihlasovacích

1. Windows Hello biometrický autentizátor. Všetky musia povinne obsahovať opatrenia proti podvrhom a voliteľne obsahovať spôsob ako tieto opatrenia konfigurovať.
2. Efektívny, skutočný FRR s opatreniami proti podvrhom a s detekciou živosti vzorku.

Hello[1]	*FAR*	*FRR*	*Skutočný FRR*[2]
Biometrika tváre	$< 0.001\%$	$< 5\%$	$< 10\%$
Biometrika otlačkov prstov pre malé a veľké dotykové senzory	$< 0.001 - 0.002\%$	$< 5\%$	$< 10\%$
Biometrika otlačkov prstov pre swipe senzory	$< 0.002\%$	$< 5\%$	$< 10\%$
Biometrika očnej dúhovky	$< 0.001\%$	$< 5\%$	$< 10\%$

Tabuľka 5.1: Prijateľné požiadavky Windows Hello biometrických autentizátorov

údajov, aj keď je v zariadení nastavený jeden z biometrických autentizátorov ako preferovaný. PIN slúži ako záloha v prípade, že biometrický senzor je poškodený, nedostupný alebo nefunguje správne [4].

Biometrika tváre

Jedná sa o pokročilú 3D biometriku tváre, ktorá je možná za pomoci Intel®RealSense™kamery[3]. Touto kamerou disponuje senzor Kinect, ktorý môže byť využitý na autentifikáciu [25]. Ešte si môžeme spomenúť výrobcu senzoru Tobii na sledovanie pohybu očí, ktorý oznámil podporu biometriky tváre do systému Windows Hello [26].

Prijateľné požiadavky na rozpoznávanie tváre

- Opatrenia proti podvrhom sú podmienkou a konfigurácia opatrení je voliteľná, tak ako v prípade biometrických senzorov.
- FAR: $< 0.001\%$

3. `http://www.intel.com/content/www/us/en/architecture-and-technology/realsense-overview.html`

- FRR bez opatrení proti podvrhom a detekcie živosti vzorku: $<$ 5%
- Efektívny, skutočný FRR s opatreniami proti podvrhom a s detekciou živosti vzorku: $<$ 10%

Biometrika otlačkov prstov

Využíva kapacitné senzory na skenovanie otlačkov prstov. Čítačky však pre počítače s operačným systémom Windows boli dostupné niekoľko rokov, ale súčasná generácia senzorov je výrazne spoľahlivejšia a menej náchylná na chyby. Väčšina existujúcich čítačiek otlačkov prstov (či už externých alebo integrovaných do notebookov alebo USB klávesníc) pracuje s Windows 10 a je ich možné využiť so systémom Windows Hello [4].

Senzor na snímanie otlačkov prstov môžu byť dvojakého typu: *dotykový senzor* (s veľkou alebo malou dotykovou plochou) alebo *swipe*[4] *senzor*. Každý typ má vlastnú množinu požiadavkov, ktoré musia byť implementované výrobcom. Všetky však musia povinne obsahovať opatrenia proti podvrhom a voliteľne obsahovať spôsob ako ich konfigurovať.

Prijateľné požiadavky pre malé a veľké dotykové senzory

- FAR: $<$ 0.001 – 0.002%
- FRR bez opatrení proti podvrhom a detekcie živosti vzorku: $<$ 5%
- Efektívny, skutočný FRR s opatreniami proti podvrhom a s detekciou živosti vzorku: $<$ 10%

Prijateľné požiadavky pre swipe senzory

- FAR: $<$ 0.002%
- FRR bez opatrení proti podvrhom a detekcie živosti vzorku: $<$ 5%

4. z anglického swipe, ťahať, pretože otlačok je získaný potiahnutím prsta cez senzor

- Efektívny, skutočný FRR s opatreniami proti podvrhom a s detekciou živosti vzorku: $< 10\%$

Biometrika očnej dúhovky

V čase písania práce, predchádzajúce dva typy Windows autentizátorov dostupné len pre počítače (alebo notebooky), ktoré disponovali potrebným hardvérom. Biometrika očnej dúhovky bola v čase písania dostupná, len na mobilnom zariadení Microsoft Lumia 950. Takže, zatiaľ sme sa bavili o systéme Windows 10, ale v tomto sa budeme baviť o systéme Windows 10 pre mobilné zariadenia (Windows 10 Mobile).

Ku svojmu fungovaniu potrebuje diódu vyžarujúcu a senzor zachytávajúci svetlo. Ku skenovaniu očnej dúhovky sa zvyčajne využíva neviditeľné infračervené svetlo. Senzor tak môže pracovať pri malom svetle (napríklad v tmavej miestnosti) a tiež môže pracovať s čírymi kontaktnými šošovkami, okuliarmi a nezrkadlovými slnečnými okuliarmi [27].

Prijateľné požiadavky na rozpoznávanie očnej dúhovky

- Opatrenia proti podvrhom sú podmienkou a konfigurácia opatrení je voliteľná, tak ako v prípade biometrických senzorov.
- FAR: $< 0.001\%$
- FRR bez opatrení proti podvrhom a detekcie živosti vzorku: $< 5\%$
- Efektívny, skutočný FRR s opatreniami proti podvrhom a s detekciou živosti vzorku: $< 10\%$

5.2 Microsoft Passport

Microsoft Passport je dvojfaktorový autentifikačný systém, ktorý kombinuje špecifické zariadenie s PINom alebo biometrikami cez Windows Hello. Koncepčne je podobný čipovým kartám tým, že autentifikácia sa vykonáva pomocou kryptografických primitív a užívateľské

kľúče sú chránené v hardvéri odolnom proti neoprávnenej manipulácii. Je použiteľný i v prípade, že zariadenie užívateľa neobsahuje žiaden biometrický senzor, pretože pracuje s tak ako s Hello tak i s PINom [4].

V roku 2013 sa Microsoft pridal k FIDO aliancii. Microsoft Passport súvisí so špecifikácii vyvinutých FIDO alianciou, keďže Windows 10 je referenčnou implementáciou konceptov špecifikácii UAF a U2F. Viac v samostatnej kapitole.

Passport znižuje dve hlavné bezpečnostné riziká. Po prvé, odstraňuje znovu použitie hesiel k prihlasovaniu. Znižuje riziko, že poverenia užívateľa budú skopírované alebo znovu použité. Zo zariadenie s TPM modulom, v ktorom je sú uložené kľúče užívateľa, je zložíte ich získať. Po druhé, Passport nie je závislý na jednom, centralizovanom serveri, takže je odstránené riziko kompromitovania bezpečnosti tohoto servera. Toto odstránenie rizika je dôsledok dvoch skutočností. Passport nezdieľa žiadne tajné informácie so serverom a pre každú aplikáciu alebo službu vytvára nový pár kľúčov. Takže kompromitovanie servera a získanie prihlasovacích údajov spojených s užívateľským účtom nebude viesť k tomu, že by bol útočník schopný sa prihlásiť k účtu užívateľa, pretože získa iba verejný kľúč užívateľa.

5.2.1 Ako Microsoft Passport pracuje

Aby sme mohli začať využívať Microsoft Passport, musíme najskôr nastaviť Windows Hello v nastaveniach operačného systému v zariadení. To znamená, že je nutné, ako prvé, nastaviť PIN. Od momentu úspešného nastavenia PINu je Microsoft Passport pripravený k použitiu. Minimálnou požiadavkov je Microsoft Account užívateľa. Poznamenajme, že Microsoft Passport závisí na spárovaní zariadenia a prihlasovacích údajoch užívateľa, takže zvolený je spojený iba s kombináciou aktívneho účtu a špecifického zariadenia. Keď máme nastavený PIN môžeme nastaviť Hello autentizátory, ak sú nejaké dostupné.

Keď je dokončený tento proces, Microsoft Passport vygeneruje nový pár kľúčov v zariadení. TPM, ak je dostupný, generuje kľúče a ukladá súkromný kľuč. Ak nie je TPM dostupný, tak je kľuč generovaný a súkromný kľúč ukladaný v zašifrovanej podobe softvérom. Tento počiatočný kľúč sa označuje ako *Protector key*. *Protektor key* vzniká s nastavením gesta na prihlásenie (PIN alebo Hello). To zna-

mená, že každé gesto má vlastný *Protector Key*. *Protector Key* logicky obaľuje *Authentication key* do kontajnera. Microsoft Passport tiež vytvára *Administrative Key*, ktorý sa môže použiť v prípade resetovania poverení.

Kontajner v kontexte Microsoft Passport je iba skratka pre logické zoskupenie kľúčového materiálu. Windows 10 podporuje dva kontajneri. Default kontajner zoskupuje kľúče pre osobné účty (Microsoft Account alebo iného IdP spotrebiteľa) a enterprise kontajner, ktorý zoskupuje kľúče pre pracovné alebo školské účty.

Uviedli sme si niekoľko typov kľúče. Tieto kľúče chránia iné kľúče v nasledujúcej hierarchii: *Protector Key* je použitý na šifrovanie *Authentication key* a *Authentication key* je použitý na šifrovanie individuálnych kľúčov v kontajneri.

Kontajneri môžu obsahovať niekoľko typov kľúčov:

- *Authentication key* je vždy súkromný/verejný pár kľúčov asymetrickej kryptografie, ktorý je generovaný počas procesu registrácie.
- *Virtual smart card keys* sú vygenerované, keď je virtuálna čipová karta vytvorená a bezpečne uložené v kontajneri.
- *Secure/Multipurpose Internet Mail Extensions (S/MIME) keys and certificates* sú generované certifikačnou autoritou (CA). Kľúče sú spojené s S/MIME certifikátom užívateľa, ktorý môže byť uložený v Microsoft Passport kontajneri.
- *IdP key*. Tieto kľúče sú buď symetrické alebo asymetrické. Záleží na tom, ktorý typ používa IdP.

Ako bolo spomenuté skorej v texte, Microsoft Passport využíva na ochranu Kľúčov TPM modul, takže v zariadeniach, ktoré obsahujú TPM modul, sa TPM hardvér využíva na generovanie, ukladanie a spracovávanie kľúčov. Microsoft Passport štandardne nevyžaduje prítomnosť TPM modulu. Ak zariadenie neobsahuje TPM, tak generovanie, ukladanie a spracovávanie kľúčov sa prevádza softvérovo.

5.2.2 Microsoft Passport API

Hlavná trieda pre prácu s Microsoft Passport API je trieda *KeyCredentialManager*[5]. Trieda spravuje[6] užívateľské poverenia, ktoré sú spojené s aktuálnym užívateľom[7] a s aplikáciou, ktorá pracuje s API. API poskytuje nasledujúce metódy, s ktorými budeme pracovať pri implementácii riešenia, ktoré je popísané v kapitole 6:

- *IsSupportedAsync()* – metóda zisťuje, či je možné použiť API pre aktuálne zariadenie a užívateľa.
- *RequestCreateAsync(string id, bool overwriteExisted)* – metóda vytvára užívateľské poverenie (KeyCredential) pre aktuálneho užívateľa a aplikáciu. Poverenie je identifikované identifikátorom (*id*) predaným ako parameter metódy.
- *OpenAsync(string name)* – metóda získava predtým vytvorené užívateľské poverenie (KeyCredential) pre aktuálneho užívateľa, aplikáciu a daný identifikátor (*name*).
- *DeleteAsync(string name)* – metóda vymaže užívateľské poverenie (KeyCredential) pre aktuálneho užívateľa, aplikáciu a daný identifikátor (*name*).

Užívateľské poverenie reprezentuje trieda *KeyCredential*[8]. Trieda reprezentuje pár kľúčov asymetrickej kryptografie, ktorý je generovaný pri volaní *KeyCredentialManager.RequestCreateAsync* metódy manažéra. Poskytuje jednoduché API, ktoré tvorí metóda na získanie verejného kľúča a metóda na vytvorenie digitálneho podpisu. Trieda ponúka nasledujúce metódy:

- *RetrievePublicKey()* – metóda vracia verejný kľúč z dvojice asymetrických kľúčov.

---

5. Dokumentácia `https://msdn.microsoft.com/en-us/library/windows/apps/windows.security.credentials.keycredentialmanager.aspx`.
6. Poskytuje metódy na vytvorenie, získanie a mazanie užívateľských poverení.
7. V zmysle, aktuálne prihláseným užívateľom.
8. Dokumentácia `https://msdn.microsoft.com/en-us/library/windows/apps/windows.security.credentials.keycredential.aspx`.

- *RequestSignAsync(IBuffer data)* – metóda vytvára digitálny podpis zadaných údajov (*data*).

5.3 Podpora FIDO vo Windows 10

Z predchádzajúcich informácii, konkrétne z kapitol 4 a 5.2 môžeme pozorovať podobnosť systému Microsoft Passport a špecifikácie FIDO UAF. Microsoft deklaruje, že Microsoft Passport je štandardizovaný a toto tvrdenie spája so špecifikáciami FIDO aliancie [4]. Microsoft, ešte pred oficiálnym vydaním svojho operačného systému oznámil, že sa pripojil k FIDO aliancii, s ktorou pokračuje vo vývoji ďalšej verzii špecifikácie FIDO 2.0 [28]. Toto oznámenie prišlo dva mesiace po oficiálnom uvoľnení prvej verzie špecifikácie FIDO 1.0 (pozri 3.1). Na konferencii Build 2016[9] oznámil, že Microsoft Edge natívne podporuje skorú implementáciu FIDO 2.0, ktorá sa však premenovala na *Web Authentication*[10] špecifikáciu, a ktorá je vyvíjaná i v spolupráci s novovzniknutou *Web Authentication* pracovnou skupinou[11] konzorcia W3C.

V tejto kapitole sa pozrieme bližšie na to, do akej miery implementácia systému Microsoft Passport vo Windows 10 implementuje špecifikáciu FIDO 1.0, teda preberá z tejto špecifikácie, konkrétne sa zameriame na UAF protokol, ktorú sme popísali v kapitole 4. Najskôr si však zopakujme kľúčové body fungovania Microsoft Passport [29]:

- Passport poverenia sú založené na certifikáte alebo páre kľúčov asymetrickej kryptografie. Tieto poverenia sú viazané na konkrétne zariadenia.
- IdP (Active Directory, Azure AD alebo Microsoft Account) overuje identitu a spája Passport verejný kľúč s účtom užívateľa počas registrácie.
- Kľúče môžu byť vytvorené v hardvéri (TPM 1.2 alebo TPM 2.0) alebo softvérom.

9. `https://build.microsoft.com/`
10. `http://w3c.github.io/webauthn/`
11. `https://www.w3.org/Webauthn/`

- Autentifikácia je dvojfaktorová. Je to kombinácia kľúča alebo certifikátu viazaného na zariadenie a niečoho čo užívateľ pozná (PIN) alebo niečoho čo užívateľ je (Windows Hello). Passport "gesto"sa neprenáša medzi zariadeniami a nie je zdielané ani so serverom. Je uložené iba lokálne v zariadení.

- Súkromný kľúč nikdy neopustí zariadenie. Server má verejný kľúč, ktorý bol spojený s účtom užívateľa počas registrácie.

- Zadanie PINu a Windows Hello spúšťa operačný systém Windows 10, aby overil identitu užívateľa a autentizoval ho pomocou Passport poverení.

Microsoft Passport nepreberá celú špecifikáciu FIDO UAF, ale preberá a implementuje niektoré vlastnosti definované špecifikáciou. Ak sa pozrieme na architektúru špecifikácie na obrázku 4.1 a na definované vlastnosti pre FIDO autetizátor popísané v kapitole 4.2.1, tak môžeme Microsoft Passport v kombinácii s Windows Hello označiť za FIDO autentizátor. Pre potreby analýzy budeme vo zvyšku kapitoly uvažovať kombináciu Windows Hello a Microsoft Passport ako celok, a teda ako biometrický autentizátor. Pozrime sa na to bližšie. Nasledujúci zoznam zhŕňa implementáciu vlastností FIDO autentizátora v porovnaní s naším autentizátorom:

1. *Mechanizmus overenia užívateľa vstavaný do autentizátora.* Túto funkciu vo Windows 10 poskytuje biometrický systém Windows Hello.

2. *Vykonávanie kryptografických operácii.* Túto funkciu poskytuje systém Microsoft Passport, ktorý poskytuje všetky operácie potrebné pre podporu FIDO protokolu. Bližšie, tieto operácie sú vygenerovanie páru kľúčov asymetrickej kryptografie a podpísanie výzvy zo serveru.

3. *Vytvorenie dátovej štruktúry, ktorá je spracovaná FIDO serverom.* V tejto roli nevystupuje žiadna časť Windows 10.

4. *Poskytnúť overenie autentizátora pre FIDO server, ak existuje podpora pre overenie.* Microsoft Passport API má podporu pre získanie osvedčenia.

Ďalej, keď sa pozrieme na jednotlivé scenáre použitia, konkrétne registrácie a autentifikácie, tak zistíme, že sú takmer identické. To čo chýba nášmu autentizátoru je podpora pre FIDO protokol a spracovanie správ protokolu. V nasledujúcej kapitole 6 navrhneme a implementujeme riešenia, ktoré využije využije technológie Windows Hello a Microsoft Passport, na pridanie podpory FIDO protokolu podľa špecifikácie FIDO 1.0, na mobilnom zariadení s operačným systémom Windows 10.

6 UWP UAF

Cieľom tejto kapitoly je navrhnúť a implementovať riešenie, ktoré pridá podporu pre FIDO UAF protokol, podľa špecifikácie FIDO 1.0, na mobilnom zariadení s operačným systémom Windows 10. V implementácii sa pokúsime využiť vstavanú podporu operačného systému pre biometrickú autentifikáciu užívateľov, konkrétne technológie Windows Hello a Microsoft Passport[1]. Nami implementované riešenie sa volá *UWP UAF* a meno je kombináciou dvoch skratiek, ktoré majú symbolizovať spojenie Windows 10 (UWP[2]) a FIDO protokolu (UAF[3]).

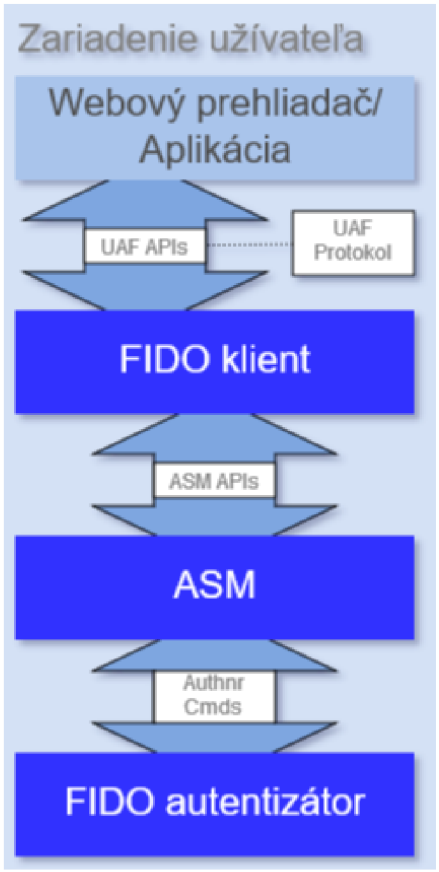
Najskôr sa budeme venovať návrhu architektúry riešenia a analyzujeme všetky potrebné požiadavky na implementáciu navrhnutého riešenia. Z analýzy nám vyplynú otázky, ktorým sa budeme venovať v nasledujúcich kapitolách. Po vyriešení otázok, ktoré je nutné vyriešiť k úspešnej realizácii riešenia, predstavíme demo aplikácie, ktoré demonštrujú funkčnosť realizácie.

6.1 Návrh

Pri návrhu architektúry UWP UAF musíme zvážiť detailnejší pohľad na architektúru UAF protokolu na strane zariadenia užívateľa, ktorá je zobrazená na obrázku 6.1. Na obrázku 6.1 vidíme jednotlivé časti UAF architektúry. Konkrétne komponenty FIDO klient, ASM a FIDO autentizátor. A takisto vidíme rozhrania medzi komponentami, pomocou ktorých medzi sebou komunikujú. V našom prípade je vytvorená zjednodušená architektúra UWP UAF, ktorá je zobrazená na obrázku 6.2.

V UAF aplikácia komunikuje s FIDO klientom pomocou rozhrania *UAF API*, ktoré je definované v dokumente [10] a je špecifikované pre mobilné operačné systémy Android a iOS. Chýba špecifikácia pre mobilný operačný systém Windows 10, z dôvodu, že špecifiká-

1. Pozri kapitolu 5
2. Je skratka pre *Universal Windows Platform*, čo je vývojárska platforma na tvorbu aplikácii, ktoré sa dajú spustiť na všetkých zariadeniach, na ktorých beží operačný systém Windows 10. Pre viac informácii pozri `https://msdn.microsoft.com/en-us/windows/uwp/get-started/whats-a-uwp`.
3. Pre viac informácii pozri kapitolu 4.

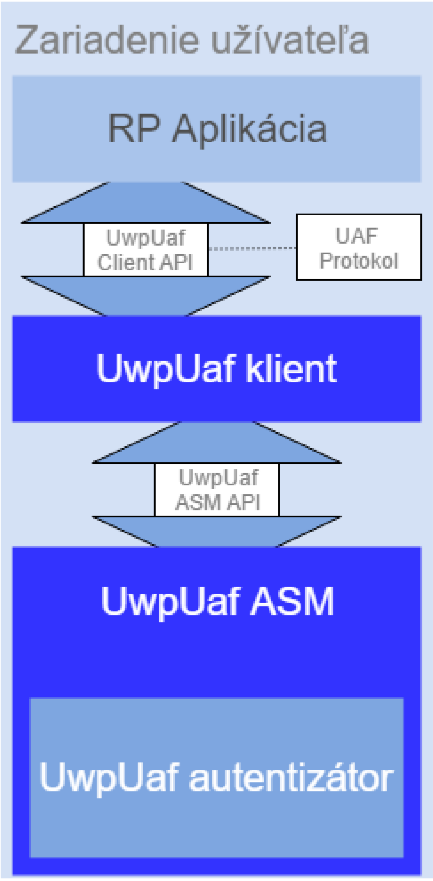


Obr. 6.1: Detailný pohľad na architektúru FIDO UAF na strane klienta. Obrázok vychádza z [30] a [31].

cia FIDO 1.0 vznikla a bola uvoľnená pred oficiálnym predstavením Windows 10. Z tohoto vyplýva, že je potrebné navrhnúť komunikačné rozhranie medzi aplikáciou a FIDO klientom pre operačný systém Windows 10 a platformu UWP. Toto rozhranie nazveme *UwpUaf Client API* a poskytuje komunikáciu s *UwpUaf klientom* (pozri obrázok 6.2). UwpUaf klient je názov pre FIDO klienta v našom UWP UAF riešení. Návrh riešenia pre UwpUaf Client API je popísaný v kapitole 6.3. Implementácia UwpUaf klienta je popísaná v kapitole 6.5.

Komunikácia FIDO klienta, v našom prípade UwpUaf klienta, a ASM, v našom prípade *UwpUaf ASM*, je realizovaná pomocou ASM API, v našom prípade *UwpUaf ASM API*. Špecifikácia [11], v tomto

prípade definuje komunikačné rozhranie iba pre mobilný operačný systém Android. Spôsob definovaného návrhu pre komunikáciu je podobný ako pre UAF API, a tak *UwpUaf ASM API* je navrhnuté spolu s UwpUaf Client API v kapitole 6.3. Musíme brať na vedomie, že FIDO klient môže komunikovať s viacerými ASM komponentami.



Obr. 6.2: UwpUaf architektúra

Posledné komunikačné rozhranie, zobrazené na obrázku 6.1, je rozhranie *Authnr Cmds*, ktoré umožňuje komunikovať komponente ASM s FIDO autentizátormi. Špecifikácia definuje nízko úrovňový protokol, sadu príkazov FIDO autentizátora, ktoré komponenta ASM volá. Autentizátory môžu mať rôznu podobu [14] a jedna ASM komponenta môže komunikovať s viacerými autentizátormi, ktoré sú pripojené

k zariadeniu pomocou rozličných fyzických rozhraní (SPI[4], USB, Bluetooth, atď.) [11]. V našom prípade, ako je zobrazené na obrázku 6.2, nie je toto rozhranie implementované a UwpUaf autentizátor je súčasťou komponenty UwpUaf ASM. Implementácia rozhrania Authnr Cmds má zmysel pre hardvérové autentizátory, ktoré môžu byť použité s rôznymi systémami, teda existuje ASM pre každý takýto systém. V našom prípade, sa však jedná o technológie, ktoré sú dostupné iba pre operačný systém Windows 10 a tak nie je nutné oddeľovať komponenty UwpUaf ASM a UwpUaf autentizátor. Implementácia UwpUaf ASM je popísaná v kapitole 6.6.

Prvou úlohou, ktorou sa budeme zaoberať je návrh a implementácia komunikačných rozhraní *UwpUaf Client API* a *UwpUaf ASM API*. Keďže špecifikácia nedefinuje návrh rozhrania pre operačný systém Windows 10 a platformu UWP, tak sa najskôr pozrieme na možnosti komunikácie medzi aplikáciami, ktoré sú poskytované programátorom vývojárskou platformou UWP vo Windows 10. Tieto možnosti sú popísané v nasledujúcej kapitole 6.2. Komunikačné rozhrania *UwpUaf Client API* a *UwpUaf ASM API* sú popísané v kapitole 6.3.

Riešenie *UWP UAF* sa skladá z troch aplikácií, aplikácie *UwpUaf Demo* (kapitola 6.4), aplikácie *UwpUaf Client demo* (kapitola 6.5) a aplikácie *UwpUaf Asm demo* (kapitola 6.6), ktoré predstavujú komponenty zobrazené na obrázku 6.2 a to *RP Aplikáciu*, *UwpUaf klienta* a *UwpUaf ASM*.

Zdrojové kódy riešenia sa nachádzajú na priloženom CD a zároveň boli umiestnené na GitHub portál, dostupné na webovej adrese `https://github.com/feki/UwpUaf`.

6.2 Možnosti komunikácie aplikácii vo Windows 10

Windows 10, teda platforma UWP poskytuje množstvo vylepšení oproti predchádzajúcej verzii operačného systému a vývojárskej platformy. V predchádzajúcej verzii operačného systému Windows 8.1 bolo možné komunikovať z jednej aplikácie do druhej pomocou súborovej alebo URI asociácie [33]. To znamená, že vývojár registruje operačnému systému buď typ súboru, v prípade súborovej asociácie,

4. *Serial Peripheral Interface* sériová zbernica [32].

alebo URI protokol, v prípade URI asociácie, ktoré aplikácia dokáže spracovať. Takže súborová asociácia dovoľuje spustiť aplikáciu podľa konkrétneho typu súbora. URI alebo protokolová asociácia dovoľuje spustiť aplikáciu podľa konkrétneho protokolu, čo je prvá časť URI[5], napr. `org.fidoalliance.fido.uaf-client+json://`. Údaje sa medzi aplikáciami predávajú pomocou obsahu súboru alebo ako súčasť URI. Príklad volania je vo výpise A.1.

V tejto kapitole sa pozrieme bližšie na nové možnosti komunikácie medzi aplikáciami, ktoré ponúka trieda `Launcher`[6] a to konkrétne nasledujúcim možnostiam:

- Spustenie URI protokolu (kapitola 6.2.1)
- Spustenie URI protokolu určenou aplikáciou (kapitola 6.2.2)
- Spustenie URI protokolu a čakanie na výsledok (kapitola 6.2.3)
- Zdieľanie údajov medzi aplikáciami (kapitola 6.2.4)
- Zistenie podpory spracovania URI protokolu v systéme (kapitola 6.2.5)
- App Services – Služby aplikácie (kapitola 6.2.6)

6.2.1 Komunikácia pomocou spustenia URI protokolu

Spustenie URI protokolu spôsobí spustenie aplikácie[7], ktorá dokáže narábať s daným URI protokolom. Údaje môžu byť medzi aplikáciami predávané, ako bolo spomenuté vyššie v texte, pomocou validného URI a to pomocou nasledujúcich častí URI: *path*, *query* a *fragment*[34]. Celé URI je poslané do aplikácie a spracované pomocou `OnActivated` udalosti [35], pozri výpis A.2. Podporu pre spracovanie URI protokolu je potrebné pridať do manifestu aplikácie, pozri výpis A.3.

5. Pozri [34], protokol je označený ako *scheme*.
6. `https://msdn.microsoft.com/en-us/library/windows/apps/windows.system.launcher.aspx`
7. Alebo inak povedané aktiváciu aplikácie.

6.2.2 Spustenie URI protokolu určenou aplikáciou

V UWP bola pridaná možnosť spustiť URI protokol zadanou aplikáciou, čoho docielime pomocou triedy `LauncherOptions`[8], ktorej inštancii nastavíme vlastnosť `TargetApplicationPackageFamilyName` a následne predáme objekt ako parameter metóde `LaunchUriAsync`. Príklad použitia je uvedený vo výpise A.4.

Na spustenie URI protokolu zadanou aplikáciou je nutné poznať jej *PackageFamilyName*, čo je unikátne meno aplikácie, ktoré dostáva aplikácia pri jej registrácii [36]. Hodnotu pre *PackageFamilyName* môžeme získať niekoľkými spôsobmi:

- z manifestu[9] [37]
- pomocou `Windows.ApplicationModel.Package.Current.Id.FamilyName`[9] [38]
- pomocou `Launcher.FindUriSchemeHandlersAsync`, pozri kapitolu 6.2.5

6.2.3 Spustenie URI protokolu a čakanie na výsledok

UWP umožňuje spustiť aplikáciu pomocou URI protokolu a počkať na výsledok. Teda volaná aplikácia môže vrátiť nejaký výsledok (nejakú odpoveď) aplikácii, ktorá ju spustila. Pre tento účel použijeme metódu `Launcher.LaunchUriForResultsAsync`. Spustenie URI protokolu s čakaním na výsledok nemôžeme vykonať bez špecifikovanej volanej aplikácie [37], musíme nastaviť *TargerApplicationPackageFamilyName*. Príklad volania môžeme vidieť vo výpise A.4 a príklad spracovania vo výpise A.2.

8. `https://msdn.microsoft.com/en-us/library/windows/apps/windows.system.launcheroptions.aspx`
9. Za predpokladu, že sme vývojármi aplikácie, s ktorou chceme komunikovať.

6.2.4 Zdieľanie údajov medzi aplikáciami

Okrem zdieľania údajov medzi aplikáciami pomocou URI, UWP pridáva novú možnosť a to pomocou `ValueSet`[10] dátovej štruktúry. *ValueSet* je slovník, ktorého kľúče sú reťazce a hodnoty sú objekty, ktoré sa dajú serializovať, jednoduché typy, ako celé čísla, čísla s plávajúcou desatinnou čiarkou, reťazce a bajtové polia [36]. Príklad použitia môžeme vidieť vo výpise A.4 a vo výpise A.2.

6.2.5 Zistenie podpory spracovania URI protokolu v systéme

Užitočné metódy, ktoré pribudli do triedy `Launcher`, zisťujú podporu spracovania URI protokolu v systéme a ďalšie užitočné informácie o aplikáciach, ktoré URI protokol dokážu spracovať. Jedná sa o nasledujúce metódy:

- `Launcher.QueryUriSupportAsync`[11] – nezisťuje žiadne informácie o aplikáciach. Volanie metódy vracia iba informáciu o tom, že sa v systéme nachádza aplikácia, ktorá dokáže spracovať dané URI, alebo nie. Alebo môžeme skontrolovať podporu spracovania URI protokolu konkrétnej aplikácie.
- `Launcher.FindUriSchemeHandlersAsync`[12] – narozdiel od prvej metódy, získava ďalšie informácie o aplikáciach, ktoré dokážu spracovať daný URI protokol. Volanie metódy vracia zoznam *AppInfo* objektov. V *AppInfo* sa nachádza informácia o *PackageFamilyName* aplikácie, ktorá je potrebná pre vyššie uvedené možnosti komunikácie, ale i pre komunikáciu pomocou *App Services*, ktoré sú predstavené v nasledujúcej kapitole 6.2.6.

Príklad volania je uvedený vo výpise A.5.

---

10. `https://msdn.microsoft.com/en-us/library/windows/apps/windows.foundation.collections.valueset.aspx`
11. `https://msdn.microsoft.com/en-us/library/windows/apps/windows.system.launcher.queryurisupportasync.aspx`
12. `https://msdn.microsoft.com/en-us/library/windows/apps/mt186434.aspx`

6.2.6 *Application Services* – Služby aplikácie

Je ďalším, úplne novým prostriedkom komunikácie medzi aplikáciami vo Windows 10. O *Application Services* vo Windows 10 môžeme premýšľať ako o webových službách v zariadení [39], teda poskytuje možnosť vytvorenia klient–server komunikácie medzi aplikáciami. Aplikačná služba je špeciálny druh úlohy, ktorá je spúšťaná na pozadí[13], a ktorá nevyžaduje spustenie volanej aplikácie do popredia [37], na rozdiel od aktivácie URI protokolu.

Na poskytnutie služby aplikáciou je potrebné vykonať nasledujúce kroky [40]:

1. Služba aplikácie je implementovaná ako úloha spúšťaná na pozadí. Úlohu je treba implementovať v Windows Runtime Component projekte.

2. Pridať odkaz na projekt so službou do našej aplikácie.

3. Pridať podporu pre protokol služby do manifestu aplikácie a spojiť ho s implementovanou službou.

Príklad implementácie poskytovateľa služby znázorňujú výpisy A.6 a A.7.

Implementácia klienta vyžaduje poznať názov protokolu služby a meno[14] aplikácie, ktorej službu chce využiť. Meno aplikácie je rovnaká informácia, ako v prípade komunikácie pomocou spustenia URI protokolu. Názov protokolu má obmedzenie na dĺžku a nesmie byť dlhšie ako 39 znakov. Príklad implementácie aplikácie, ktorá volá poskytovateľa služby aplikácie je uvedený vo výpise A.8.

6.3 UwpUaf Client API a UwpUaf ASM API

Kapitola popisuje komponenty *UwpUaf Client API* a *UwpUaf ASM API*, ktoré umožňujú komunikovať aplikácii tretej strany s aplikáciou

13. anglický výraz pre úlohu, ktorá sa spúšťa na pozadí je *background task*
14. `PackageFamilyName`

UwpUaf klient a komunikáciu *UwpUaf klient* aplikácie s aplikáciou *UwpUaf ASM*.

Implementácia komponentov vychádza zo špecifikácie komunikácie medzi aplikáciami pre platformu Android, ktorá je súčasťou špecifikácie FIDO 1.0. Konkrétne, špecifikácia komunikácie s FIDO klientom sa nachádza v [10], v kapitole *6. Android Intent API* a komunikácia s FIDO ASM sa nachádza v [11], v kapitole *5.1 Android ASM Intent API*.

Implementácia *UwpUaf Client API* a *UwpUaf ASM API* využíva na komunikáciu medzi aplikáciami prostriedky, ktoré sú uvedené v predchádzajúcej kapitole 6.2. Niektoré príkazy komunikačného protokolu sa spracovávajú bez zobrazenia užívateľského rozhrania[15]. Takéto príkazy sú spracovávané na pozadí, a na komunikáciu sa využíva *Application Service – Služba aplikácie*, prostriedok komunikácie popísaný v kapitole 6.2.6. Pre ostatné príkazy komunikačného protokolu, ktoré vyžadujú zobrazenie užívateľského rozhranie je použitý prostriedok komunikácie *Spustenie URI protokolu a čakanie na výsledok*, popísaný v kapitole 6.2.3.

Je možné zistiť informácie o aplikácii, ktorá má podporu pre spracovanie URI protokolu[16], ale nedokážeme zistiť informácie o aplikácii, ktorá poskytuje *Application Service* hľadaného protokolu. Keďže komponenty *UwpUaf Client API* a *UwpUaf ASM API* vyžadujú implementáciu oboch prostriedkov komunikácie, tak sa v našom riešení schéma URI protokolu a meno protokolu služby aplikácie zhodujú, a tak v prípade, ak zistíme, že aplikácia spracováva URI protokol, tak sa budeme spoliehať na to, že bude spracovávať protokol aplikačnej služby rovnakého mena. Táto informácia je dôležitá pri integrácii API do aplikácie, do ktorej sú integrované, pretože je nutné tieto hodnoty nastaviť v manifeste aplikácie[17].

UwpUaf Client API a *UwpUaf ASM API* sú implementované ako knižnice, ktoré sa integrujú do aplikácii *UwpUaf klient* a *UwpUaf ASM* a abstrahujú komunikáciu medzi aplikáciami pomocou spomenutých UWP prostriedkov na komunikáciu. Komponenty poskytujú API na

15. Napr. príkaz `DISCOVER`, ktorý odhaľuje dostupné autentizátory, a je spracovávaný FIDO klientom alebo `GetInfo` požiadavok, ktorý spracováva FIDO ASM, a ktorý vracia informácie o autentizátoroch daného ASM.
16. Pozri 6.2.5.
17. Príklad nastavenia manifest súboru môžeme nájsť v demo aplikácii *UwpUaf klient*

spracovanie komunikácie[18], aj na vyvolanie komunikácie[19], čím zjednodušujú implementáciu komunikácie medzi komponentami nášho riešenia *UwpUaf*. Zároveň je ich možné využiť pri implementácii ďalších komponent.

6.4 UwpUaf demo aplikácia

Aplikácia *UwpUaf demo* demonštruje volanie *UwpUaf Client demo* aplikácie. Demo aplikácia komunikuje s *UwpUaf Client demo* aplikáciou pomocou *UwpUaf Client API* a skrz API predáva správy UAF protokolu. Po spustení je zobrazený rozcestník, ktorý užívateľa naviguje na stránky aplikácie, ktoré demonštrujú spracovanie správ UAF protokolu. Hlavná stránka aplikácie s rozcestníkom je zobrazená na obrázku 6.3 a má nasledujúce položky:

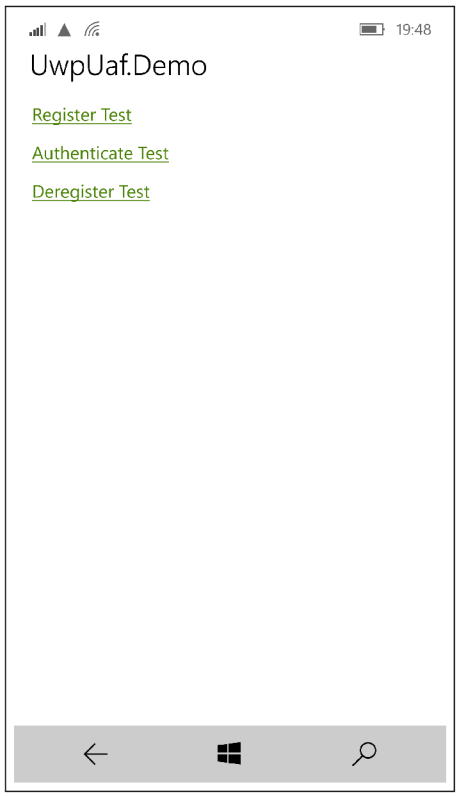
- Register Test – demonštruje spracovanie požiadavku na registráciu autentizátora, pozri obrázok 6.4.
- Authenticate Test – demonštruje spracovanie požiadavku na overenie užívateľa, pozri obrázok 6.5.
- Deregister Test – demonštruje spracovanie požiadavku na zrušenie registrovaného autentizátora, pozri obrázok 6.6.

Výpis 6.1: Nastavenie PackageFamilyName FIDO klienta

```
1 ClientApiSettings.UwpUafClientPackageFamilyName = "8453f20d-8176-488e-
      b710-2a95626b46b4_6m15y1xak46te";
```

Ako už bolo spomenuté, aplikácia komunikuje s *UwpUaf Client demo* aplikáciou pomocou *UwpUaf Client API*. Podľa špecifikácie pre implementáciu FIDO klienta pre operačný systém Android, by mal byť v systéme nainštalovaný iba jeden FIDO klient. V Android systéme, to

18. Spracovanie aktivácie URI protokolu a spracovanie požiadavku na službu aplikácie.
19. Spustenie URI protokolu a vytvorenie požiadavku na službu aplikácie.



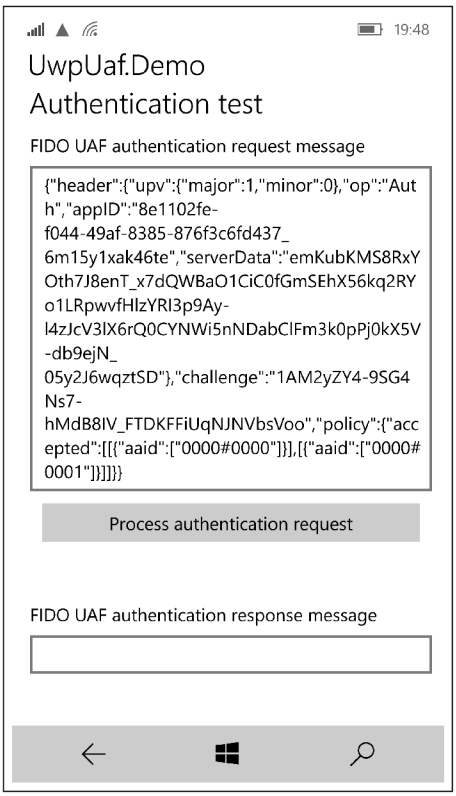
Obr. 6.3: UwpUaf demo - Hlavná stránka aplikácie

Obr. 6.4: UwpUaf demo - Demonštrácia registrácie

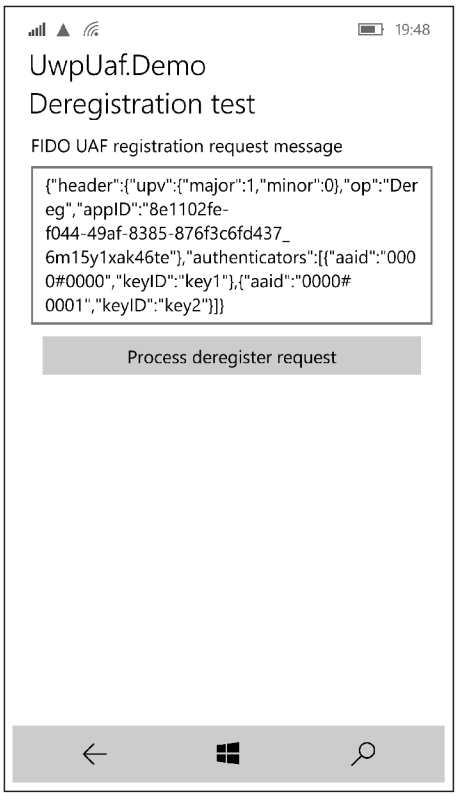
zabezpečíme definovaním `permission`[20] a `uses-permission` v manifeste aplikácie. Pokus o inštaláciu inej aplikácie, ktorá definuje rovnakú `permission`, bude zamietnutý [10]. Takýto mechanizmus v UWP neexistuje, a tak na to, aby aplikácia komunikovala iba s jedným FIDO klientom je nutné nastaviť jeho `PackageFamilyName` vybraného klienta. Ukážku nastavenia môžeme vidieť vo výpise 6.1.

Po spracovaní požiadavkou FIDO protokolu aplikáciou *UwpUaf Client demo*, je výsledok vrátený aplikácii *UwpUaf demo*, ktorá správu

20. Pre viac informácii pozri `https://developer.android.com/guide/topics/manifest/manifest-intro.html`



Obr. 6.5: UwpUaf demo - Demonštrácia overenia užívateľa



Obr. 6.6: UwpUaf demo - Demonštrácia zrušenia registrácie

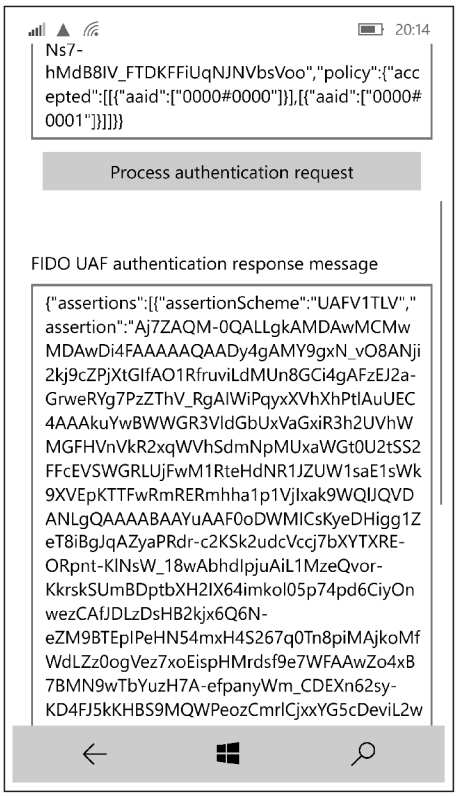
odpovede zobrazí. Zobrazený výsledok spracovanie môžeme vidieť na obrázkoch 6.7, 6.8 a 6.9[21].

6.5 UwpUaf klient demo

Aplikácia je demonštratívna implementácia FIDO klienta na platforme UWP, ktorá integruje *UwpUaf Client API*, a tak umožňuje iným aplikáciam komunikovať s ňou, ako napr. UwpUaf demo aplikácia. Spracováva správy FIDO UAF protokolu a ďalej komunikuje s FIDO

21. V prípade zrušenia registrácie sa neposiela späť žiadna správa, iba chybový kód v prípade, že sa pri spracovaní požiadavku vyskytla nejaká chyba.

Obr. 6.7: UwpUaf demo - Zobrazenie odpovede spracovania registrácie



Obr. 6.8: UwpUaf demo - Zobrazenie odpovede spracovania overenia užívateľa

ASM komponentami skrz *UwpUaf ASM API*. Komunikuje s viacerými ASM komponentami, z ktorých každá môže pracovať s viac ako jedným autentizátorom, takže užívateľovi je zobrazený zoznam autentizátorov, ktoré sú dostupné v systéme a užívateľ si môže zvoliť, ktorý autentizátor použije.

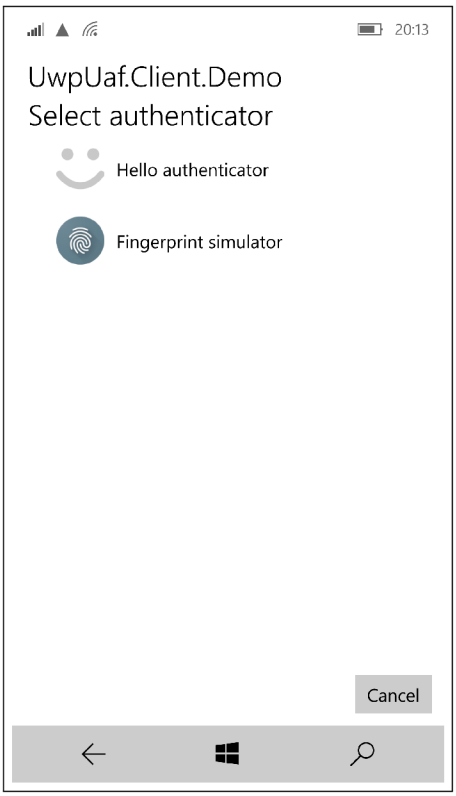
Konkrétnejšie, pri spracovaní požiadavku na registráciu sú užívateľovi ponúknuté autentizátory, ktoré si predtým neregistroval. V prípade overenia identity užívateľa sú užívateľovi zobrazené autentizátory, ktoré si naopak predtým registroval. Výber autentizátora môžeme vidieť na obrázku 6.10. Po výbere autentizátora je užívateľ navigovaný na zvolený autentizátor (ASM), kde pokračuje spracovanie požiadavku.

Obr. 6.9: UwpUaf demo - Zobrazenie spracovania zrušenia registrácie

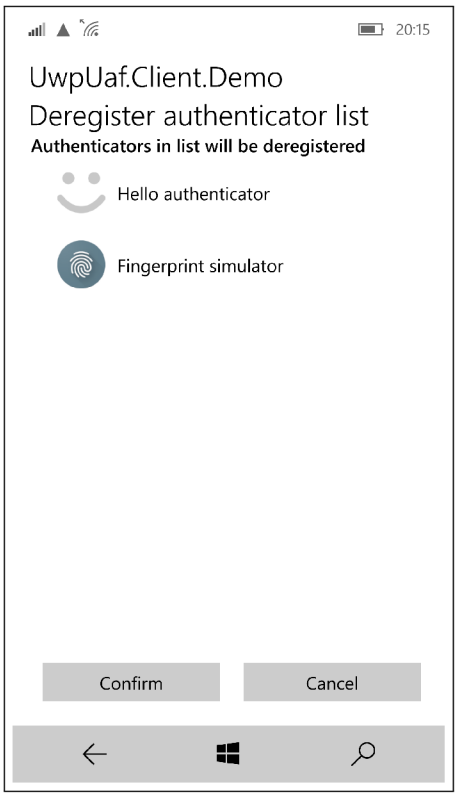
Na obrázku 6.11 vidíme potvrdzovaciu stránku na zrušenie registrácie u zobrazených autentizátoroch, keďže môžeme naraz zrušiť registráciu u viacerých autentizátoroch.

6.6 UwpUaf ASM demo

Aplikácia je demonštratívnou implementáciou FIDO ASM komponenty na platforme UWP, ktorá integruje *UwpUaf ASM API* pre komunikáciu s *UwpUaf Client demo* aplikáciou. Aplikácia poskytuje dva autentizátory:



Obr. 6.10: UwpUaf Client demo - Výber autentizátora

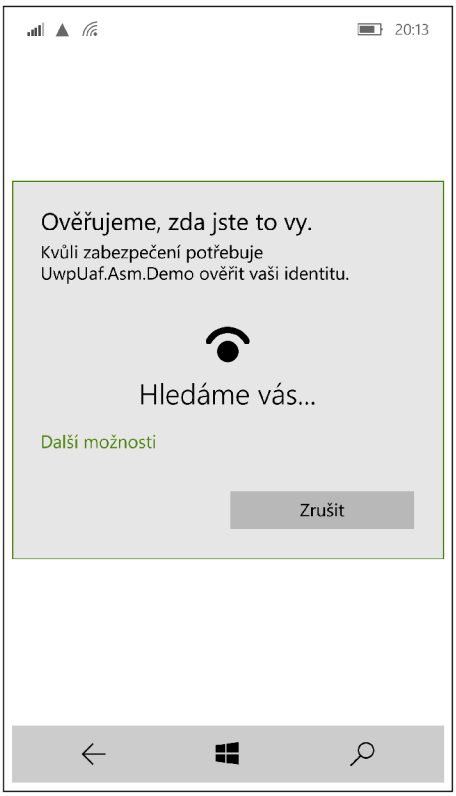


Obr. 6.11: UwpUaf Client demo - Zrušenie registrácie

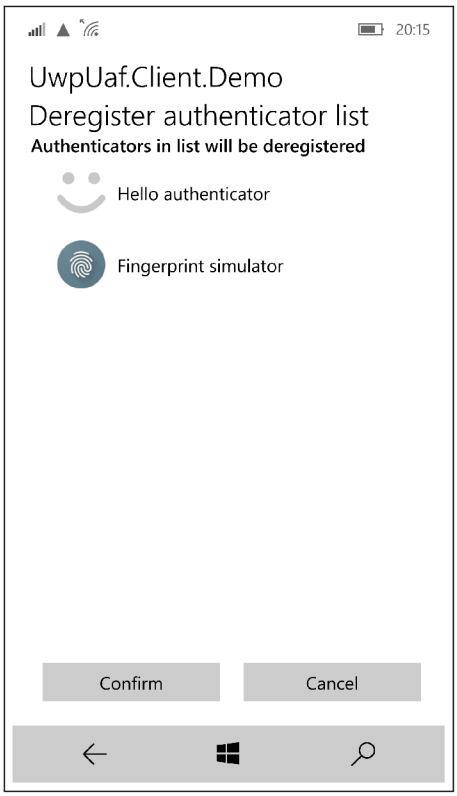
- Hello autentizátor
- Fingerprint simulátor

6.6.1 Hello autentizátor

Je implementácia autentizátora, ktorá využíva na overenie identity užívateľa technológie *Microsoft Passport* a *Windows Hello*, ktoré sme popísali v kapitole 5. Na testovanom mobilnom zariadení Microsoft Lumia 950 bolo dostupné overenie identity užívateľa pomocou biometriky očnej dúhovky. Windows Hello automaticky spúšťa overenie identity užívateľa pomocou očnej dúhovky, hneď po spustení procesu overenia, čo môžeme vidieť na obrázku 6.12. Po úspešnom overení



Obr. 6.12: UwpUaf ASM demo - Overenie identity užívateľa pomocou biometriky očnej dúhovky

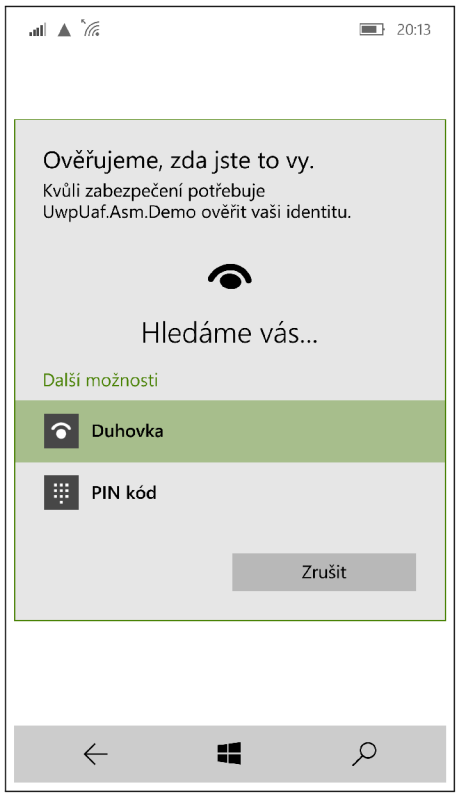


Obr. 6.13: UwpUaf ASM demo - Úspešné overenie identity užívateľa pomocou biometriky očnej dúhovky

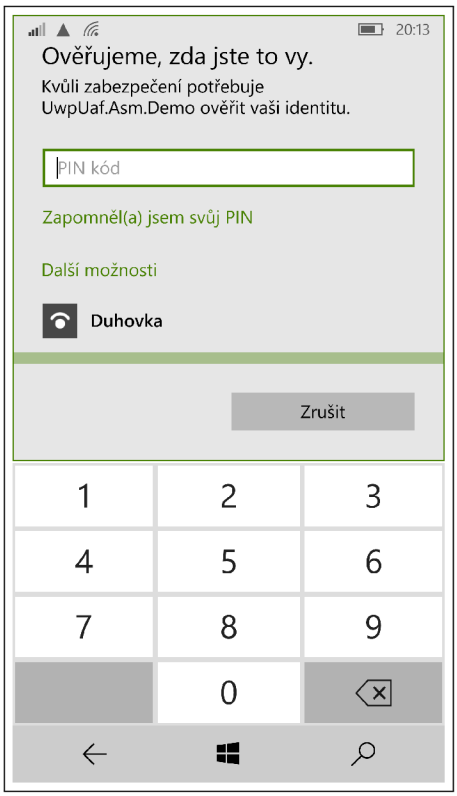
pomocou biometriky je nutné potvrdiť overenie, čo zobrazuje obrázok 6.13.

Okrem overenia pomocou biometriky očnej dúhovky ponúka Windows Hello ďalšie možnosti overenia, ako môžeme vidieť na obrázku 6.14. Windows Hello ponúka overenie pomocou zadania PINu, čo zobrazuje obrázok 6.15. Pri procese registrácie i procese overenia užívateľa je zobrazené rovnaké užívateľské systémové rozhranie. Pri zrušení registrácie je zobrazená potvrdzovacia stránka, ako môžeme vidieť na obrázku 6.16, táto stránka však nie je systémová.

Jedným z cieľov práce je implementácia FIDO UAF autentizátora, ktorý autentizuje užívateľa pomocou Windows Hello. FIDO autenti-



Obr. 6.14: UwpUaf ASM demo - Možnosti overenia vo Windows Hello

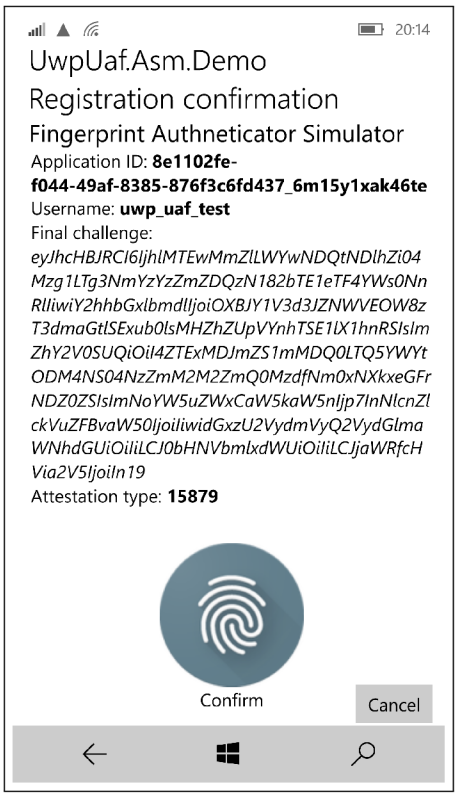


Obr. 6.15: UwpUaf ASM demo - Overenie identity užívateľa pomocou PINu

zátor vytvára štruktúru, ktorá je špecifikovaná v [14] a takisto definuje podporované algoritmy, ktorými je možné podpísať výzvu obsiahnutú v požiadavku. Algoritmus, ktorý na vytvorenie digitálneho podpisu používa Microsoft Passport[22], sa nenachádza medzi podporovanými algoritmami v špecifikácii FIDO 1.0 a podpisovací algoritmus pre Microsoft Passport nie je možné zmeniť. Preto nie je možné využiť

22. Algoritmus popísaný v `https://msdn.microsoft.com/en-us/library/windows/apps/windows.security.credentials.keycredential.aspx`. Implementáciu overenia podpisu nájdeme v `https://mtaulty.com/2016/01/04/windows-hello-microsoft-passport-and-the-keycredentialmanager-api/`.

Obr. 6.16: UwpUaf ASM demo - Zrušenie registrácie Windows Hello



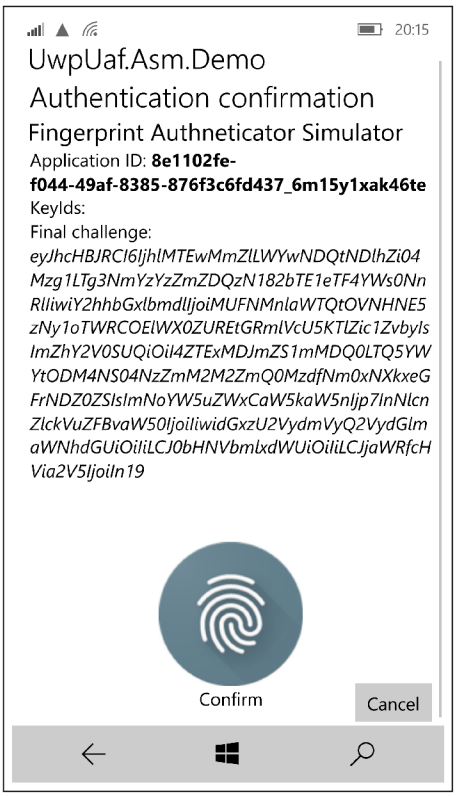
Obr. 6.17: UwpUaf ASM demo - Fingerprint autentizátor simulátor, Registrácia

Microsoft Passport spolu s Windows Hello ako FIDO autentizátor pre špecifikáciu FIDO 1.0[23].

6.6.2 Fingerprint simulátor

Je implementácia druhého dostupného autentizátora. Je to simulátor autentizácie užívateľa pomocou otlačku prsta. Dôvod implementácie spočíva v nemožnosti využiť *Hello autentizátor* ako FIDO autentizá-

23. Podpora pre algoritmus, ktorý využíva Microsoft Passport sa nachádza až vo verzii špecifikácie FIDO 2.0 (`https://fidoalliance.org/specs/fido-v2.0-ps-20150904/fido-key-attestation-v2.0-ps-20150904.html`), na ktorej vývoji Microsoft spolupracuje [28].



Obr. 6.18: UwpUaf ASM demo - Fingerprint autentizátor simulátor, Autentifikácia

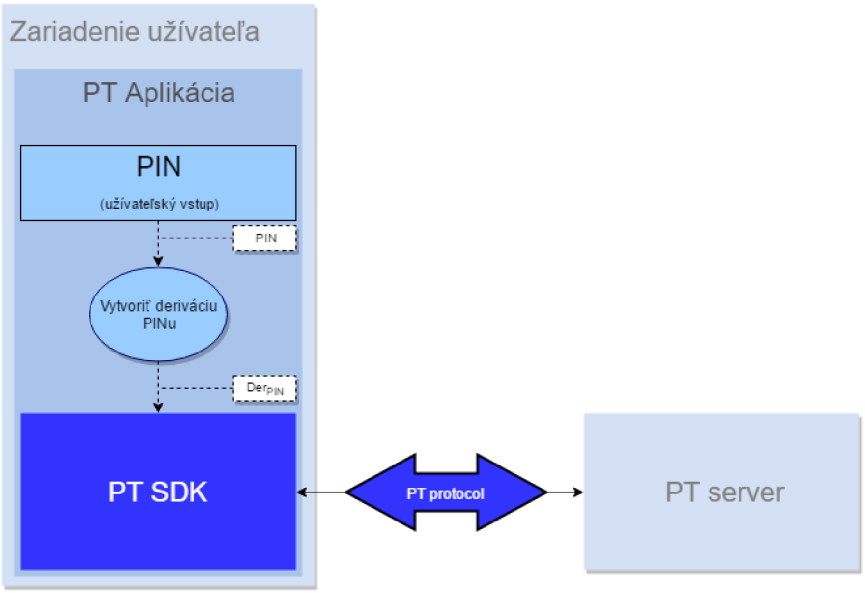
Obr. 6.19: UwpUaf ASM demo - Zrušenie registrácie Fingerprint autentizátor simulátora

tor, kvôli nepodporovanému algoritmu na vytvorenie podpisu. Tým pádom by nebolo možne správne overiť výslednú UAF správu.

Užívateľské rozhranie autentizátora je zobrazené na obrázkoch 6.17, 6.18 a 6.19. Všetky tri stránky autentizátora čakajú na potvrdenie užívateľa, aby autentizátor mohol ďalej pokračovať v spracovaní požiadavku.

6.7 Integrácia UwpUaf riešenie do existujúcej aplikácie

Jedným z cieľov práce je integrácia vyvinutého riešenia do existujúcej aplikácie, ktorá rieši bezpečné potvrdzovanie transakcii. Pre túto aplikáciu budeme v tejto kapitole používať skratku *PT*, respektíve *PT aplikácia*. V PT aplikácii je užívateľ autentifikovaný zadaním PINu. Cieľom je nahradiť autentifikáciu užívateľa PINom na autentifikáciu pomocou nášho riešenia *UwpUaf* s využitím Windows Hello.



Obr. 6.20: Zjednodušená architektúra PT aplikácie

Najskôr sa však pozrieme na zjednodušenú architektúru PT aplikácie, do ktorej sa pokúsime naše riešenie integrovať. Zjednodušený pohľad na architektúru aplikácie je zobrazený na obrázku 6.20. PT aplikácia primárne pracuje s PINom, ktorý zadáva užívateľ ako vstup, pri každom požiadavku na autentifikáciu užívateľa. PIN je v aplikácii transformovaný jednosmernou funkciou na derivát PINu (Der_{PIN}), ktorý vstupuje do PT SDK komponenty. PT SDK následne pracuje s Der_{PIN} tak, že z neho odvodí kľúč pre symetrickú šifru, ktorou je v zariadení užívateľa zašifrované tajomstvo, ktoré je zdielané medzi

PT aplikáciou a PT serverom. Následne s využitím znalosti tajomstva sa aplikácia dokáže autentifikovať voči PT serveru, čím dôjde k potvrdeniu transakcie.

Predchádzajúci odstavec popisuje PT aplikáciu veľmi zjednodušene a bez akýchkoľvek implementačných detailov. Pre túto prácu je však tento zjednodušený pohľad dostačujúci. Pre prácu je dôležitý fakt, že PT SDK ako vstup pri autentifikácii užívateľa berie Der_{PIN}, ktorý musí mať vždy rovnakú hodnotu, ak chceme aby autentifikácia voči PT serveru prebehla úspešne. Je dôležitá aj jednosmernosť funkcie, ktorá transformuje PIN na Der_{PIN}, takže nedokážeme z Der_{PIN} získať pôvodný PIN. PT SDK má na všetkých mobilných platformách (Android, iOS a Windows Phone) rovnakú architektúru.

Základný predpoklad pre spracovanie FIDO UAF protokolu je, že vyžaduje FIDO UAF server, ktorý na základe overenia správy, odpovede, autentifikuje užívateľa alebo nie. Táto komponenta v PT serveri chýba a jej implementácia je nad rámec tejto práce. Tým pádom bola uvažovaná možnosť simulovať funkciu FIDO UAF servera v PT aplikácii, prípadne v PT SDK komponente, čiže v zariadení užívateľa. Obidve možnosti boli zamietnuté kvôli rovnakému dôvodu, a to, že autentifikácia pomocou FIDO UAF protokolu nám dokáže odpovedať iba *áno*, užívateľ bol autentifikovaný, alebo *nie*, užívateľ bol autentifikovaný a to na základe overenia FIDO UAF odpovede. To znamená, že nedokážeme odvodiť správny Der_{PIN}, z ktorého dostaneme kľúč, alebo samotný kľúč, ktorým môžeme dešifrovať tajomstvo PT aplikácie v PT SDK komponente. A ako sme sa dozvedeli v kapitole 6.6, tak ďalším dôvodom, prečo nie je možné využiť UwpUaf s Hello autentizátorom je, že vytvorená odpoveď by nebola správna, kvôli nepodporovanému podpisovaciemu algoritmu.

Ďalšou uvažovanou možnosťou bolo využitie technológii Windows Hello a Microsoft Passport a ich integrácia bez UwpUaf a tak bez FIDO protokolu. Najskôr sa však pozrime na ostatné mobilné platformy. PT aplikácie na mobilných platformách Android a iOS implementujú nahradenie PINu pomocou biometriky otlačkov prstov. Každá platforma má svoj špecifický biometrický systém. Pozrieme sa teda bližšie na to, ako je implementovaná autentifikácia pomocou biometrík na týchto platformách.

Mobilná platforma iOS používa na biometrickú autentifikáciu systém *Touch ID*[24]. Touch ID poskytuje API na lokálnu autentifikáciu užívateľa v zariadení pomocou biometriky otlačkov prstov. Spolu s Touch ID sa používa *KeyChain*[25] služba, ktorá poskytuje API pre bezpečné uloženie citlivých údajov v zariadení užívateľa. Pre povolenie prístupu ku KeyChain službe sa dá použiť Touch ID[26]. V PT aplikácii pre iOS to funguje podľa tohoto prístupu. Do KeyChain sa uloží Der_{PIN}, ktorý je po úspešnej biometrickej autentifikácii pomocou Touch ID sprístupnený a predaný do PT SDK komponenty na ďalšie spracovanie.

V PT aplikácii pre mobilnú platformu Android funguje biometrická autentizácia veľmi podobne ako na platforme iOS. Prácu so snímačom otlačkov prstov sa stará trieda `FingerprintManager`[27]. Android nemá bezpečné úložisko citlivých údajov, ale má bezpečné úložisko pre uloženie kryptografických kľúčov[28], ktoré predstavuje trieda `KeyStore`[29]. Prístup ku kryptografickému kľúču je udelený podľa výsledku biometrickej autentifikácie, o ktorú sa stará *FingerprintManager*. Der_{PIN} je zašifrovaný symetrickým kľúčom, ktorý je uložený do *KeyStore*. Pri autentifikácii je naopak z *KeyStore* vytiahnutý a použitý na dešifrovanie zašifrovaného Der_{PIN}, ktorý je po dešifrovaný predaný do PT SDK komponenty na ďalšie spracovanie.

Dôležité je, že pri oboch platformách je Der_{PIN} chránený v zariadení, či už bezpečným uložením v KeyChain alebo uložením v zašifrovanej podobe, pričom je bezpečne uloží šifrovací kľúč v KeyStore.

Rozhranie Microsoft Passport API, ktoré sme si popísali v kapitole 5.2.2, obsahuje iba jednu kryptografickú operáciu a tou je vytvorenie digitálne podpisu. Podpis vytvorený Microsoft Passport API, spolu s verejným kľúčom, mi povie iba či je vytvorený podpis správny alebo

---

24. Dokumentácia `https://developer.apple.com/reference/localauthentication`.
25. Dokumentácia `https://developer.apple.com/library/content/documentation/Security/Conceptual/keychainServConcepts/iPhoneTasks/iPhoneTasks.html#//apple_ref/doc/uid/TP30000897-CH208-SW1`.
26. Pozri `https://developer.apple.com/videos/play/wwdc2014/711/`.
27. Dokumentácia `https://developer.android.com/reference/android/hardware/fingerprint/FingerprintManager.html`.
28. Symetrických aj asymetrických.
29. Dokumentácia `https://developer.android.com/training/articles/keystore.html`.

nie[30]. Na základe týchto dvoch informácii nie je možné dopracovať k Der_{PIN}, ktorý autentifikuje užívateľa voči PT serveru. Preto je nutné usúdiť, že biometrickú autentifikáciu integrovanú do Windows 10, teda technológie Microsoft Passport a Windows Hello, nie je možné integrovať do verzie PT aplikácie, ktorá mi bola poskytnutá v čase písanie tejto práce.

30. Teda či verejný kľúč patrí k súkromnému kľúču, ktorý vytvoril podpis.

7 Záver

Cieľom práce bolo najskôr popísať špecifikácie FIDO Alliance s dôrazom na špecifikáciu Universal Authentication Framework (UAF). Podľa tejto špecifikácie, vo verzii 1.0, implementovať FIDO klienta pre operačný systém Windows 10 Mobile. Ďalej podľa špecifikácia implementovať autentizátor, ktorý na autentifikáciu užívateľa bude využívať biometrický systém Windows Hello, ktorý mal byť následne integrovaný do existujúcej aplikácie, ako náhrada za zadávanie PINu, pri procese overenia identity užívateľa.

Kapitoly 3 a 4 sa venujú predstaveniu FIDO aliancie a popisu špecifikácii vyvinutých alianciou. Dôraz sa kladie na špecifikáciu protokolu Universal Authentication Framework, ktorá má za cieľ nahradiť autetifikáciu užívateľov pomocou hesla za autentifikáciu pomocou biometrických autentizátorov. Práca detailnejšie popisuje architektúru UAF, ako aj prípady použitia UAF. Čitateľovi poskytujú prehľad o FIDO aliancii a špecifikáciách aliancie, ako aj popis UAF protokolu.

Ďalšia kapitola 5 sa venuje biometrickej autentifikácii užívateľov integrovanej do operačného systému Windows 10, za čím sa skrývajú technológie Microsoft Passport a Windows Hello. Kapitola poskytuje čitateľovi prehľad o možnostiach týchto technológii, ako aj o obmedzeniach, ktoré sú uvedené ďalej v texte. Takisto je čitateľovi predstavená spojitosť so špecifikáciou UAF.

Praktická časť, ktorej je venovaná kapitola 6, sa venuje implementácii FIDO klienta a FIDO autentizátora podľa špecifikácie FIDO 1.0 UAF pre platformu UWP operačného systému Windows 10. Predstavuje jednotlivé aplikácie nami implementovaného riešenia UWP UAF, ako aj návrh a implementáciu komunikačných rozhraní, pomocou ktorých aplikácie komunikujú medzi sebou. Čitateľovi poskytuje ukážkové riešenie implementácie UAF komponent pre platformu UWP. Okrem toho poskytuje detailný popis možností komunikácie medzi aplikáciami pre platformu UWP.

Niektoré definované ciele práce však neboli splnené. Jedná sa o implementáciu FIDO autentizátora s využitím biometrickej autentifikácie integrovanej do Windows 10 a integráciu autentizátora do existujúcej aplikácie. O dôvodoch, prečo uvedené ciele neboli splnené sa čitateľ dozvie na konci praktickej časti práce.

Za najväčší prínos práce považujem samotnú implementáciu podpory FIDO 1.0 UAF protokolu na platforme UWP, keďže v čase písania práce neexistovalo žiadne podobné riešenie. Zdrojové kódy boli publikované pre verejnosti, a tak je možné, aby bolo riešenie využité inými vývojármi pri implementácii vlastných UAF komponent.

Literatúra

[1] Ijlal Loutfi and Audun Jøsang. „1,2, Pause: Lets Start by Meaningfully Navigating the Current Online Authentication Solutions Space“. English. In: *Trust Management IX*. Ed. by Christian Damsgaard Jensen et al. Vol. 454. IFIP Advances in Information and Communication Technology. Springer International Publishing, 2015, pp. 165–176. ISBN: 978-3-319-18490-6. DOI: 10.1007/978-3-319-18491-3_12. URL: http://dx.doi.org/10.1007/978-3-319-18491-3_12.

[2] HyunJin Kim et al. „Mobile and Wireless Technology 2015“. In: ed. by J. Kuinam Kim and Naruemon Wattanapongsakorn. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015. Chap. Biometric Authentication Technology Trends in Smart Device Environment, pp. 199–206. ISBN: 978-3-662-47669-7. DOI: 10.1007/978-3-662-47669-7_23. URL: http://dx.doi.org/10.1007/978-3-662-47669-7_23.

[3] Rolf Lindemann. „The Evolution of Authentication“. English. In: *ISSE 2013 Securing Electronic Business Processes*. Ed. by Helmut Reimer, Norbert Pohlmann, and Wolfgang Schneider. Springer Fachmedien Wiesbaden, 2013, pp. 11–19. ISBN: 978-3-658-03370-5. DOI: 10.1007/978-3-658-03371-2_2. URL: http://dx.doi.org/10.1007/978-3-658-03371-2_2.

[4] *Microsoft Passport guide*. https://technet.microsoft.com/itpro/windows/keep-secure/microsoft-passport-guide. 2016.

[5] Kim Nguyen. „Authentification and identification — Taking the user into account“. In: *Datenschutz und Datensicherheit - DuD* 38.7 (2014), pp. 467–469. ISSN: 1862-2607. DOI: 10.1007/s11623-014-0208-4. URL: http://dx.doi.org/10.1007/s11623-014-0208-4.

[6] FIDO Alliance. *About The FIDO Alliance*. https://fidoalliance.org/about/overview/. 2014.

[7] Malte Kahrs and Kim Nguyen. „ISSE 2015: Highlights of the Information Security Solutions Europe 2015 Conference“. In: ed. by Helmut Reimer, Norbert Pohlmann, and Wolfgang Schneider. Wiesbaden: Springer Fachmedien Wiesbaden, 2015. Chap. Fu-

ture Ecosystems for Secure Authentication and Identification, pp. 12–21. ISBN: 978-3-658-10934-9. DOI: 10.1007/978-3-658-10934-9_2. URL: http://dx.doi.org/10.1007/978-3-658-10934-9_2.

[8] FIDO Alliance. *The FIDO Alliance: December 2014 Whitepaper*. https://fidoalliance.org/wp-content/uploads/FIDOMessagingWPv1.pdf. 2014.

[9] FIDO Alliance. *Specifications Overview*. https://fidoalliance.org/specifications/overview/. 2014.

[10] FIDO Alliance. *FIDO UAF Application API and Transport Binding Specification v1.0*. https://fidoalliance.org/specs/fido-uaf-v1.0-ps-20141208/fido-uaf-client-api-transport-v1.0-ps-20141208.html. 2014.

[11] FIDO Alliance. *FIDO UAF Authenticator-Specific Module API*. https://fidoalliance.org/specs/fido-uaf-v1.0-ps-20141208/fido-uaf-asm-api-v1.0-ps-20141208.html. 2014.

[12] FIDO Alliance. *FIDO UAF High-Level Architecture*. https://fidoalliance.org/specs/fido-uaf-v1.0-ps-20141208/img/fido-uaf-architecture.png. 2014.

[13] Ijlal Loutfi and Audun Jøsang. „Secure IT Systems: 20th Nordic Conference, NordSec 2015, Stockholm, Sweden, October 19–21, 2015, Proceedings“. In: ed. by Sonja Buchegger and Mads Dam. Cham: Springer International Publishing, 2015. Chap. FIDO Trust Requirements, pp. 139–155. ISBN: 978-3-319-26502-5. DOI: 10.1007/978-3-319-26502-5_10. URL: http://dx.doi.org/10.1007/978-3-319-26502-5_10.

[14] FIDO Alliance. *FIDO UAF Authenticator Commands v1.0*. https://fidoalliance.org/specs/fido-uaf-v1.0-ps-20141208/fido-uaf-authnr-cmds-v1.0-ps-20141208.html. 2014.

[15] FIDO Alliance. *FIDO UAF Architectural Overview*. https://fidoalliance.org/specs/fido-uaf-v1.0-ps-20141208/fido-uaf-overview-v1.0-ps-20141208.html. 2014.

[16] FIDO Alliance. *FIDO UAF Protocol Specification v1.0*. https://fidoalliance.org/specs/fido-uaf-v1.0-ps-20141208/fido-uaf-protocol-v1.0-ps-20141208.html. 2014.

[17] FIDO Alliance. *Registration Message Flow*. https://fidoalliance.org/specs/fido-uaf-v1.0-ps-20141208/img/fido-uaf-overview-reg-flow.png. 2014.

[18] FIDO Alliance. *Authentication Message Flow*. https://fidoalliance.org/specs/fido-uaf-v1.0-ps-20141208/img/fido-uaf-overview-auth-flow.png. 2014.

[19] FIDO Alliance. *Confirmation Message Flow*. https://fidoalliance.org/specs/fido-uaf-v1.0-ps-20141208/img/fido-uaf-overview-transact-flow.png. 2014.

[20] FIDO Alliance. *Deregistration Message Flow*. https://fidoalliance.org/specs/fido-uaf-v1.0-ps-20141208/img/fido-uaf-overview-dereg-flow.png. 2014.

[21] Alexander Koren. *Microsoft Passport and Windows Hello*. https://msdn.microsoft.com/en-us/windows/uwp/security/microsoft-passport. 2016.

[22] *Convenient two-factor authentication with Microsoft Passport and Windows Hello*. https://blogs.windows.com/buildingapps/2016/01/26/convenient-two-factor-authentication-with-microsoft-passport-and-windows-hello/. 2016.

[23] *Windows Hello biometric requirements*. https://msdn.microsoft.com/en-us/library/windows/hardware/mt587095%28v=vs.85%29.aspx.

[24] Říha Zdeněk Rak Roman Matyáš Vašek. *Biometrie a identita člověka ve forenzních a komerčních aplikacích*. Praha: Grada, 2008. ISBN: 9788024723655.

[25] Greg Duncan. *Windows Hello with the Kinect v2*. https://channel9.msdn.com/coding4fun/kinect/Windows-Hello-with-the-Kinect-v2. 2016.

[26] Shane McGlaun. *Tobii eye-tracking supports facial recognition with Windows Hello*. http://www.slashgear.com/tobii-eye-tracking-supports-facial-recognition-with-windows-hello-17418897/. 2016.

[27] Daniel Rubino. *Windows Hello – What's the difference between an iris scanner and 3D facial recognition?* http://www.windowscentral.com/windows-hello-iris-scanners-vs-facial-recognition. 2015.

[28] Dustin Ingalls. *Microsoft Announces FIDO Support Coming to Windows 10*. https://blogs.windows.com/buildingapps/2016/01/26/convenient-two-factor-authentication-with-microsoft-passport-and-windows-hello/. Feb. 2015.

[29] J. Decker. *Manage identity verification using Microsoft Passport*. https://technet.microsoft.com/en-us/itpro/windows/keep-secure/manage-identity-verification-using-microsoft-passport. 2016.

[30] FIDO Alliance. *UAF Application API Architecture and Transport Layers*. https://fidoalliance.org/specs/fido-uaf-v1.0-ps-20141208/img/fido-uaf-client-api-transport-layers.png. 2014.

[31] FIDO Alliance. *UAF ASM API Architecture*. https://fidoalliance.org/specs/fido-uaf-v1.0-ps-20141208/img/fido-uaf-asm-api-arch.png. 2014.

[32] Wikipedia. *Serial Peripheral Interface Bus — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Serial%20Peripheral%20Interface%20Bus&oldid=720036558. 2016.

[33] *Auto-launching apps using file and URI associations for Windows Phone 8*. https://msdn.microsoft.com/en-us/library/windows/apps/jj206987(v=vs.105).aspx. 2016.

[34] Wikipedia. *Uniform Resource Identifier — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/wiki/Uniform_Resource_Identifier. 2016.

[35] *Handle URI activation*. https://msdn.microsoft.com/en-us/windows/uwp/launch-resume/handle-uri-activation. 2016.

[36] Rajen Kishna. *Using cross-app communication to make apps work together*. https://blogs.windows.com/buildingapps/2015/09/22/using-cross-app-communication-to-make-apps-work-together-10-by-10/. 2015.

[37] Edward Moemeka and Elizabeth Moemeka. *Real world Windows 10 development*. Apress, 2015. ISBN: 978-1-4842-1449-7.

[38] Selva Ganapathy. *Inter App Communications in Windows 10 UWP*. http://www.c-sharpcorner.com/UploadFile/6f0898/inter-app-communications-in-windows-10-uwp535/. 2015.

[39] *App Services, Developer's Guide to Windows 10*. https://view.officeapps.live.com/op/view.aspx?src=http://files.channel9.msdn.com/thumbnail/83e22e1b-0d3e-48ed-8c40-e087527afd06.pptx. 2016.

[40] *Create and consume an app service*. https://msdn.microsoft.com/en-us/windows/uwp/launch-resume/how-to-create-and-consume-an-app-service. 2016.

A Možnosti komunikácie aplikácii vo Windows 10 – Ukážky kódov

Výpis A.1: Aktivácia URI protokolu alebo súboru

```
// Aktivácia/Spustenie protokolu
await Launcher.LaunchUriAsync(new Uri("org.fidoalliance.fido.uaf-
    client+json://"));

// Spustenie súboru
await Launcher.LaunchFileAsync(file);
```

Výpis A.2: Spracovanie URI protokolu

```
sealed partial class App : Application
{
    protected override void OnActivated(IActivatedEventArgs e)
    {
        if (e.Kind == ActivationKind.Protocol)
        {
            var args = e as ProtocolActivatedEventArgs;
            var uri = e.Uri;
            // ...
        }
        else if (e.Kind == ActivationKind.ProtocolForResults)
        {
            var args = e as ProtocolForResultsActivatedEventArgs;
            var uri = e.Uri;
            // ...
            if (args.Data.ContainsKey("request"))
            {
                var response = new ValueSet();
                response.Add("response", "response");

                args.ProtocolForResultsOperation.ReportCompleted(
                    response)
            }
        }

        base.OnActivated(e);
    }
}
```

Výpis A.3: Príklad manifest súboru aplikácie, ktorá spracováva URI protokol

```
<Applications>
  <Application Id="App"
```

```
3       Executable="$targetnametoken$.exe"
4       EntryPoint="UriProtocolHandler.App">
5       ...
6       <Extensions>
7         <uap:Extension Category="windows.protocol">
8           <uap:Protocol Name="org.fidoalliance.fido.uaf-client+json">
9             <uap:Logo>images\icon.png</uap:Logo>
10            <uap:DisplayName>FIDO UAF Client</uap:DisplayName>
11          </uap:Protocol>
12        </uap:Extension>
13      </Extensions>
14      ...
15    </Application>
16  </Applications>
```

Výpis A.4: Aktivácia URI protokolu

```
1  var options = new LauncherOptions();
2  options.TargerApplicationPackageFamilyName = "<package family>";
3
4  ValueSet data = new ValueSet();
5  data.Add("request", "request");
6
7  var uri = new Uri("org.fidoalliance.fido.uaf-client+json://");
8
9  var result = await Launcher.LaunchUriForResultsAsync(uri, options,
       data);
10 var response = result.Result?["response"] as string;
```

Výpis A.5: Zistenie podpory spracovania URI protokolu v systéme

```
1  // Launcher.QueryUriSupportAsync
2  var uri = new Uri("org.fidoalliance.fido.uaf-client+json://");
3  var packageFamilyName = "<package family>";
4  var launchQuerySupportStatus = await Launcher.QueryUriSupportAsync(uri
       , LaunchQuerySupportType.UriForResults, packageFamilyName);
5
6  // Launcher.FindUriSchemeHandlersAsync
7  var uriScheme = "org.fidoalliance.fido.uaf-client+json";
8  var appInfos = await Launcher.FindUriSchemeHandlersAsync(uriScheme,
       LaunchQuerySupportType.UriForResults);
```

Výpis A.6: Služba aplikácie

```
1  using Windows.ApplicationModel.AppService;
2  using Windows.ApplicationModel.Background;
3  using Windows.Foundation.Collections;
4
5  public sealed class MyAppService : IBackgroundTask
6  {
7    private BackgroundTaskDeferral backgroundTaskDeferral;
```

```
8     private AppServiceConnection appServiceconnection;
9
10    public void Run(IBackgroundTaskInstance taskInstance)
11    {
12      // Informuje systém, že úloha na pozadí sa spracováva a úloha
            nemá byť ukončená
13      this.backgroundTaskDeferral = taskInstance.GetDeferral();
14      // Associate a cancellation handler with the background task.
15      taskInstance.Canceled += OnTaskCanceled;
16
17      // Získanie spojenia na službu aplikácie a nastavenie listenerov
            na požiadavky na službu aplikácie
18      var details = taskInstance.TriggerDetails as
            AppServiceTriggerDetails;
19      appServiceconnection = details.AppServiceConnection;
20      appServiceconnection.RequestReceived += OnRequestReceived;
21    }
22
23    private async void OnRequestReceived(AppServiceConnection sender,
          AppServiceRequestReceivedEventArgs args)
24    {
25      // Informuje systém, že udalosť je spracovávaná
26      var messageDeferral = args.GetDeferral();
27      // Údaje požiadavku, ktoré zaslala klientská aplikácia, volajúca
            službu
28      ValueSet request = args.Request.Message;
29
30      // Spracovanie požiadavku
31      // ...
32
33      // Odpoveď
34      ValueSet response = new ValueSet();
35      // Údaje odpovedi
36      response.Add("key", "value");
37
38      // Zaslanie odpovedi späť klientskej aplikácii
39      await args.Request.SendResponseAsync(response);
40
41      messageDeferral.Complete();
42    }
43
44    private void OnTaskCanceled(IBackgroundTaskInstance sender,
          BackgroundTaskCancellationReason reason)
45    {
46      if (this.backgroundTaskDeferral != null)
47      {
48        // Informuje systém o dokončení spracovania úlohy
49        this.backgroundTaskDeferral.Complete();
50      }
51    }
52  }
```

Výpis A.7: Príklad manifest súboru s poskytnutím služby aplikáciou

```
1   <Applications>
```

```
2     <Application Id="App"
3       Executable="$targetnametoken$.exe"
4       EntryPoint="AppServiceProvider.App">
5       ...
6       <Extensions>
7         <uap:Extension Category="windows.appService" EntryPoint="
              MyAppService.MyAppService">
8           <uap:AppService Name="org.fidoalliance.fido.uaf-client+json"/>
9         </uap:Extension>
10      </Extensions>
11      ...
12    </Application>
13  </Applications>
```

Výpis A.8: Volanie služby aplikácie

```
1  async private void CallAppService(object sender, RoutedEventArgs e) {
2      AppServiceConnection connection = new AppServiceConnection();
3
4      // Protokol
5      connection.AppServiceName = "org.fidoalliance.fido.uaf-client+json
           ";
6      // Meno aplikácie
7      connection.PackageFamilyName = "<package family>";
8
9      // Otvorenie komunikačného kanálu
10     AppServiceConnectionStatus status = await connection.OpenAsync();
11     if (status == AppServiceConnectionStatus.Success)
12     {
13         // Údaje požiadavku
14         var message = new ValueSet();
15         message.Add("UAFIntentType", "DISCOVER");
16
17         // Zasiela požiadavok na službu a čaká na odpoveď
18         AppServiceResponse response = await connection.
               SendMessageAsync(message);
19         if (response.Status == AppServiceResponseStatus.Success)
20         {
21             await new MessageDialog("sent").ShowAsync();
22         }
23         else
24             await new MessageDialog("Failed when sending message to
                   app").ShowAsync();
25     }
26     else
27     {
28         await new MessageDialog(status.ToString()).ShowAsync();
29     }
30     connection.Dispose();
31 }
```

B Obsah elektronickej prílohy

- `uwpUaf.zip` – Archív obsahujúci zdrojové kódý implementovaného riešenie UwpUaf.