

**SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY**

Evidenčné číslo: FEI-5382-91764

**PASSWORD MANAGER
BAKALÁRSKA PRÁCA**

2020

Peter Čuřík

**SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY**

Evidenčné číslo: FEI-5382-91764

**PASSWORD MANAGER
BAKALÁRSKA PRÁCA**

Študijný program: aplikovaná informatika
Názov študijného odboru: informatika
Školiace pracovisko: Ústav informatiky a matematiky
Vedúci záverečnej práce: prof. Ing. Pavol Zajac, PhD.
Konzultant: unknown

Bratislava 2020

Peter Čuřík

SÚHRN

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Študijný program:	aplikovaná informatika
Autor:	Peter Čuřík
Bakalárská práca:	Password manager
Vedúci záverečnej práce:	prof. Ing. Pavol Zajac, PhD.
Konzultant:	unknown
Miesto a rok predloženia práce:	Bratislava 2020

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean et est a dui semper facilisis. Pellentesque placerat elit a nunc. Nullam tortor odio, rutrum quis, egestas ut, posuere sed, felis. Vestibulum placerat feugiat nisl. Suspendisse lacinia, odio non feugiat vestibulum, sem erat blandit metus, ac nonummy magna odio pharetra felis. Vivamus vehicula velit non metus faucibus auctor. Nam sed augue. Donec orci. Cras eget diam et dolor dapibus sollicitudin. In lacinia, tellus vitae laoreet ultrices, lectus ligula dictum dui, eget condimentum velit dui vitae ante. Nulla nonummy augue nec pede. Pellentesque ut nulla. Donec at libero. Pellentesque at nisl ac nisi fermentum viverra. Praesent odio. Phasellus tincidunt diam ut ipsum. Donec eget est. A skúška mäkčeňov a dĺžnov.

Klúčové slová: tbd

ABSTRACT

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF ELECTRICAL ENGINEERING AND INFORMATION TECHNOLOGY

Study Programme:	Applied Informatics
Author:	Peter Čuřík
Bachelor's thesis:	Password manager
Supervisor:	prof. Ing. Pavol Zajac, PhD.
Consultant:	unknown
Place and year of submission:	Bratislava 2020

On the other hand, we denounce with righteous indignation and dislike men who are so beguiled and demoralized by the charms of pleasure of the moment, so blinded by desire, that they cannot foresee the pain and trouble that are bound to ensue; and equal blame belongs to those who fail in their duty through weakness of will, which is the same as saying through shrinking from toil and pain. These cases are perfectly simple and easy to distinguish. In a free hour, when our power of choice is untrammelled and when nothing prevents our being able to do what we like best, every pleasure is to be welcomed and every pain avoided. But in certain circumstances and owing to the claims of duty or the obligations of business it will frequently occur that pleasures have to be repudiated and annoyances accepted. The wise man therefore always holds in these matters to this principle of selection: he rejects pleasures to secure other greater pleasures, or else he endures pains to avoid worse pains.

Keywords: tbd

Podčakovanie

Rád by som sa podčakoval vedúcemu tejto práce, prof. Ing. Pavlovi Zajacovi, Phd., za všetky rady, pripomienky a vedenie. Vážim si jeho trpezlivosť, každú ochotu navyše a taktiež jeho priateľský a úctivý prístup.

Obsah

Úvod	1
1 Základné pojmy	2
1.1 Heslo	2
1.2 Autentizácia	2
1.3 Sila hesla	2
1.4 Password manager	4
1.5 Šifrovanie	4
1.6 Advanced Encryption Standard	5
1.7 Secure Hash Algorithms	5
1.8 Password-Based Key Derivation Function	5
1.9 Transport Layer Security	5
2 Problém password managerov	6
2.1 Súčasný stav na trhu	6
2.1.1 LastPass	6
2.1.2 Dashlane	7
2.1.3 iCloud Keychain	8
2.2 Nízka popularita password managerov a nesprávne alternatívne spôsoby ukladania hesiel	9
2.3 Ekosystém	13
2.4 Sumarizácia	14
3 Riešenie	16
3.1 Idea	16
3.2 Aplikácia - Passer	17
3.2.1 Outsider	22
3.3 Webstránka	27
3.4 Server	29
3.5 Vzájomné interakcie	30
3.5.1 Passer - Server (šestciferný kód)	30
3.5.2 Webstránka - Server (šestciferný kód)	33
3.5.3 Passer - Server - Webstránka (QR kód)	34
3.6 Bezpečnosť	36
3.6.1 Passer	36

3.6.2	Server a webstránka	41
4	Hrozby	43
4.1	Kopírovanie do schránky	43
	Záver	45
	Zoznam použitej literatúry	46

Zoznam obrázkov a tabuliek

Obrázok 1	Schéma šifrovania aplikácie LastPass.	7
Obrázok 2	Emergency kit od 1Password - ukážka pdf súboru, ktorý používateľ obdrží.	8
Obrázok 3	Diagram implementácie nášho riešenia.	17
Obrázok 4	Úvodná obrazovka po prvom spustení.	17
Obrázok 5	Pridávanie novej položky.	18
Obrázok 6	Pridávanie novej položky do skupín.	19
Obrázok 7	Úvodná obrazovka s heslami.	20
Obrázok 8	Úvodná obrazovka s heslami po aplikovaní filtrov.	22
Obrázok 9	Outsider - výber položiek.	23
Obrázok 10	Outsider - výber typu verifikácie.	24
Obrázok 11	Outsider - QR kód.	25
Obrázok 12	Outsider - Šestciferný kód.	26
Obrázok 13	Webstránka - Úvodná obrazovka.	28
Obrázok 14	Webstránka - Zobrazenie Passer položiek.	28
Obrázok 15	Porovnanie rôznych typov pamäte v závislosti od rýchlosťi a kapacity.	30
Obrázok 16	Ukladanie dát do cache servera (šestciferný kód).	32
Obrázok 17	Ukladanie dát do cache servera (QR kód).	35
Obrázok 18	Interakcia Passera s iOS a Secure Enclave.	37
Obrázok 19	Proces šifrovania podľa algoritmu ECIES.	39
Obrázok 20	Proces dešifrovania podľa algoritmu ECIES.	40
Tabuľka 1	Zložitosť prelomenia hesiel pomocou útoku brute-force podľa webovej stránky grc.com/haystack.htm	3

Úvod

V dnešnej dobe je svet každým dňom stále väčšmi digitalizovaný. Neustále vznikajú nové produkty a služby. Systémy, ktoré nám pomáhajú mať všetko na jednom mieste. Bezpečnosť týchto systémov je rovnako dôležitá, ako jej funkčnosť. Keďže uchovávajú citlivé informácie používateľov, je absolútne klúčové chrániť ich pred útokmi. Preto veľká väčšina aplikácií a systémov, ktoré pracujú s informáciami, používa účty. Používateľ si vytvorí svoj účet a dostane sa doň pomocou dvoch vstupov: používateľského mena a hesla. Heslo jeho účet chráni, keďže používateľské meno je verejné.

Na začiatku som uvádzal, že neustále vznikajú nové produkty, služby, či systémy. Môžeme teda očakávať, že bežný človek ich bude využívať viacero na dennej báze. Povedzme, že používa email, má účty v niekoľkých sociálnych sietach, používa aplikáciu na elektronické bankovníctvo, je zaregistrovaný v niekoľkých internetových obchodoch, pravidelne pristupuje k svojim dátam na cloude (online úložisko) a podobne. Každá z týchto položiek pracuje s nejakým heslom, ktorá autentifikuje osobu, ktorá heslo zadala.

Ak hovoríme o hesle ako o retazci, teda postupnosti znakov, používateľ si tento reťazec musí pamätať, aby mohol vstúpiť do systému. Tu vzniká problém. Problém pamätania si každého hesla pre každú aplikáciu. Existujú dve riešenia. Prvou možnosťou je nastavenie ľahko zapamäteľného, prípadne rovnakého hesla do všetkých účtov. Týmto sa dramaticky znížuje úroveň bezpečnosti. Zároveň sa ale zvyšuje level komfortu pri interakcii s aplikáciami.

Tou druhou možnosťou je použitie aplikácie typu password manager (správca hesiel). Môžeme o ňom uvažovať ako o trezore. Dovnútra môžeme uložiť všetky naše heslá a zamknúť ich pod jedným klúčom. Situácia sa odrazu mení. Zrazu si nemusíme pamätať niekoľko hesiel, ale iba jedno. Úroveň komfortu pri interakcii s aplikáciami ostáva zachovaná, avšak rovnako je dosiahnutá vysoká úroveň bezpečnosti.

V praxi vytvára implementácia tohto managera určitý ekosystém. Teda, password manager dokáže poskytovať svoje služby len zariadeniu, na ktorom je nainštalovaný. Mimo tohto prostredia používateľ stráca znalosť o svojich údajoch. V súčasnosti považujeme oblasť ekosystému password managera za dostatočne rozvinutú. Preto sme sa rozhodli venovať oblasti mimo neho.

Cieľom práce je vyvinúť aplikáciu password manager a nájsť bezpečný, pohodlný a efektívny spôsob prístupu používateľa k heslám na cudzom zariadení. Teda na takom zariadení, kde password manager s citlivými údajmi používateľa nie je prítomný (mimo ekosystém). Snahou bude vymyslieť riešenie (riešenia), ktoré by toto umožňovali.

1 Základné pojmy

Pred samotným vnorením do problému, ktorým sa zaoberá táto práca je dôležité vysvetliť a definovať základné pojmy, ktoré sú spojené s danou problematikou a využívané v texte.

1.1 Heslo

Heslo je prostriedok, pomocou ktorého je overená totožnosť používateľa. [1] Pomocou neho vieme získať prístup k informáciám, dátam atď. ktoré sú pod ním uzamknuté. Teda iba ten, kto heslo pozná, môže pristupovať k týmto materiálom. Z tohto môžeme usúdiť, že heslo by malo byť dostatočne silné. Musí byť tažko uhádnuť a komplexné. Jeho vlastník by ho mal ukryť pred odhalením alebo uhádnutím útočníka. Týmto nám vznikajú rôzne otázky: *Aké miesto je bezpečné na ukrytie hesla? Kedy môžeme prehlásit, že heslo je „silné“?*

1.2 Autentizácia

Proces, pri ktorej je overená totožnosť osoby, sa nazýva autentizácia. Predchádza ju proces identifikácie, kedy sa osoba „predstaví“ a povie, kto je. Systém ho dalej v procese autentizácie „vyzve“, aby dokázal, že dotyčná osoba je naozaj tou, za ktorú sa prehlásil. Tým dôkazom myslíme vyššie spomínané heslo. [2]

V praktickej rovine sú spôsoby autentizácie rôzne, ako napríklad: biometrický odtlačok, fráza vo forme hlasu, textového retazca, číselný PIN a podobne [3].

1.3 Sila hesla

Uvažujme heslo ako textový retazec. Sila hesla označuje stupeň obtiažnosti s akou ho neautorizovaná osoba dokáže uhádnuť [4]. Heslo môže byť silné alebo slabé, v závislosti od toho, ako ľahké ho je uhádnuť [4]. Slabé heslo je napríklad používanie iba malých písmen alebo iba číslík. Dôvod, prečo to tak je, je príliš malý priestor výberu znaku. Pri číselnom hesle hovoríme o priestore desiatich znakov. Uvažujme štandardnú telegrafnú abecedu s 26 písmenami. Potom je priestor pri použití hesla iba z malých písmen veľký 26 znakov. Útočník môže predpokladať, že používateľ má heslo zložené iba z číslík alebo iba z malých písmen¹.

Preto na druhej strane hovoríme, že silné heslo je také heslo, ktoré obsahuje kombináciu veľkých a malých písmen a číslík. Už len kombináciou veľkých a malých písmen sa nám

¹Možnosť použitia hesla iba z veľkých písmen nespomíname, pretože z matematického hľadiska náročnosti prelomenia hesla ide o rovnaký prípad ako pri malých písmenách.

priestor zdvojnásobí. Abeceda veľkých písmen aj malých písmen má 26 znakov, čo je spolu 52 znakov. Zrazu je pre útočníka pri každom písmene nutné uvažovať, či sa použilo ako veľké, alebo ako malé. Z matematického hľadiska, teda z hľadiska permutácií sa celkový počet možných usporiadanií exponenciálne zvýši. Permutácia znamená usporiadanie.

Tabuľka 1: Zložitosť prelomenia hesiel pomocou útoku brute-force podľa webovej stránky grc.com/haystack.htm

Typ Hesla	Heslo	Priestor	Počet možností
Číslice (ďalej len C)	01234	10	$1,11 * 10^5$
Malé písmená (ďalej MP)	heslo	26	$1,24 * 10^7$
MP + veľké písmená (VP)	hEsLo	52	$3,88 * 10^8$
MP + VP + C	h3sL0	62	$9,31 * 10^8$
MP + VP + C	h3sL0jeSiLn3	62	$3,28 * 10^{21}$
MP + VP + C + špec. znaky	h3sL0=%SiLn3	95	$5,46 * 10^{23}$

Heslo	Čas (online útok)	Čas (offline útok)
	pri 1000 pokusoch/s	pri miliarde pokusoch/s
01234	1,85 min	0,00000111 s
heslo	3,43 hod	0,000124 s
hEsLo	4,49 dní	0,00388 s
h3sL0	1,54 týždňov	0,00931 s
h3sL0jeSiLn3	104 miliárd rokov	1043 rokov
h3sL0=%SiLn3	17,4 biliónov rokov	1740 rokov

Z tabuľky sme pozorovaním zistili, že rovnako ako bohatý priestor znakov je dôležitá aj dĺžka hesla. S použitím veľkých aj malých písmen pri dĺžke hesla 5 by bol útočník schopný zistiť naše heslo za veľmi krátke čas. Môžeme si z časových výsledkov všimnúť, že z praktického hľadiska skoro ani nezáleží, či použijeme C, MP, MP + VP alebo MP + VP + C, pokial je heslo krátke. Najmä pri offline útoku zjavne vidieť, že vo všetkých prípadoch by stroj uhádol heslo doslova do sekundy.

Silu exponenciálneho rastu si všimame pri zmene dĺžky hesla na 12 znakov. Celá situácia sa dramaticky zmenila a kombinácie C, MP a VP už dávajú zmysel. Ďalší veľký skok spôsobilo pridanie špeciálnych znakov. Kedže sme zväčšili priestor rôznych znakov o viac ako polovicu, významná zmena je vidieť aj vo výsledkoch.

Predpokladajme, že útočník má informáciu, že používateľ vlastní heslo zložené iba

z MP. Potom platí, že ak by používateľ zväčšil dĺžku hesla o jeden znak, útočník musí vykonať v priemere o 26-krát viac pokusov. [5]

Ďalej predpokladajme, že používateľ vlastní heslo zložené z MP, VP a C. Takáto kombinácia je dnes pri registrácii vo veľkej miere povinnosťou na rôznych webových stránkach. Potom platí, že pri zväčšení dĺžky hesla o jeden znak by sa zložitosť hesla nezvýšila iba 26, ale až 62-násobne. Z toho vyplýva, že útočník by musel mať 62-násobne väčší výkon, aby mohol za rovnaký čas zlomiť heslo z pôvodnej dĺžkou. Ten sa zvyšuje každé dva roky dvojnásobne, podľa Moorovho zákona. [5]

Táto úvaha spolu s ďalšími typmi útokov a ochranou pred nimi je hlbšie obsiahnutá v práci [5].

1.4 Password manager

Password manager (alebo po sl. správca hesiel) je aplikácia, ktorá umožňuje vytváranie, uchovávanie a používanie rôznych hesiel [6]. Zhromažďuje ich na jednom mieste a vytvára pre používateľa prehľad jeho prístupových údajov. Tento „trezor“ je chránený master heslom.

Master heslo (často sa s ním stretnete v angl. forme „master password“) je heslo, ktoré je použité na sprístupnenie iných hesiel [7]. Táto skutočnosť nám odhalila jednu pozitívnu a jednu negatívnu stranu password managerov všeobecne. Pozitívom je, že namiesto n hesiel si stačí pamätať práve jedno heslo - master heslo. Týmto bolo rovno zodpovedané aj negatívum. Ak sa vieme jedným heslom dostať ku všetkým ostatným, stačí zlomiť master heslo a útočník získa všetky údaje v trezore, teda v password manageri.

1.5 Šifrovanie

Nasledujúci text vychádza zo zdroja [8].

Šifrovanie je prepis otvoreného (čitateľného) textu do zašifrovaného textu, ktorý nazývame šifra. Abeceda, z ktorého vychádza otvorený text budeme označovať ako \mathcal{P} a abecedu, z ktorého bude vychádzať zašifrovaný text budeme označovať ako \mathcal{C} .

$$e_k = \mathcal{P} \rightarrow \mathcal{C} \tag{1}$$

Vidíme, že ide o zobrazenie. Toto zobrazenie je bijektívne, no nie vždy je tomu tak (napríklad pri znáhodnených šifrách). Je závislé na tajnom parametri k , ktorý nazývame kľúč. Ten patrí do množiny kľúčov K .

Aby vedel príjemca šifru prečítať, musí byť spomínané zobrazenie invertovateľné. To

znamená, že musí byť použité inverzné zobrazenie

$$d_k = \mathcal{C} \rightarrow \mathcal{P} \quad (2)$$

také, aby platilo: $d_k(e_k(x)) = x$, kde x je nezašifrovaná správa. Z týchto vzťahov je zrejmé, že príjemca musí použiť totožný klúč k s klúčom odosielateľa, aby sa dostal k x . Útočník sa snaží nájsť tento klúč. Preto čím väčšia je množina K , tým náročnejšie, niekedy až nemožné z hľadiska výpočtovej sily, je nájsť k .

V dobe klasických šifier (obdobie do roku 1945) väčšinou spočívala bezpečnosť niektorých algoritmov v ukrytí algoritmu samotného. To ale nie je správny prístup, pretože podľa Kerckhoffovho princípu [8] má bezpečnosť šifrovacieho algoritmu spočívať na utajení klúča, nie algoritmu samotného. Týmto princípom sa riadia dnešné, moderné šifry dodnes.

1.6 Advanced Encryption Standard

TODO

1.7 Secure Hash Algorithms

TODO

1.8 Password-Based Key Derivation Function

TODO

1.9 Transport Layer Security

TODO

2 Problém password managerov

Password manager ponúka používateľovi mnohé výhody a možnosti. Od generátora hesiel a ich automatického vyplnenia pri prihlásovaní do stránok, cez synchronizáciu medzi zariadeniami, zálohovanie, až po automatickú, pravidelnú zmenu jednotlivých hesiel. Napriek tomu existuje problém týchto managerov, ktorý ostal neriešený. Jemu sa bude táto kapitola venovať.

2.1 Súčasný stav na trhu

Ak hovoríme o heslách, priame (explicitné) používanie nižšie uvedených aplikácií pri dennom používaní je minimálne. A to vďaka automatickému vyplňaniu hesla. Používateľ vstúpi na webovú stránku alebo do aplikácie. Pred vstupom sa musí prihlásiť do svojho účtu. Tam mu password manager výzvou ponúkne automatické vyplnenie (angl. výraz „*autofill*“) prihlasovacieho mena a hesla. Toto poskytuje vysokú úroveň komfortu. Používateľ nielen že nemusí vypisovať svoje prihlasovacie údaje manuálne, ale aj ich bezpečnosť porastla na vyššiu úroveň. Vďaka password manageru.

Podobne to funguje pri registrácii nového konta. Niektoré password manager aplikácie ponúknu náhodne vygenerované silné heslo (iCloud Keychain, [9]). Používateľ sa môže rozhodnúť ho prijať. V takom prípade sa heslo pre vytvorený účet uloží do password manageru a ten ho pri každom ďalšom prihlásení vyplní.

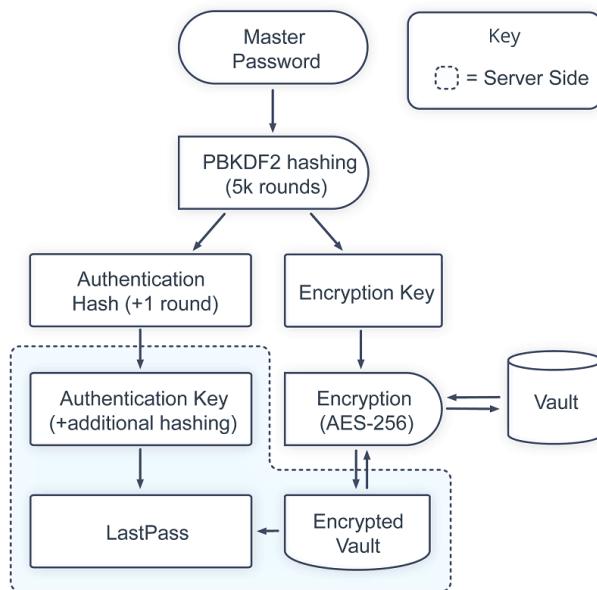
2.1.1 LastPass

LastPass patrí medzi jeden z najpopulárnejších, čo sa týka počtu používateľov [10]. Všetky heslá a ostatný obsah sú uzamknuté pod jedným master heslom. LastPass ale povoľuje aj vstup do jeho aplikácie pomocou biometrickej autentifikácie, ktorú väčšina smartfónov podporuje. Používateľ tak nemusí každý raz písat dlhé master heslo. Jediné, čo stačí, je priložiť prst, či pozrieť sa na smartfón (sken tváre).

LastPass je client-server aplikácia. To znamená, že niektoré operácie a procesy prebiehajú na strane klienta, teda priamo v danom zariadení, ktoré používateľ drží a niektoré prebiehajú na strane LastPass serverov. Samotné šifrovanie, aj dešifrovanie prebieha na strane zariadenia [11]. LastPass vytvorí kľúč z master hesla pomocou šifry AES (Advanced Encryption Standard) s hašovaním PBKDF2 (Key Derivation Function), SHA (Secure Hash Algorithm) s pridaným saltom [12].

Dáta sú synchronizované pomocou serverov, čo umožňuje zálohovanie a synchronizáciu medzi zariadeniami. Dátový prenos po sieti je chránený pomocou TLS/SSL ([13] sa bližšie venuje tomuto protokolu). LastPass v bezpečnosti pokračuje v možnosti dvojfak-

torovej autorizácie a overovania na základe lokácie: kedykoľvek sa užívateľ prihlásuje do aplikácie z inej lokality, je vyzvaný prostredníctvom emailu s linkom, ktorý po otvorení overí používateľa ako verifikovaného. LastPass má mnoho možností, ako napríklad zdieľanie hesiel s iným LastPass účtom, generovanie hesiel s používateľom zvolenou dĺžkou a podmienkami, hodnotenie sily hesiel (systém usúdi, či je heslo dostatočne bezpečné), či import hesiel pomocou CSV súboru alebo iného password manageru.



Obr. 1: Schéma šifrovania aplikácie LastPass.

2.1.2 Dashlane

Tento password manager je tiež vysoko využívaný [14]. Jedna z jeho jedinečných funkcia je automatická zmena hesla [15]. Používateľ si môže vybrať v zozname svojich hesiel, ktoré by sa mali automaticky meniť. Dashlane tak bude pravidelne generovať nové, komplexné heslá pre vybrané položky. Kedže medzi rôznymi webovými stránkami nie je jednotná architektúra a dizajn, nie všetky položky účtov vie Dashlane meniť automaticky. V takom prípade si užívateľ vie meniť heslo pre danú položku iba manuálne, v konkrétnej aplikácii alebo webovej stránke pre službu, ku ktorej prislúcha daný účet v Dashlane.

Tak ako LastPass, aj tento password manager poskytuje silnú bezpečnosť. Podporuje dvojfaktorovú autentifikáciu (dodatočné overenie po prvotnom prihlásení, viac v [16]), šifru AES a synchronizáciu medzi zariadeniami. Mnohé z funkcia a možností má spoľahlivé s aplikáciou LastPass. Nebudeme ich opäť spomínať, nakoľko cielom tejto kapitoly je iba ukázať už existujúce výmožnosti rôznych password managerov.

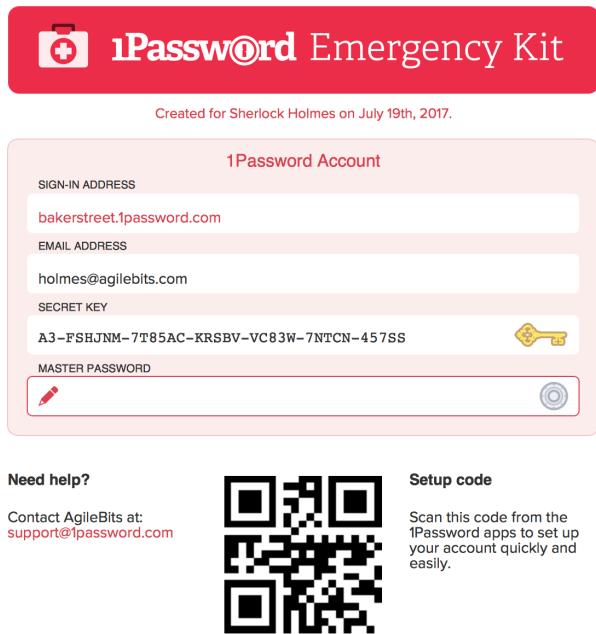
2.1.3 iCloud Keychain

Za spomenutie určite stojí vstavaný password manager od spoločnosti Apple. Obsahuje určité funkcionality password managera, ktoré sú základnou súčasťou každého iOS, iPadOS, či MacOS zariadenia. Každý používateľ nejakého Apple zariadenia je identifikovaný pomocou AppleID konta. K nemu má pridelené cloud úložisko, kde sa mu všetky dátá zálohujú a synchronizujú so všetkými Apple zariadeniami.

Okrem fotiek, poznámok, kontaktov a iných dát tam sú uložené aj heslá z Keychainu. Keychain si pamäta nielen heslá, ale aj certifikáty, dôležité pri rôznych verifikáciách v rámci systému. Taktiež si pamäta klúče, ktoré sa v kryptografii používajú na šifrovanie, digitálne podpisovanie, či overenie totožnosti komunikačných strán. Existujú verejné, ale aj súkromné klúče. Keychain obsahuje oboje. Systém ich používa napríklad pri používaní iMessage (četovacia aplikácia).

Z ostatných managerov ukážeme zopár ďalších funkcií, ktoré neboli spomenuté, respektíve ich LastPass, Dashlane, či Keychain neposkytujú.

1Password ponúka takzvaný „Emergency kit“ (Obr. 2). Po vytvorení 1Password účtu je používateľovi poskytnutý dokument. Následne je vyzvaný, aby si ho vytlačil, prípadne uložil na pamäťové médium. Dokument obsahuje prihlásovacie údaje a master heslo. Tiež obsahuje QR kód, ktorý automaticky vyplní tieto dátá pri núdzovom prihlásovaní.



Obr. 2: Emergency kit od 1Password - ukážka pdf súboru, ktorý používateľ obdrží.

Password manager Remembeare ponúka jednoduchú aktiváciu aplikácie na novom zariadení prostredníctvom QR kódu. Za spomenutie stojí aj menej známy správca, konkrétnie Myki.

Myki ako jeden z mála nepoužíva svoje servery na zálohovanie a ukladanie hesiel. Namiesto toho využíva server iba ako sprostredkovateľa [17] spojenia, keď nastáva synchronizácia medzi zariadeniami. Teda, používať Myki na viacerých zariadeniach je akousi formou „zálohy”.

Vyššie spomenuté aplikácie fungujú na rôznych platformách: verzia pre smartfóny (iOS, Android), počítače (Windows, MacOS, Linux), inteligentné hodinky a podobne.

2.2 Nízka popularita password managerov a nesprávne alternatívne spôsoby ukladania hesiel

Lahko sme vedeli ukázať, že bohatosť a rôznorodosť trhu s aplikáciami na správu hesiel je naozaj veľká. Používateľ by teda nemal mať problém vybrať si takú aplikáciu, ktorá vyhovuje jeho požiadavkám. Napriek tomu je popularita password managerov nízka.

Už nadpis článku [18] začína slovami „*Hardly Anybody Uses a Password Manager*”. Teda v slovenskom znení - „Horko-ťažko niekto vôbec používa password manager”. Píše o prieskume na túto tému a jeho výsledkoch. V Spojených štátach a v Anglicku sa pýtali 1000 respondentov rôzne otázky. Týkali sa bezpečnosti pri používaní internetu. Išlo najmä o praktiky pri používaní hesiel; či pre každú stránku používajú iné heslo, či je heslo silné, dlhé alebo krátke, „náhodné” alebo ľahko zapamäteľné. Spomedzi opýtaných, 59% uviedlo, že používa 5 alebo menej rôznych hesiel, ktoré si pamätajú v hlave. Pritom 74% sa denne prihlási do 6 a viac účtov. Môžeme už teraz hovoriť o pravdepodobnosti faktu, že niektorí z týchto používateľov používajú rovnaké heslo pre rôzne účty. Túto skutočnosť využívajú útočníci. Ak získajú heslo k jednému účtu, predpokladajú, že ho používateľ využíva aj v iných účtoch. Najmä, ak je heslo jednoduché. A ak hovoríme o respondentoch, ktorí si dokážu pamätať až do 5 hesiel v hlave a používajú ich na prihlásование do aspoň šiestich účtov, je veľmi pravdepodobné, že nie sú náročné (ani z hľadiska zapamäteľnosti, ani z hľadiska kryptografie).

Ďalšia negatívna skutočnosť, ktorá vyplýva z prieskumu je písanie hesiel na papier. Až 42% opýtaných používa túto metódu. Niekoľko argumentoval, že rovnako ako papier s heslami vie niekto ukradnúť aj server, kde sú uložené heslá password manageru. Aj keď sa môže zdať, že princíp je ten istý, na rozdiel od papiera s ručne vypísanými heslami sú heslá na serveroch password managerov zašifrované. Dnešné moderné šifry, ktoré sa na ich zašifrovanie používajú sú matematicky silné a v súčasnosti neprekloniteľné. Tá

tažšia časť je ukryť kľúč tak, aby ho nikto nenašiel. Dôvodom úspešných útokov teda nie je nedokonalosť moderných šifier. Väčšinou ide o postranné kanály, zlé ukrytie klúča, spiknutie zvnútra a podobne.

Ked máme hovoriť o popularite password managerov, tento prieskum silno podporuje nadpis tejto podkapitoly. Len 8% opýtaných používa na svoje heslá nejakého správcu. Naopak, menej než tri štvrtiny si vystačí s tým, že si ich heslá zapamätá prehliadač. Tu treba uviesť skutočnosť, že ukladanie hesiel do prehliadača je náchylné na krádež pomocou malvéru (škodlivý softvér, ktorý sa snaží infikovať zariadenie, väčšinou využívaný na krádež citlivých údajov, najmä hesiel [19]).

Populárny prehliadač Google Chrome ponúka používateľovi možnosť zapamätať si heslá pri prihlásovaní na rôznych stránkach. Tieto heslá sú synchronizované do všetkých zariadení pomocou Google konta. Je ľahké sa dostať do tohto trezoru, stačí do URL text-boxu prehliadača napísat „chrome://settings/passwords“ [20]. Zobrazí sa zoznam hesiel pre dané stránky, ktoré si prehliadač ukladal. Jedným tlačidlom sa odokryje heslo ako otvorený text. Chrome nevyžaduje žiadnu autentizáciu. Je jednoduché pre útočníka pri fyzickom prístupe k zariadeniu tieto dátu získať pári klikmi. Chrome vývojár Justin Schuh avšak argumentuje [20]. Hovorí, že keď má niekto prístup do účtu operačného systému používateľa, vie, vidí a má prístup ku všetkému. Môžeme z tohto tvrdenia vyvodíť záver, že Chrome nevidí zmysel v chránení trezoru prehliadača (okrem hesla do Googlu účtu), lebo rozumie faktu, že ten, kto má prístup do operačného systému, má aj tak prístup ku všetkému.

Mozilla Firefox tiež nechráni svoj trezor hesiel, avšak narozenie od Chromu ponúka aktiváciu master hesla. Spomeňme ešte Safari od Apple, ktorý chráni celý trezor heslom účtu operačného systému. Z výroku Schuha vyplýva, že takéto zabezpečenie je zbytočné. Apple pravdepodobne počíta s tým, že môže prísť k zneužitiu zariadenia už po prihlásení do systému, teda útočník nemusí poznáť heslo. V takom prípade považujeme takéto zabezpečenie za múdre, avšak určite existujú bezpečnejšie spôsoby (za cenu komfortu).

Spomínaný prieskum z roku 2015 nepriniesol pozitívne výsledky. O tri roky neskôr sa uskutočnil ďalší [21]. Spomedzi 2500 Američanov si 35% nikdy heslá nemení. Robí tak iba po vyzvaní. Veľkým prekvapením bolo 11% používateľov, ktorí si ich menia každý deň. The National Institute of Standards and Technology radí používateľom, aby si heslá menili nie pravidelne, ale až keď nastane ich prelomenie. Ked padla otázka, aký nástroj respondenti používajú na svoju ochranu na internete, víťazom bol antivírový software

(53%), password manager získal 24%, čo je trojnásobný nárast za obdobie troch rokov².

Napriek pozitívнемu nárustu popularity password managerov považujeme 24% za malé číslo. Je pravdou, že tieto aplikácie sú na trhu nie tak dlho. Antivírusový softvér má v tomto časový náskok. Tento softvér má avšak iný cieľ a zameranie v bezpečnosti, než password manager. Bolo by nezmyselné tieto dva nástroje na bezpečnosť porovnávať ako dve technologické riešenia, ktoré slúžia na rovnaký účel. U niektorých čitateľov možno vzniká otázka typu: *Prečo používatelia nepoužívajú viac bezpečnostné nástroje na ochranu ich osobných údajov?* V druhom spomínanom prieskume z roku 2018 sa opýtaných pýtali aj na otázku, kedy boli poučení, respektíve vzdelaní na tému bezpečnosti na internete. Až 36% nedostalo na túto tému žiadne vzdelanie. Aj toto môže byť odpovedou na vyššie spomínanú otázku.

Ďalším dôvodom môže byť nedôvera. Používatelia nemusia byť presvedčení o tom, že password manager naozaj ich heslá ochráni, nezverejní, prípadne nezneužije. Jedna práca [22] študovala, prečo má password manager stále málo používateľov. Už v úvode vyjadrila počudovanie nad faktom, že ľudia využívajú password manager minimálne. Podľa štúdia tejto práce má priemerný používateľ 25 rôznych online účtov, každý s jedným heslom. Je náročné si toto všetko pamätať. Experti na bezpečnosť odporúčajú používať ako riešenie password manager. Napriek týmto odporučeniam väčšina ľudí zvolí nepoužívať ho. Práca sa preto rozhodla urobiť prieskum, kde prizve 137 respondentov používajúcich password manager a 111 takých, ktorí ho nepoužívajú. Vznikla štúdia, ktorá porovnáva odpovede na 6 otázok týchto dvoch skupín a snaží sa vyvodíť záver, prečo je popularita týchto aplikácií taká, aká je. Kvôli jednoduchosti budeme nazývať respondentov používajúcich password manager ako „používajuci“ a respondentov nepoužívajúcich password manager ako „nepoužívajúci“.

Používajúci vidia password manager ako nástroj zvyšujúci pohodlie, či použiteľnosť. Tvrdia, že jeho význam narastá so zvyšujúcim sa počtom hesiel. Nepoužívajúci, na druhej strane, vyjadrujú nedôveru v bezpečnosť takej aplikácie. Považujú za nemúdre dávať všetky heslá na jedno miesto. Tieto poznatky naznačujú, že to, ako funguje password manager nie je dostatočne pochopené. Používatelia si neuvedomujú bezpečnostné benefity, ktoré im systém ponúka. Je preto možné, že zlepšenie informovanosti a úrovne vzdelania v tejto oblasti napomôže k väčšej adaptácii password managerov. Štúdia ďalej vysvetluje, že ak je používateľ menej zručný v ovládaní počítača, môže sa zdráhať priať nové nástroje. A to preto, lebo tento akt vyžaduje naučiť sa niečo nové. To môže vyvolať emócie

²vychádzajúc zo vzorky opýtaných, teda určitá štatistická odchylka je pri týchto úvahách samozrejmostou.

frustrácie, úzkosti, diskomfortu. Na druhej strane, pravdepodobnosť krádeže hesla môže v niekom vzbudzovať strach. Ak je dostatočne silný, používateľ je motivovaný k použitiu password managera.

Vo vyššie spomínaných šiestich otázkach, ktoré boli v tejto práci respondentom položené nás najviac zaujala otázka 4. V nej má respondent pomenovať dôvod, prečo používa, respektíve nepoužíva password manager. Za uvedením tohto dôvodu nasleduje stručné odôvodnenie.

Hlavný dôvod (80%) je pohodlnosť. Týmto respondentom password manager v prvom rade uľahčuje pamätať si nekonečné množstvá hesiel. Najmä tie komplexné. Už ich mali príliš veľa na to, aby si to písali na papier. Bezpečnosť prichádza na druhom mieste. Používatelia, ktorí bezpečnosť považujú sa dôvod č. 1 hovoria, že password manager robí ich heslá viac bezpečnými. Je to viac bezpečné, ako písanie na papier a viac presné, než pamätanie v hlave. Niektorí respondenti uviedli viac dôvodov súčasne, preto má bezpečnosť až 25% zastúpenie.

Presne to, čo používajúci považujú za silné stránky považujú nepoužívajúci za tie negatívne. Vyjadrujú obavy o bezpečnosť (46%), nepohodlnosť (9%) tvrdia, že takú aplikáciu nepotrebuju (42%). Hovoria, že nie je bezpečná. Je riskantné mať všetky heslá na jednom mieste, lepšou cestou je pre nich pamätať si ich v hlave. Okolo 11% nemá čas na študovanie fungovania password managerov. Priznávajú aj lenivosť.

Po celkovej štúdii práca sumarizuje výsledky. Tvrď, že používajúci sú ľudia s vyššiou zručnosťou v práci s počítačom. Majú lepšie skúsenosti s počítačovou bezpečnosťou. Nepoužívajúci tvrdia, že sú v práci s počítačmi menej zruční, majú málo účtov, ktoré používajú často. Priznávajú, že ich heslá môžu zlepšiť, ale nesúhlasia, že by password manager bol vhodným nástrojom pre tento účel. Použiteľnosť je hlavným pozitívom pre používajúcich. Obavy o bezpečnosť je hlavným negatívom pre nepoužívajúcich. Tvrď, že nepovažujú správcov hesiel za bezpečných. To je pre niektoré aplikácie nepravdivé tvrdenie, ak sú používané správne. Navyše, používajúci si myslia, že nepoužívajúci nevedia o bezpečnostných benefitoch password managera. Toto by mohlo viesť k nesprávnemu porozumeniu celého nástroja.

Ked' uvážime fakt, že reputácia password managerov je o tom, že ide o pohodlný nástroj, ktorý má zdokumentované bezpečnostné slabiny, výpovede nepoužívajúcich sa začínajú viac a viac javiť ako racionálne. Riešením na všeobecné prijatie password managerov by mohli byť kampane a lepšie vzdelávanie používateľov.

2.3 Ekosystém

Vráťme sa ku kapitole Súčasný stav na trhu, kde sme rozoberali existujúcich password managerov. Pri ich podrobnej analýze sme dospeli k problému.

Všetky z nich (pochopiteľne) uchovávajú a chránia heslá používateľa. To znamená, že sú prístupné iba na zariadení, ktoré v sebe obsahuje aplikáciu. Preto napríklad LastPass ponúka browser extension [11]. Používateľ si do internetového prehliadača môže nainštalovať rozšírenie. Prihlási sa so svojim LastPass účtom a tak môže k svojim heslám pristupovať. Teda, používateľ by mal LastPass nainštalovaný na svojom smartfóne, aj počítači s ľubovoľným prehliadačom. Toto mu ulahčí prístup k heslám.

Problémom je fakt, že používateľ potrebuje svoje heslá vždy, keď sa chce prihlásiť do nejakého z jeho účtov. Bez ohľadu na to, na akom zariadení prihlasovanie vykonáva. Zistili sme, že súčasné riešenia nedokážu pokryť tento problém v plnosti. Aj keď väčšina password managerov poskytuje podporu na rôznych zariadeniach (smartfóny, počítače, inteligentné hodinky...), väčšinou ide o niečo, čo používateľ vlastní a pravidelne využíva.

Teda hľadáme odpoveď na otázku, čo má používateľ robiť, ak používa password manager, no potrebuje sa prihlásiť na cudzom zariadení.

Priblížme si vyššie uvedenú problematiku na konkrétnom príklade. Majme používateľa password managera, ktorý sa momentálne nachádza na letisku. Svoju letenkou si potrebuje vytlačiť z emailu na jednom z letiskových verejných počítačov. Môžeme už teraz povedať, že je malá pravdepodobnosť, že daný počítač použije niekedy opäť. Navyše, má málo času, pretože lietadlo odchádza o pári desiatok minút. Najprv sa musí prihlásiť do svojho emailu. Heslo si nepamätá, je príliš komplexné. Použije svoj smartfón s nainštalovaným password managerom, aby zistil, aké má heslo do svojho emailu.

Teraz potrebuje ručne opísat používateľské meno a heslo do cudzieho počítača. Už počas tejto operácie vidíme niekoľko bezpečnostných hrozieb. Počas ručného opisovania je telefón príliš dlho vystavený krádeži. No možno ani nie je nutné ukradnúť samotný telefón. Útočníkovi stačí odfotiť obrazovku telefónu, prípadne si nenápadne heslo opísat na papier spoza chrbáta používateľa (toto je jednoduché docielit, najmä ak je pred verejnými počítačmi rad ľudí alebo práve prechádza okolo veľký dav).

Nehovoríme o bezpečnostnej hrozbe softvéru, ale o jeho praktickom používaní. Na prácu so svojimi účtami používajú používateelia väčšinou osobné zariadenie. Ale ak sa už stane, že musia určitý úkon s prihlasovaním vykonať na cudzom zariadení, v mnohých prípadoch to je na verejnosti (riziko napadnutia). Taktiež v mnohých prípadoch ide o časový zhon. V bežných situáciách nie je súčasťou plánu používať na osobné záležitosti

cudzie zariadenie. Ručné prepisovanie hesla teda nie je len málo bezpečné, ale aj časovo nepraktické. Navyše, môže sa stať, že komplikované heslo používateľ správne prepíše až na niekolký pokus. Pri zadávaní hesla je väčšinou heslo nahradené hviezdičkami, prípadne bodkami. Používateľ si nemusí všimnúť chybu pri písaní. Alebo si nevšimne, že je v počítači nastavený iný jazyk klávesnice. Na zahraničných letiskách môže byť nastavená anglická klávesnica. Tá má znaky *z* a *y* vymenené oproti slovenskej. Taktiež nie je potrebné držať klávesu *Shift* pri písaní čísel v hornej časti klávesnice. Špeciálne znaky majú tiež špecifickú klávesovú distribúciu v závislosti od jazyka klávesnice. Možnosti, ako nesprávne zadať heslo je viacero.

Existujú aj iné spôsoby, ako by používateľ na letisku mohol takúto situáciu riešiť. Ak jeho password manager podporuje aplikáciu pre operačný systém daného počítača, môže si ju stiahnuť a nainštalovať. Tu sa ale časová náročnosť oveľa viac preťahuje. Používateľ potrebuje aplikáciu vyhľadať na internete. Potom ju musí stiahnuť. Stahovanie môže trvať dlho, ak je internetové pripojenie pomalé. Potom musí aplikáciu (prípadne browser extension) nainštalovať. Ak má počítač slabý výkon, táto operácia bude trvať o to dlhšie. Inštalácia sa ani nemusí podaríť spustiť, kvôli právam v operačnom systéme. Musíme počítať s tým, že letisko mohlo zakázať inštalovať akýkoľvek softvér na ich počítačoch. To je pochopiteľné, nakoľko cudzí softvér môže obsahovať vírusy. Ak sa inštalácia podarí, používateľ sa do aplikácie musí prihlásiť, nájst si heslo, ktoré práve potrebuje a použiť ho na prihlásenie. Vidíme tu náročný proces, ktorý nemusí byť praktický.

Hovoríme o ekosystéme, ktorý password manager vytvára. Iba tam, kde je nainštalovaná aplikácia manager funguje. Mimo aplikácie je používateľ vo veľmi nepríjemných podmienkach.

2.4 Sumarizácia

Po analýze existujúcich riešení a mienke vzorky spoločnosti vieme veľa poznatkov. Existuje mnoho password managerov. Väčšina z nich je naozaj bezpečná a ponúka používateľovi mnohé možnosti. Táto technológia je avšak celkom nová. Správcovia hesiel sa (nielen) preto netešia vysokej popularite. Ďalšími dôvodmi môžu byť faktory ako: nevzdelenosť v danej problematike, nedôvera (aj vdaka rôznym bezpečnostným zlyhaniam populárnych password managerov), či fakt, že tieto aplikácie niektorí používatelia nepotrebuju alebo nepreferujú. Podľa prieskumov z rôznych rokov je avšak vidno, že ich popularita napriek všetkému rastie. Z tých, ktorí ho používajú na bežnej báze vieme, že klúčová je pre nich pohodlnosť a bezpečnosť.

Náš cieľ je po analytickej časti práce jasný. Potrebujeme password manager, ktorý

je jendoduchý na používanie. Týmto zminimalizujeme odpor, strach, či úzkosť naučiť sa niečo nové. Toto uviedlo ako problém viacero respondentov zo skupiny nepoužívajúcich password manager. Taktiež potrebujeme password manager, ktorý je kvalitne zabezpečený. To je samozrejmostou, nakoľko bezpečnosť je hlavným princípom takejto aplikácie. Z odpovedí používajúcich vieme, že oceňujú na prvom mieste pohodlnosť používania. Napriek tomu, že mimo ekosystému tento faktor dramaticky klesá. Preto sme presvedčení, že respondenti uviedli túto odpoveď vzhľadom k ich osobným zariadeniam, ktoré pravidelne využívajú a majú v nich nainštalovanú aplikáciu.

Ako sme spomínali v úvode práce, oblasť ekosystému password managerov považujeme za dostatočne rozvinutú. Toto potvrdzujú aj uvedené prieskumy. Respondentmi uvedené výhody ako bezpečnosť, či pohodlnosť používania ukazujú výborný stav password managerov súčasnosti. Preto sa ich ekosystému venovať nebudeme.

Cieľom bakalárskej práce je tento ekosystém otvoriť. Vyvinúť password manager a nájsť bezpečný, pohodlný a efektívny spôsob prístupu používateľa k heslám daného password managera na cudzom zariadení. Výsledkom môže byť aj ukážka toho, že takéto niečo nie je možné. Je totiž pravdepodobné, že to už existujúce, veľké password manager spoločnosti skúšali, ale neuspeli. My si však myslíme, že nejaký spôsob existuje. No jeho zložitosť používania, časová zložitosť, prípadne iný faktor môžu byť horšie, než existujúce spôsoby (príklad z letiska: prepisovanie hesla z telefónu do cudzieho zariadenia pri prihlásovaní). V takom prípade nebudeme považovať výsledok tejto práce za riešenie.

3 Riešenie

V predchádzajúcej kapitole sme predstavili existujúce riešenia, pohľad ľudskej spoločnosti, existujúce problémy, náš cieľ. Na základe týchto skutočností sme sa rozhodli vyvinúť riešenie, ktorého hlavné komponenty popisujeme v tejto kapitole.

3.1 Idea

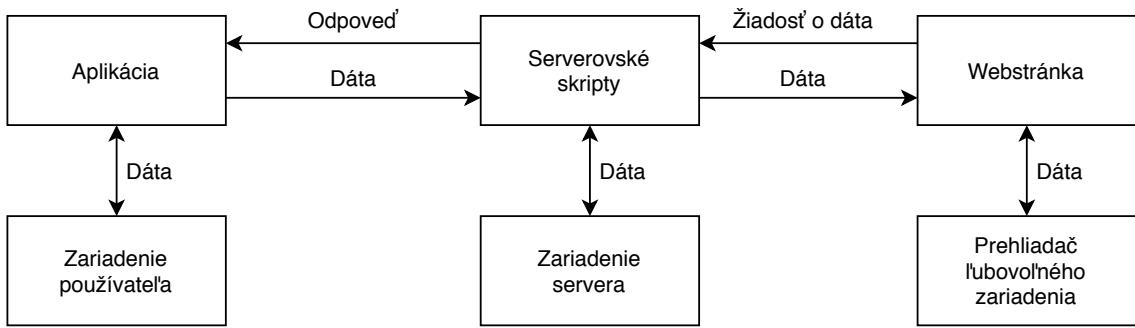
Pre náš projekt potrebujeme vytvoriť tri veci. Potrebujeme aplikáciu, ktorá bude password managerom pre používateľa. Bude podporovať štandardné operácie už existujúcich password managerov, ako: pridanie hesla, vymazanie hesla, zobrazenie všetkých hesiel, kategorizácia hesiel. Aplikáciu obohatíme o novú funkciu, ktorá nejakým spôsobom dostane heslá na cudzie zariadenie. A to všetko bezpečne.

Potrebujeme webstránku. Toto bude jediný komponent, ktorý bude prístupný na cudzom zariadení, keďže verejná webstránka je prístupná všade, kde existuje pripojenie k internetu. Na nej by sa používateľ verifikoval jednorázovým kódom alebo QR kódom a dostal by sa ku svojim dátam jednorázovo. Zo stránky by si údaje mohol skopírovať a používať počas používania cudzieho zariadenia.

Potrebujeme server. Kvôli rozsahu bakalárskej práce nebude podporovať synchronizáciu hesiel medzi zariadeniami používateľa. Teda, heslá budú existovať iba lokálne, vo filesystéme daného zariadenia. Server bude mať inú úlohu. Bude kľúčovým elementom, ktorý je zodpovedný za bránu von z ekosystému. Bez servera by nebolo možné dostať dátá z aplikácie von. Môžeme ho vnímať ako spojku medzi aplikáciou a webstránkou. Mal by nejakým spôsobom uchovávať dátá vyslané používateľom na obmedzenú dobu.

Heslá uložené v zariadení, ktoré má nainštalovanú aplikáciu musia byť chránené. Preto v čase, keď je aplikácia vypnutá, musia byť heslá zašifrované. Klíč od šifry musí byť kvalitným spôsobom ochránený a nedostupný. Cestovanie dát po sieti (od aplikácie na server, od servera na webstránku) musí prebiehať rovnako bezpečne. Ak nevieme toto docieliť, neposkytli sme riešenie na celú problematiku. Implementácia tohto problému musí na prvom mieste splňať podmienky silnej bezpečnosti moderného systému.

Zobrazme si tento návrh vizuálne pomocou jednoduchého diagramu na Obr. 3. Na základe tejto myšlienky budeme vyvíjať jednotlivé súčasti celého riešenia.

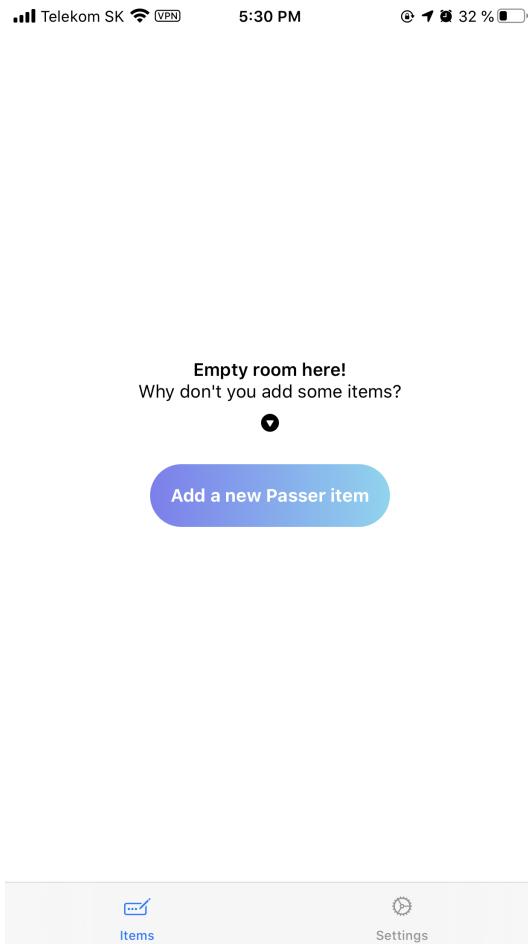


Obr. 3: Diagram implementácie nášho riešenia.

3.2 Aplikácia - Passer

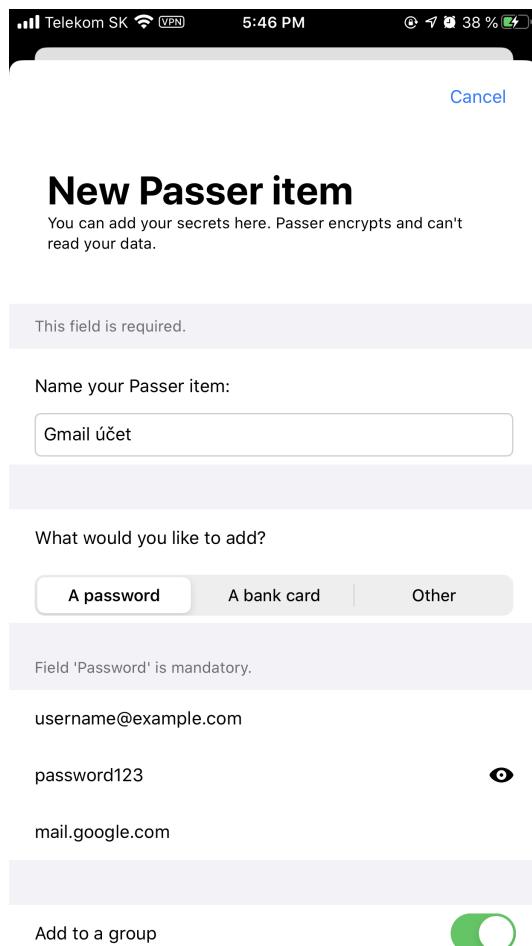
Na základe analyzovaných myšlienok sme vyvinuli aplikáciu nazvanú Passer. Je určená pre používateľov operačného systému iOS, teda ide o telefóny značky Apple.

Hned po naštartovaní Passera je používateľovi ponúknutá možnosť vytvoriť nové heslo:



Obr. 4: Úvodná obrazovka po prvom spustení.

Po kliknutí na tlačidlo *Add a new Passer item* si môže používateľ vytvoriť novú položku:



Obr. 5: Pridávanie novej položky.

Používateľ má k dispozícii mnohé atribúty. Môže dať názov svojej položke. Potom si vyberie o akú položku ide. Passer ponúka tri možnosti: heslo, banková karta, iné. V závislosti od výberu sa menia textové polia nižšie. Na Obr. 5 je zvolená možnosť *A password*, takže v tomto prípade ide o heslo. Používateľovi sa dostavia k dispozícii tri textové polia:

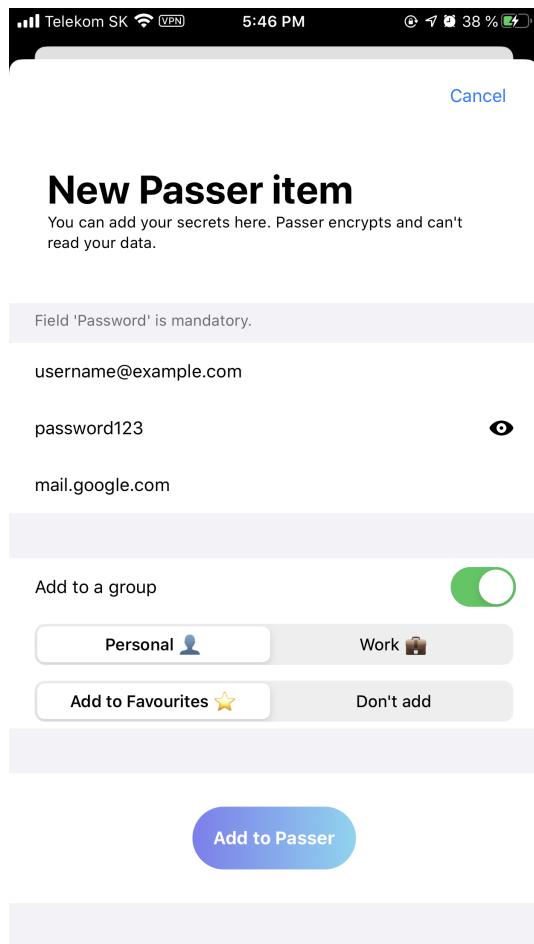
- Email alebo používateľské meno
- Heslo
- Webová stránka

Môžeme vidieť, že v textovom poli s heslom je napravo malá ikona s okom. Používateľ môže na ňu kliknúť a tým sa heslo skryje za znaky bodiek alebo zobrazí ako plaintext. V

kóde reprezentuje tlačidlo oka akýsi prepínač UI elementov vo frameworku SwiftUI jazyka Swift. V závislosti od stlačenia oka sa striedajú **TextField** a **SecureField**, ktoré zdielajú svoj obsah v jednej premennej. Na rozdiel od **TextField** [23], ktorý je len klasickým textovým poľom ukladajúcim jej obsah do nejakej premennej, **SecureField** [24] svoj obsah ukrýva.

Pri testovaní Passera sme pri tomto bezpečnom textovom poli spozorovali zaujímavé správanie. Po vytvorení snímky obrazovky sme zistili, že ani samotné bodky, ktoré nahradzajú plaintext v **SecureField** neboli na screenshote. Jednoducho zmizli.

Nasledujúci obrázok zobrazuje nasledujúce možnosti pri pridávaní hesla. Prepínač *Add to a group* umožňuje používateľovi zaradiť položku do skupiny. Passer poskytuje kategórie *Personal* (osobné) a *Work* (pracovné). Mimo tohto výberu si môže používateľ vybrať, či bude daná položka zaradená do jeho oblúbených.

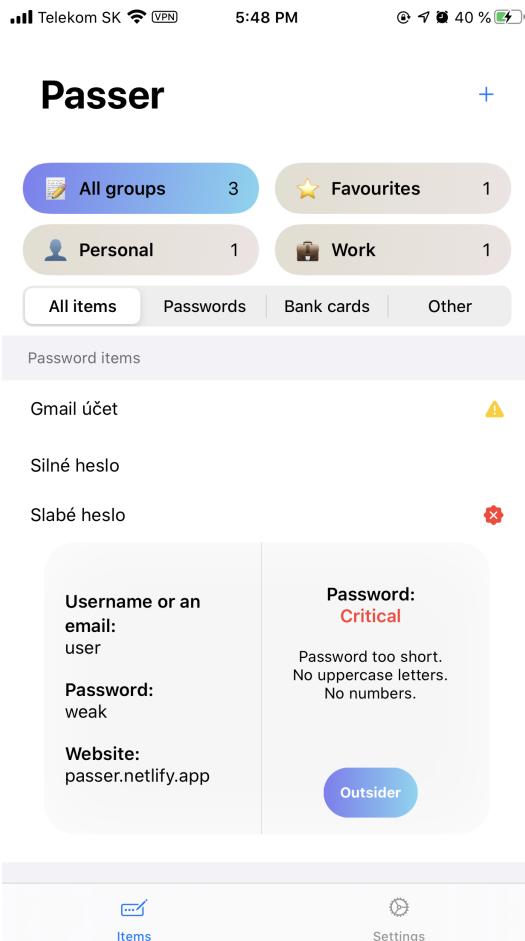


Obr. 6: Pridávanie novej položky do skupín.

Ked' je používateľ so všetkým spokojný, kliknutím na *Add to Passer* sa položka

úspešne pridá medzi ostatné položky.

Takýmto spôsobom sme vytvorili ešte ďalšie dve položky, všetky typu heslo. Po týchto úkonoch má používateľ v Passerovi už tri položky. Nasledujúci obrázok ukazuje úvodnú obrazovku Passera, ak má používateľ aspoň jednu položku:³



Obr. 7: Úvodná obrazovka s heslami.

Obr. 7 nám ukazuje tri heslá. Všimnime si ikonu žltého výkričníka pri položke *Gmail účet* a červenú ikonu kríža pri položke *Slabé heslo*. To je indikátor, ktorý hovorí, že heslo nie je dostatočne silné. Po kliknutí na jednu z položiek sa nám o nej rozbalia ďalšie informácie.

Screenshot ukazuje, že používateľ klikol na *Slabé heslo*, takže sa zobrazili aj dodatočné informácie. Vidíme všetky tri atribúty, ktoré sme vyplňali pri pridávaní novej položky do Passera. Napravo od týchto informácií vidíme červeným nápis *Critical* (kritické). To znamená, že heslo je kriticky slabé. Passer nám vypísal aj dôvody. Sú až tri: Heslo je

³obrazovka z Obr. 4 sa už neukáže po ďalšom spustení.

príliš krátke (minimálna dĺžka je osem znakov), heslo neobsahuje veľké písmená, heslo neobsahuje čísla.

V prípade žltého výkričníka je ohodnotenie *Vulnerable* (zraniteľné). Heslo, ktoré je dostatočne silné nemá pri sebe žiadny indikátor (viď. položka *Silné heslo*).

Logika vyhodnocovania je jednoduchá. Ak je heslo kratšie než osem znakov, alebo obsahuje iba malé písmená, iba veľké písmená alebo iba čísla, vyhodnotenie je *Critical*. Tu sa vráťme k podkapitole Sila hesla, ktorá vysvetluje prelomiteľnosť hesla. Tam sme hovorili, že ak aj použijeme kombináciu MP + VP + C (h3sL0), heslo je stále prelomiteľné pri offline útoku do sekundy. Preto je v tomto Passer vyhodnotenie prísne. Dĺžku hesla považuje za rovnako dôležitú ako kombináciu MP + VP + C. Keby bolo heslo h3sL0 v Passeri, stále by malo *Critical* ohodnotenie.

Ak heslu chýbajú čísla, alebo veľké, alebo malé písmená, ale inak všetko ostatné je v poriadku, vyhodnotenie je *Vulnerable*. Také heslo nie je vystavené veľkému nebezpečiu, ale stále je múdre predísť tomuto stavu.

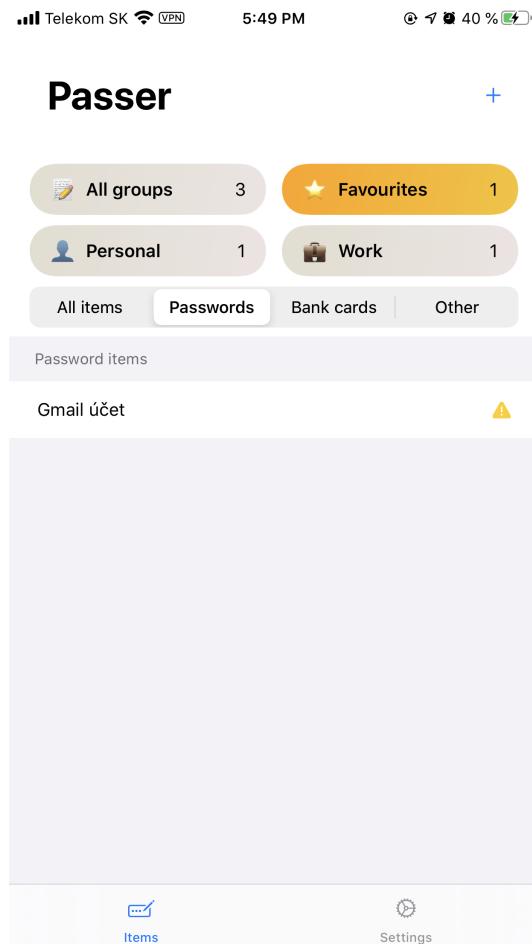
Rovnaké indikátory fungujú pri bankových kartách v Passeri. Samozrejme, tu nevieme hodnotiť silu hesla, ale vieme kontrolovať dátum expirácie karty. Ikona žltého výkričníka symbolizuje, že karta expiruje čoskoro. Ikona červeného kríža hovorí, že karta už expirovala. Absencia ikony znamená, že je všetko s kartou v poriadku.

Všimnime si horný panel na Obr. 7. Obsahuje rôzne filtre, na základe ktorých sa mení zobrazenie položiek nižšie.

Panel začína filtrom na skupiny. Počas procesu vytvárania novej položky sme ukázali, že vieme pridávať položky do skupín a/alebo medzi obľúbených položky. Predvolená selekcia je zobrazenie všetkých skupín (*All groups*).

Nižšie vidíme filter na typy položiek. Opäť sa odvoláme na proces vytvárania novej položky. V závislosti od voľby typu položky sa nám zobrazili korešpondujúce textové polia. Predvolená selekcia je zobrazenie všetkých typov (*All items*).

Po aplikovaní filtrov *Favourites* a *Passwords* vyzerá úvodná obrazovka s heslami nasledovne:



Obr. 8: Úvodná obrazovka s heslami po aplikovaní filtrov.

3.2.1 Outsider

Prejdime k hlavnému bodu celej aplikácie. Funkcionalita s názvom *Outsider* dokáže akúkoľvek položku poslať cez server na webstránku. Zároveň pre používateľa vytvorí jednorázovú verifikáciu, ktorá expiruje po dvoch minútach alebo po jej použití.

Proces začína kliknutím na položku v úvodnej obrazovke (viď Obr. 7). Môžeme vidieť, že napravo od detailov položky je prítomné tlačidlo *Outsider*. Po kliknutí je používateľ presmerovaný na nasledujúcu obrazovku:

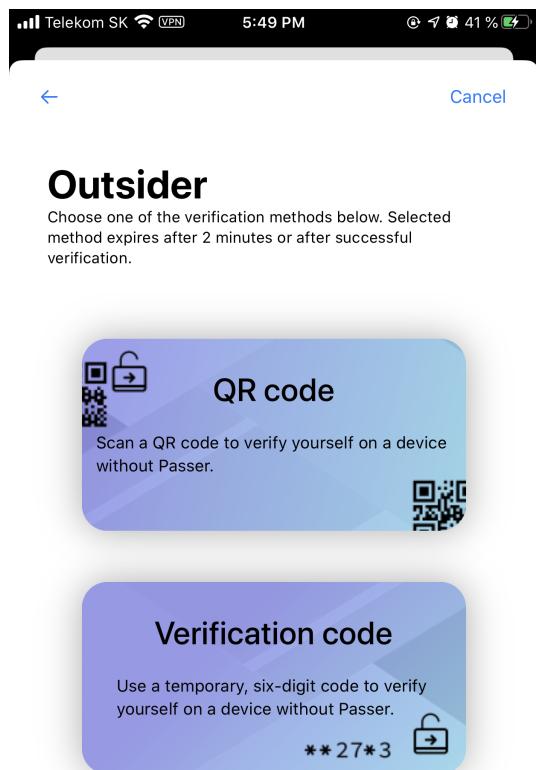


Obr. 9: Outsider - výber položiek.

Podľa toho, z akej položky klikneme na tlačidlo *Outsider* sa mení správanie na obrázku vyššie. V danom prípade vidíme, že je predvolene zvolený *Gmail účet*, pretože sme z tejto položky prešli na Outsider. Nedá sa označiť, čo symbolizuje aj šedá farba.

Na druhej strane, v dolnej polovici obrazovky môžeme pridať lubovoľný počet ďalších z našich položiek, ktoré sa odošlú spolu s *Gmail účet*. Vidíme možnosť *Select all*, ktorá označí všetky položky. Po kliknutí sa možnosť zmení na *Deselect all*, aby mohol používateľ označiť všetko v prípade, že by zmenil názor. Je dôležité povedať, že po kliknutí na *Deselect all* sa neodznačí položka, z ktorej bol používateľ presmerovaný na obrazovku Outsidera. Z tohto je zrejmé, že Outsider nikdy nepošle prázdne pole položiek na server. Pod spomínaným tlačidlom už vidíme jednotlivé položky, ktoré sú tiež tlačidlami. Ich stlačením sa daná položka pridá, respektíve odoberie zo zoznamu položiek na odoslanie.

Ak je používateľ spokojný s výberom, postupuje na ďalšiu obrazovku kliknutím na tlačidlo *Proceed*.

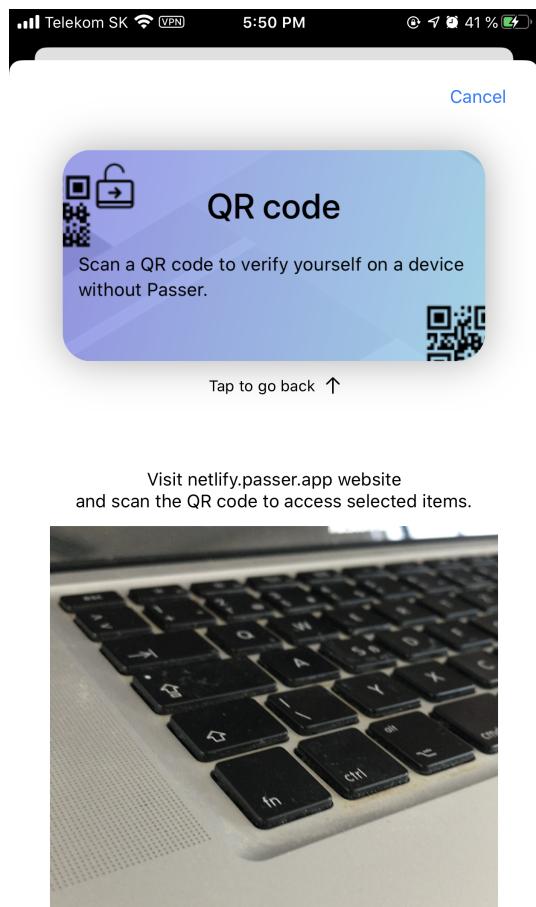


Obr. 10: Outsider - výber typu verifikácie.

Položky na odoslanie sú pripravené. Teraz nasleduje výber typu verifikácie.

Prvá z možností je nasnímanie QR kódu. Využijeme fakt, že Passer je určený pre zariadenia iPhone. Každý model má kameru. Môže byť použitá na nasnímanie QR kódu na webstránke.

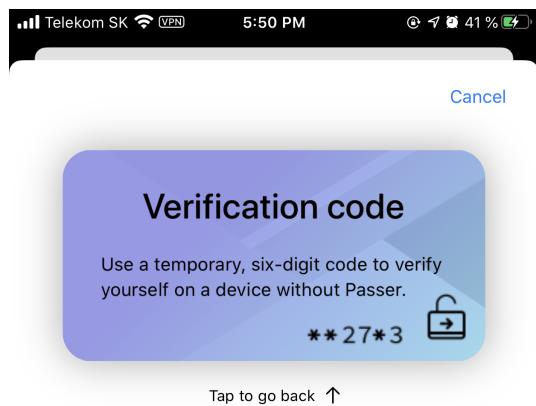
Druhá možnosť je použiť šestciferný kód, ktorý používateľ následne zadá na webstránke. Ak je kód správny, webstránka sprístupní odoslané heslá používateľovi.



Obr. 11: Outsider - QR kód.

Po kliknutí na možnosť verifikácie pomocou QR kódu sa používateľovi v dolnej časti zobrazí obraz, ktorý vidí kamera. Spolu s ním dostane inštrukcie, na akú stránku má ísť a čo má robiť.

Proces, ktorý sa deje na pozadí vysvetlíme až v sekcií Vzájomné interakcie, nakoľko tento proces je veľmi špecifický. A to z dôvodu, že táto verifikácia interaguje aj s web-stránkou, aj so serverom.



2 7 2 6 9 2

Visit netlify.passer.app website
and enter this code to access selected items.

Time remaining: 01:47

Obr. 12: Outsider - Šestciferný kód.

Pozrime sa na to, ako príde k tomu, že používateľ obdrží jednorázový, šestciferný kód. Po kliknutí na túto možnosť sa na pozadí vytvorí inštancia štruktúry `SixdigitAuth`, ktorá vyzerá v kóde nasledovne:

```
struct SixdigitAuth: Codable {  
    fileprivate let deviceID: String  
    let sixdigitCode: String?  
    let passwordItems: [PasswordItem]?  
    let bankCardItems: [BankCardItem]?  
    let otherItems: [OtherItem]?  
    let timestamp: Date?  
}
```

Atribút `deviceID` je generovaný iOS zariadením. Každé iOS zariadenie má jedinečné identifikačné číslo [25]. Tento atribút je dôležitý a to preto, aby na serveri nemal jeden používateľ dva rôzne šestciferné kódy. Ak počas dvoch minút používateľ zopakuje proces generovania kódu, na serveri nový kód prepíše ten starý. Toto opatrenie zvyšuje bezpečnosť. Keby používateľ neustále generoval nové a nové kódy (toto všetko v rozmedzí dvoch minút), tak by existovalo oveľa viac možností, ako sa na webstránke dostať k súkromným údajom daného používateľa. Týmto ošetrením má používateľ na serveri najviac jednu verifikáciu, ktorá je správna.

`sixdigitCode` je `String`, ktorý vznikol pospájaním šiestich, náhodne vygenerovaných čísel Passerom. Atribúty `passwordItems`, `bankCardItems` a `otherItems` sú polia, ktoré majú obsahovať položky rôznych typov na odoslanie. Polia môžu byť prázdne. Posledný atribút `timestamp` vytvorí časovú pečiatku, kedy bola táto inštancia vytvorená. Podľa toho Passer vie, koľko času ostáva používateľovi do konca platnosti. Zobrazuje ho v Obr. 12 ako výsledok rozdielu `timestamp` a aktuálneho času. Túto kalkuláciu Passer vykonáva každú sekundu.⁴

Nad touto štruktúrou vykoná Passer serializáciu do dátovej štruktúry JSON, aby ju server vedel čítať. JSON je následne poslaný na server. V tomto momente Passer čaká na odpoveď. Ak je pozitívna, môže považovať vygenerovaný kód za použiteľný. Až v tomto momente je zobrazený používateľovi na obrazovke. Detailnejší rozbor poskytujeme v Vzájomné interakcie.

3.3 Webstránka

Webstránka sama o sebe nemá zmysel. Slúži výhradne na prístup k dátam poslaným pomocou Outsidera v aplikácii Passer.

Úvodná obrazovka ponúka používateľovi verifikovať sa buď pomocou šestciferného kódu, alebo pomocou QR kódu.

Počas zadávania šestciferného kódu sa jednotlivé číslice ukladajú do poľa. Keď je naplnené (dĺžka poľa musí byť 6), prebehne serializácia do JSON štruktúry a kód sa posielá na server. Ten ho skontroluje a vráti odpoveď webstránke. Opäť: bližší rozbor (vrátane skenovania QR kódu) v Vzájomné interakcie.

Po úspešnej verifikácii je používateľ presmerovaný do sekcie, kde sa mu zobrazia všetky položky z Passera, ktoré si cez Outsider poslal. Z tejto obrazovky je možné skopírovať jednotlivé atribúty položiek (tlačidlo *COPY*). Pri položkách typu heslo je možnosť aj priamo prejsť na webstránku uvedenú v položke pomocou tlačidla *VISIT WEB*.

⁴častejšie netreba, nakoľko čas sa zobrazuje v minútach a sekundách.



Please verify yourself before accessing your
passwords on this device.

6-digit code

QR code



Obr. 13: Webstránka - Úvodná obrazovka.



Hello, here are your items.

Gmail účet

Username or an email:	username@example.com	COPY
Password:	COPY
Website:	mail.google.com	VISIT WEB

Silné heslo

Username or an email:	username	COPY
Password:	COPY
Website:	website.org	VISIT WEB

Obr. 14: Webstránka - Zobrazenie Passer položiek.

3.4 Server

Server funguje na základe jedného jednoduchého skriptu, ktorý je napísaný v jazyku Python, konkrétnie hovoríme o frameworku Flask.

Logika Flaska je celkom priamočiara. Pre rôzne url endpointy vykoná inú funkciu. Základom je anotácia `@app.route('/pripona/url')` [26], kde jedným z parametrov do funkcie `route()` je retazec, ktorý symbolizuje url endpoint, pre ktorý sa má funkcia vykonať. Definícia funkcie a jej telo nasledujú hned za `@app.route()`.

Pre ukážku povedzme, že server funguje na doméne `https://flask.website.com`. Potom, ak existuje nasledujúci kód:

```
@app.route('/test')
def function():
    currentDate = datetime.now().strftime('%d/%m/%Y %H:%M:%S')
    return 'Hello World! ' + currentDate
```

Tak po zadaní url `https://flask.website.com/test` sa zobrazí text

Hello World! 26/5/2020 22:55:32

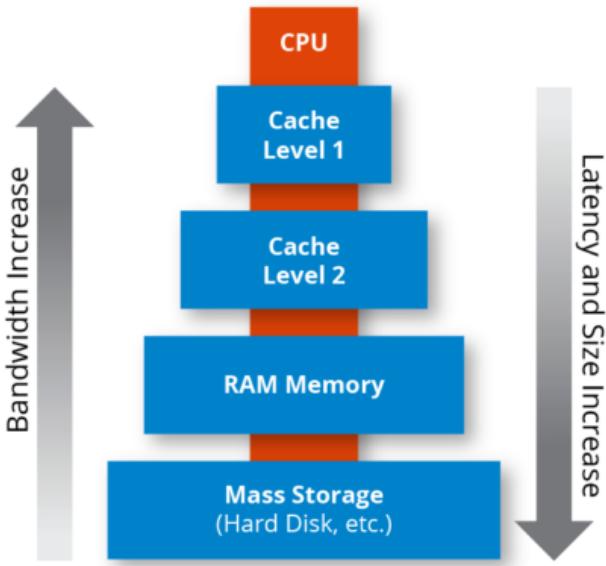
(čas, keby sme zadali url v čase písania tohto textu).

Všetky dátá, ktoré server uchováva sú dočasné. Preto je rozumné použiť pamäť RAM. Ešte lepšie je použiť cache, ktorá je súčasťou RAM a ponúka ešte rýchlejší prístup k dátam. Takzvaný „memory caching“ je technika, kedy aplikácie dočasne ukladajú svoj obsah do časti pamäte RAM, ktorá sa volá cache [27].

Ked' sa aplikácia pokúša čítať dátá, najprv skontroluje, či nie sú v cache. Ak nie, prečíta ich z databázy, čo trvá dlhšie. Následne ich zapíše do cache, aby najbližšie čítanie daného záznamu bolo rýchlejšie. Tým, že cache má limitovaný ukladací priestor, je dôležité vymyslieť stratégiu, ako staré dátá mazat a nahradzať novými.

Cache je vysoko favorizovaná najmä ak ide o časté a opakovane úkony. Ak pristupujeme k určitým dátam na pravidelnej báze, je lepšie uchovávať ich v cache, aby nemusela byť prehľadávaná celá databáza, respektívne súbory vo filesystéme.

Opísali sme ako sa cache používa v najčastejších prípadoch. My sme sa rozhodli pozerať na cache trochu inak. Náš server nemá žiadnu databázu. Ani sa na žiadnu nenapája. Jediné, čo má, je cache. Každý záznam v nej je jedinečný a po jeho použití zmizne. Takže rovnako tu nejde o opakovane úkony k rovnakým dátam.



Obr. 15: Porovnanie rôznych typov pamäte v závislosti od rýchlosťi a kapacity.

Zopakujeme ešte raz fakt, že všetky dátá servera sú dočasné. Preto považujeme za kontraproduktívne a neefektívne vytvoriť databázu a pristupovať k nej pri výbere dát. Cache je rýchla, jej veľkosť je postačujúca. Komplexita je minimálna. Toto je dôležitý faktor, nakoľko komplexita je najväčší nepriateľ bezpečnosti. V Pythone je jej implementácia jednoduchá. Dátá nemusíme šifrovať. Tým nám odpadá problém bezpečného ukladania šifrovacích kľúčov.

Na implementáciu cache v Pythone budeme používať Werkzeug knižnicu [28]. Na začiatku si vytvoríme jej inštanciu `cache = SimpleCache()`. Následne ju môžeme používať. V kóde neskôr používame jej tri najčastejšie metódy a to: `cache.set()` (pri pridávaní nových dát), `cache.get()` (pri verifikácii) a `cache.delete()` (po úspešnej verifikácii).

3.5 Vzájomné interakcie

3.5.1 Passer - Server (šestciferný kód)

Teraz ukážeme príklad, ako server spracováva HTTP POST request od aplikácie Passer. Aplikácia odošle nový, šestciferný verifikačný kód spolu s heslami, ktoré používateľ cez Outsider odosla. Inými slovami, pošle JSON štruktúru `SixdigitAuth` (spomíname v Aplikácia - Passer). Štruktúra môže vyzerat nasledovne:

```
{
    "deviceID": "F19494AE-555C-49FF-87DD-D94CA62D9634",
    "sixdigitCode": "791509",
```

```

"passwordItems": [
    {
        "password": "password123",
        "id": "595C185D-DDF9-4BEA-A722-D383C75E5B33",
        "username": "username@example.com",
        "favourites": true,
        "group": 1,
        "itemname": "Gmail účet",
        "url": "mail.google.com"
    },
],
"bankCardItems": [],
"otherItems": [],
"timestamp": 612273137.84391904
}

```

V tejto ukážke si používateľ vybral možnosť verifikácie pomocou šestciferného kódu a posiela jedno heslo - *Gmail účet*. Server tieto dát očakáva v tomto bloku kódu:

```

@app.route('/sixdigit', methods=['POST'])
def processSixDigitFromApp():
    incomingData = request.get_json()

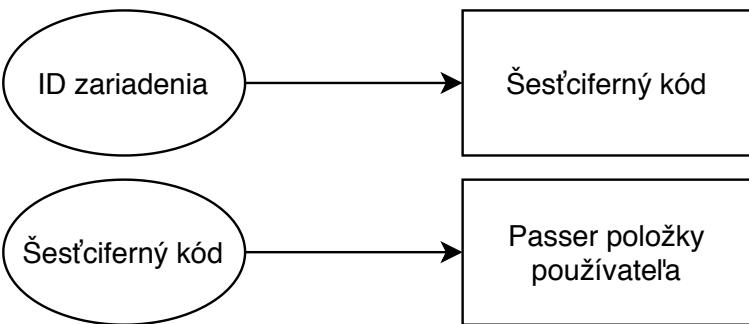
    deviceID = incomingData['deviceID']
    sixdigitCode = incomingData['sixdigitCode']
    passwordItems = incomingData.get('passwordItems')
    bankCardItems = incomingData.get('bankCardItems')
    otherItems = incomingData.get('otherItems')

```

Server si uloží jednotlivé atribúty JSON štruktúry do premenných. Všimnime si, že JSON atribút **timestamp** server vôbec nespracuje. Hovorili sme, že **timestamp** slúži len pre počítanie zostávajúceho času pre aplikáciu Passer. Tento čas je avšak orientačný. Medzi vytvorením inštancie **SixdigitAuth** (vtedy vzniká časový odtlačok) a zapísaním verifikácie do Flask cache existuje určité časové oneskorenie. Preto by bolo nepresné, aby server uvažoval **timestamp** ako čas do expirácie verifikácie.

Po spracovaní JSON štruktúry nasleduje kontrola duplicit. Nasledujúci diagram uka-

zuje logiku ukladania dát do cache servera (pri šestcifernom kóde):



Obr. 16: Ukladanie dát do cache servera (šestciferný kód).

Šesťciferný kód je namapovaný na ID zariadenia, ktorý je klúč. A následne daný kód slúži ako klúč v druhom zázname, na ktorý sú namapované Passer položky používateľa.

Dôvod, prečo potrebujeme až dve mapovania vysvetlíme. Potrebujeme zaručiť, aby sa dvom rôznym používateľom (teda dvom rôznym ID zariadenia) nemohol vygenerovať rovnaký kód. Rovnako potrebujeme, aby jeden používateľ (teda jedno ID zariadenia) nemal na sebe viacero jednorázových kódov (spomíname v Aplikácia - Passer).

Pred zápisom novej verifikácie do cache najprv skontrolujeme, či nenastal jeden z dvoch konfliktných stavov, ktoré sme práve opísali.

Ako prvé zistíme, či v cache existuje klúč, ktorý by sa zhodoval so šestciferným kódom, ktorý práve prišiel. Ak existuje, server vráti kód 409, čo znamená CONFLICT [29]. Passer na to zareaguje opakováním celej operácie. Čo obnáša zaslanie SixdigitAuth štruktúry v JSON formáte s novým kódom.

```
if cache.has(sixdigitCode):  
    return 409
```

Druhá kontrola spočíva v tom, či pre daného používateľa už existuje nejaký šesťciferný kód. Kontrolujeme, či v cache existuje klúč, ktorý by sa zhodoval s deviceID, teda s ID zariadenia. Ak existuje, musíme vymazať obe mapovania.

```
if cache.has(deviceID):  
    userOldVerifData = cache.get(deviceID)  
    cache.delete(deviceID)  
    cache.delete(userOldVerifData)
```

Po týchto operáciách môže prebehnúť samotný zápis do cache. Pomocou `cache.set()` nastavíme mapovanie klúč:hodnota a povieme, ako dlho má tento záznam existovať. Obe

mapovania (`deviceID:sixdigitCode` a `sixdigitCode:Passer` položky) budú trvať po dobu dvoch minút.

```
cache.set(deviceID,sixdigitCode,timeout=2*60)
cache.set(sixdigitCode,[passwordItems, bankCardItems, otherItems,
    deviceID],timeout=2*60)
return 'server: ok', 201
```

Po úspešnej operácii Passer zobrazí používateľovi šestciferný kód so zostávajúcim časom platnosti a inštrukciami.

3.5.2 Webstránka - Server (šestciferný kód)

Používateľ už obdržal kód, ktorý môže použiť na webstránke na prístup k svojim položkám. Po jeho zadaní sa v JavaScript kóde webstránky spustí HTTP POST request na server. Posiela mu kód zadaný používateľom a čaká, ako odpovie server. Server sa snaží získať dátá namapované ku klúču (`data = cache.get(sixdigitTyped)`). Klúč je používateľom zadaný šestciferný kód.

Ak by kód neboli správne zadaný, klúč v cache by sa nenašiel. Tým pádom by `data` boli `None`. Podľa toho sa mení správanie kódu nižšie:

```
if data != None:
    cache.delete(sixdigitTyped)
    cache.delete(data[-1])
    response[sixdigitTyped] = data
return response
return 'Wrong code'
```

Predtým, než vráti dátá späť webstránke, server vymaže záznamy v cache. Najskôr záznam pod klúčom šestciferného kódu, potom záznam pod klúčom ID zariadenia (posledný člen zoznamu `data`). Následne vráti dátá.

Webstránka presmeruje používateľa na podstránku `.../passwords.html` a to nasledovne: `window.location.replace("passwords.html")`. Predtým ale potrebuje určitým spôsobom odovzdať dátá, ktoré práve od servera získala podstránke. JavaScript ponúka mnoho riešení, jedno z nich je local storage [30].

Ide o schopnosť ukladať dátá v tvare klúč:hodnota (podobné, ako cache na serveri) lokálne v prehliadači. Pomocou `window.localStorage.setItem("serverData",xhr.responseText)` uložíme do local storage odpoveď zo servera pod klúč `serverData`. Následne sme schopní dátá z local storage vybrať `const serverData = window.localStorage.getItem("serverData")`

a pracovať s nimi. Hneď po načítaní do premennej `serverData` je potrebné vyčistiť local storage, inak by boli tieto dátá v pamäti aj počas ďalších sessions, ako uvádzia [30]. Jednoduchým kódom `localStorage.clear()` je náš problém vyriešený.

3.5.3 Passer - Server - Webstránka (QR kód)

Ako sme spomínali v sekcií Aplikácia - Passer, verifikácia pomocou QR kódu je špecifická. Počas nej prebieha interkacia so všetkými troma komponentmi (v úvodzovkách) naraz.

Hneď po načítaní webstránky sa vygeneruje takzvaná session id: `let sessionID = Date.now().toString(36) + Math.random().toString(36).substr(2, 5)`. JavaScript používa aktuálny čas pre dosiahnutie jedinečnosti. To nemusí byť dostačujúce, preto sa k výslednému retazu priprája ešte náhodný retazec pomocou funkcie `Math.random()`. Táto session id sa zakóduje do vygenerovaného QR kódu. Je zrejmé, že QR kód je zakaždým jedinečný, nakoľko obsahuje jedinečný session id retazec.

Generovanie kódu je vďaka knižnici QRCode.js nasledovné:

```
function generateQRcode() {
    var qr = new QRCode(document.getElementById("qr-image"), {
        width: 120,
        height: 120,
        correctLevel : QRCode.CorrectLevel.L
    })
    qr.makeCode(sessionID)
}
```

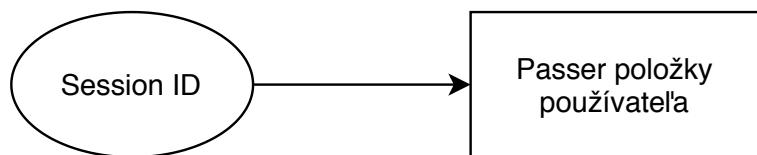
V poslednom riadku si môžme všimnúť, že vytvárame QR kód s príslušnými dátami, teda s našim session id. To znamená, že ak niekto náš QR kód naskenuje, získa toto session id.

Tento bod je kľúčový. Pomocou session id sa vie používateľ s Passerom verifikovať, že je to práve on, kto kód naskenoval. Po naskenovaní v Passeri aplikácia vytvorí štruktúru, podobnú `SixdigitAuth`. Vyzerá takto:

```
struct QRAuth: Codable {
    let sessionID: String
    let passwordItems: [PasswordItem]?
    let bankCardItems: [BankCardItem]?
    let otherItems: [OtherItem]?
```

}

`sessionID` je session id naskenovaná z QR kódu. K nemu pridáme polia Passer položiek, ktoré sa spolu so `sessionID` odošlú na server rovnakým spôsobom, ako pri šestcifernom kóde. Pri QR verifikácii je len jedno mapovanie v cache servera:



Obr. 17: Ukladanie dát do cache servera (QR kód).

Medzičasom webstránka cyklicky kontroluje server, či už existuje v cache kľúč s daným session id. Aby sa zabránilo preťaženiu, robí tak iba raz za sekundu. Akonáhle webstránka dostane pozitívnu odpoveď a dáta používateľa, prestane kontrolovať server a presmeruje používateľa na podstránku `.../passwords.html`. Kontrola prítomnosti verifikácie v cache je založená na rovnakom princípe ako pri šestcifernom kóde, takže hlbšiu analýzu tu nie je potrebné robiť.

Ukážeme ešte spôsob kontrolovania servera na strane webstránky:

```
setInterval (function() {
    let xhr = new XMLHttpRequest()
    xhr.open("POST", "https://api-passenger.herokuapp.com/verifyQRfromwebsite", true)
    xhr.setRequestHeader("Content-Type", "application/json")
    xhr.onreadystatechange = function () {
        if (xhr.readyState === 4 && xhr.status === 201) {
            window.localStorage.setItem("serverData", xhr.responseText)
            window.location.replace("passwords.html")
        }
    }
    let data = { "sessionID": sessionID }
    let jsonData = JSON.stringify(data)
    xhr.send(jsonData)
}, 1000)
```

Celý `XMLHttpRequest()` objekt sa spúšťa až exekúciou riadku `xhr.send()` v dolnej časti kódu. Podobná logika requestov sa vykonáva aj v jazyku Swift, na ktorom je postavený Passer.

Toto sú všetky interakcie našich troch komponentov, ktoré zhmotnili ideu celej práce. Týmto spôsobom sme docieliли prístup používateľa k svojim dátam aj mimo Passera.

3.6 Bezpečnosť

Na prvom mieste je dôležitá bezpečnosť. Riešenie, ktoré vyvijame v tejto práci musí byť zabezpečené, aby citlivé dáta používateľov neunikli. Najmä, ak ide o password manager.

Prvým nepriateľom bezpečnosti je komplexnosť [31]. Nehovoríme o tom, že softvér má byť malý. Aj veľké systémy vedia byť v podstate jednoduché.

Systém je len tak bezpečný ako jeho najzraniteľnejšia časť [31]. Preto musíme rovnako kvalitne zabezpečiť každý komponent a modul. Ak existuje trhlina v systéme, útočník ju využije. Je to jednoduchšie, ako prelomiť tú časť systému, ktorá je kvalitne zabezpečená.

V našom riešení sa budeme pozerať na bezpečnosť jednotlivých modulov.

- Ako chráni Passer citlivé dáta v čase, keď aplikácia nie je používaná
- Ako je zabezpečený prenos dát medzi jednotlivými komponentami
- Ako sú chránené komponenty server a webstránka

3.6.1 Passer

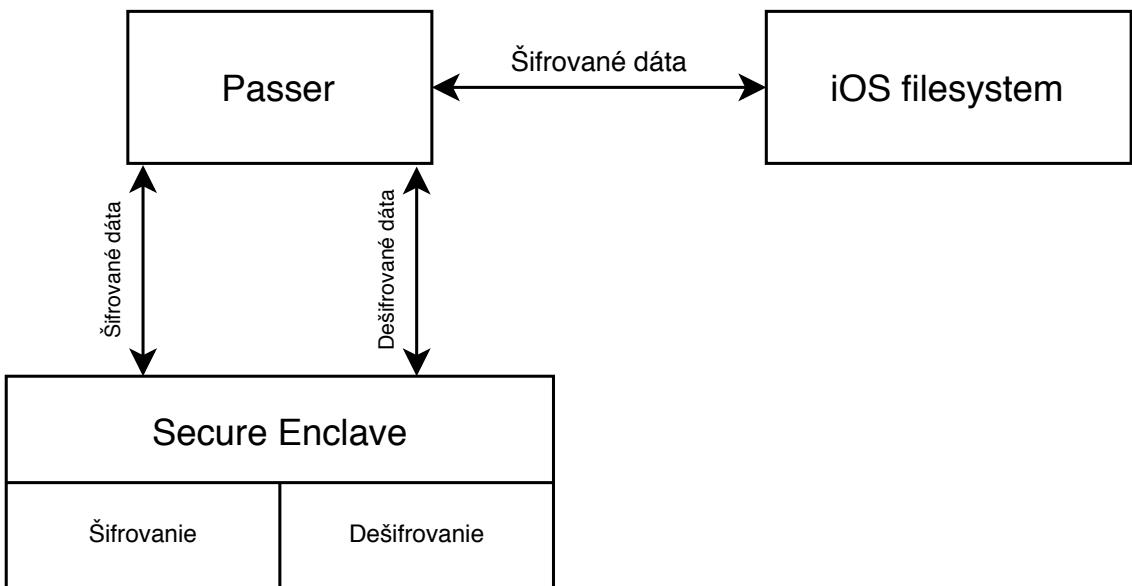
Passer potrebuje chrániť dáta, ktoré sú uložené vo filesystéme iOS. Kedže nechceme, aby používateľ strátil všetky dáta po reštarte aplikácie, potrebujeme ich ukladať mimo aplikáciu. Riešením je šifrovanie. Ak zašifrujeme dáta počas ukladania do súboru, útočník ich nebude môcť rozlúštiť, aj keby sa ku nim dostal.

Z rôznych dostupných šifrovacích algoritmov sme vybrali AES. Táto symetrická šifra používa len jeden klúč. Pomocou neho šifruje, ale aj dešifruje dáta. Vzniká nám nový problém: teraz musíme chrániť aj súbory (pred poškodením, odcudzením), aj klúč k ich zašifrovaniu. Na ukrytie klúča sme využili architektúru, ktorú obsahujú iPhone smartfóny s technológiou Face ID alebo Touch ID: Secure Enclave.

Secure Enclave (ďalej SE) je kus hardvéru, ktorý žije spolu s telefónom pod jednou konštrukciou. Má vlastný operačný systém a funguje nezávisle od iOS. Jeho úloha je chrániť citlivé údaje telefónu, najmä kryptografické klúče. Taktiež je používaný na bezpečnostné procedúry ako verifikácia biometrickým odtlačkom [32] (Touch ID/Face ID).

Má vlastný, šifrovaný firmvér, pamäť, disk a šifrovanie na úrovni hardvéru. Tento komponent využijeme na ukrytie AES klúča. Keďže sa jedná o dve komunikačné strany, musí prebehnúť asymetrická výmena klúčov, aby si iOS a SE mohli medzi sebou vymieňať dát bezpečne.

iOS nekomunikuje s SE priamo. Využíva takzvaný „mailbox“ (poštová schránka) systém. Aplikácia umiestní dátá a inštrukcie, čo sa má s nimi vykonať na špeciálne pamäťové miesto (mailbox). Odtiaľ si ich SE vezme, vykoná operácie a výsledok vloží späť do mailboxu, odkiaľ si ich aplikácia vezme.



Obr. 18: Interakcia Passera s iOS a Secure Enclave.

V Secure Enclave vieme vytvárať iba 256 bitové, súkromné, elliptic curve klúče [33]. Tieto klúče vedia byť použité na Diffie-Hellmanovu elliptic curve (ECDH) výmenu klúčov. A túto operáciu vieme použiť na symetrické šifrovanie. Tieto klúče vieme vytvárať výhradne iba v SE. Nie je možné vkladať, či vyberať už existujúce klúče. Nemožnosť týchto úkonov je základom celej bezpečnosti SE.

Vytvorenie súkromného, elliptic curve klúča vyzerá v jazyku Swift nasledovne:

```
let privateKey = SecKeyCreateRandomKey(attributes as CFDictionary,
                                         &error)
```

Kde **attributes** je **dictionary** atribútov súkromného klúča, kde definujeme jeho vlastnosti (256-bit elliptic curve a pod.). Tento klúč vytvárame priamo v Secure Enclave. Do

premennej `privateKey` ide iba referencia skutočného kľúča [33], takže zdroj je uchovaný v SE. Preto nie je možné získať z `privateKey` jeho dátu, takže tu je bezpečnosť zachovaná.

Z `privateKey` vieme získať verejný kľúč a to nasledovne:

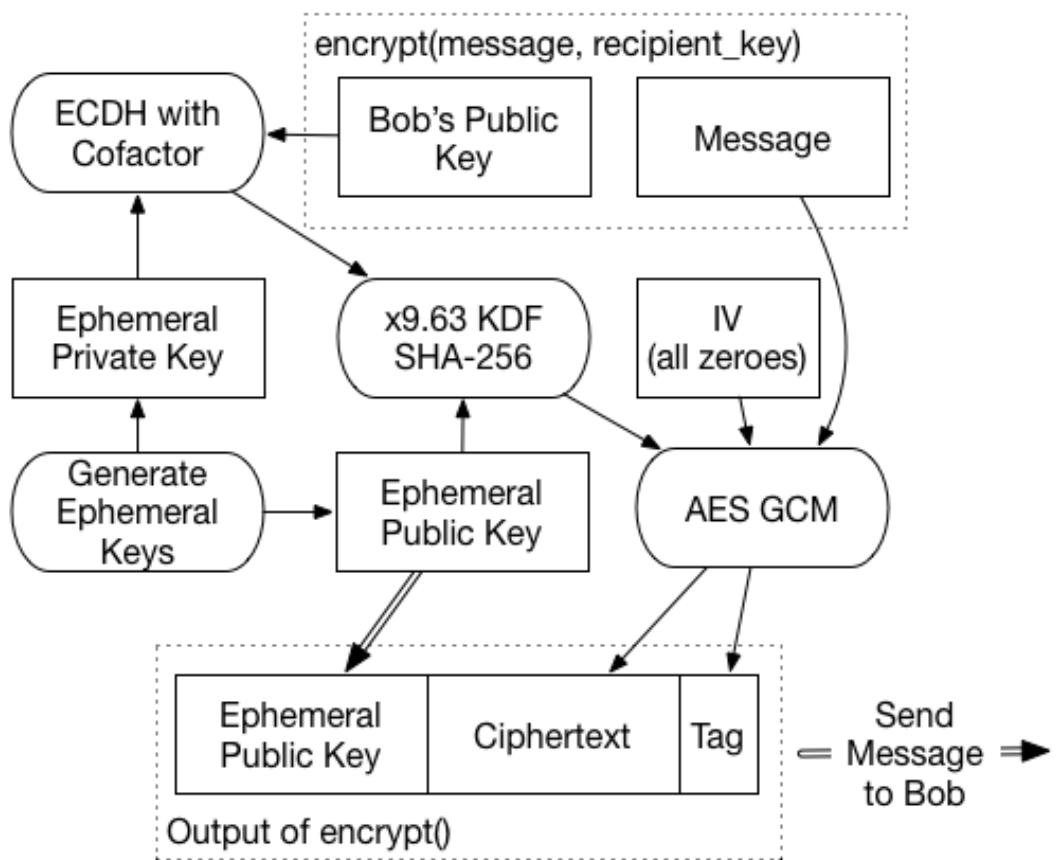
```
let publicKey = SecKeyCopyPublicKey(privateKey!)
```

Máme vytvorený súkromný aj verejný kľúč, ktorý patrí Secure Enclave. Môžeme začať so šifrovaním.

Symetrické šifrovanie prebieha podľa algoritmu ECIES [34]. V kryptografii sa komunikácia medzi dvoma stranami demonštruje ako komunikácia medzi Alicou a Bobom. Bob bude v našom prípade Secure Enclave, Alice bude Passer. Teda:

- Bob vlastní súkromný kľúč x a z neho vytvorí verejný kľúč g^x .
- Alice získa Bobov verejný kľúč g^x .
- Alice vygeneruje súkromný, efemérny kľúč y a z neho verejný, efemérny kľúč g^y .
- Alice vypočíta symetrický kľúč k pomocou key derivation function: $k = KDF(g^{xy})$.
- Alice získa šifrovaný text c zo správy m použitím symetrického kľúča k a algoritmu AES nasledovne: $c = AES(k; m)$.
- Alice pošle šifrovanú správu spolu s efemérnym kľúčom Bobovi.
- Bob extrahuje z c efemérny verejný kľúč.
- Bob vypočíta symetrický kľúč k pomocou key derivation function: $k = KDF(g^{xy})$
- Bobov kľúč k je rovnaký, ako Alicin pri šifrovaní.
- Bob dešifruje správu c a získa otvorený text: $m = AES(k; c)$.

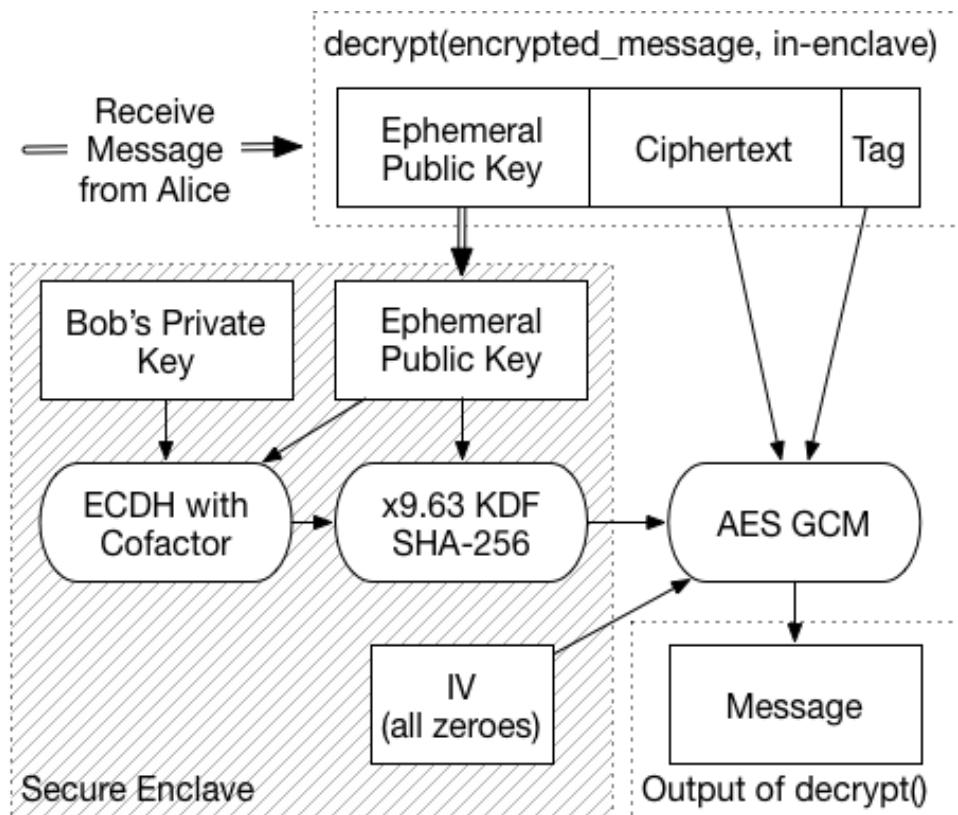
Zobrazme si tento proces vizuálne. Pri šifrovaní



Apple kSecKeyAlgorithmECIESEncryptionCofactorX963SHA256AESGCM
(encrypting)

Obr. 19: Proces šifrovania podľa algoritmu ECIES.

vznikajú efemerálne klúče. Efemerálny súkromný klúč spolu s verejným klúčom patriacim Secure Enclave na základe ECDH vytvoria takzvaný „shared secret”. Ten je pomocou KDF zmenený na symetrický klúč. Klúč vchádza do AES spolu so správou a inicializačným vektorom. Výsledok je zašifrovaný text spojený s efemerálnym verejným klúčom a tagom. Tento „balík” je odoslaný do SE.



Apple kSecKeyAlgorithmECIESEncryptionCofactorX963SHA256AESGCM
(decrypting)

Obr. 20: Proces dešifrovania podľa algoritmu ECIES.

SE spracuje to, čo prišlo. Extrahuje efemerálny verejný klúč. Použije svoj súkromný klúč a efemerálny verejný klúč na získanie verejného tajomstva. Pomocou KDF získa symetrický klúč. Ten spolu s inicializačným vektorom, šifrovaným textom a tagom vstupuje do AES. Algoritmus dešifruje text na pôvodnú správu. Jediné čo SE vráti do mailboxu je táto dešifrovaná správa.

V skratke: pri šifrovaní používame verejný klúč SE, pri dešifrovaní zase súkromný klúč SE. Efemerálne klúče sa generujú kvôli dohode o výmene klúčov (ECDH).

V zdrojovom kóde Passera existuje trieda `Vault`. V nej sú všetky položky používateľa počas behu programu: teda, v plaintexte. Po pridaní alebo vymazaní položky sa vo `Vault` zavolá metóda `vaultUpdate()`, ktorá vykonáva operácie s dátami. Najprv aktualizuje polia Passer položiek. Potom ich serializuje do JSON štruktúry. Potom túto štruktúru zašifruje pomocou `vaultEncrypt()`. Táto metóda vracia

```
return SecKeyCreateEncryptedData(publicKey!, algorithm, dataToEncrypt as
CFData, &error) as Data?,
```

čo je zašifrovaný text. Na šifrovanie je použitá vstavaná metóda Swiftu `SecKeyCreateEncryptedData()`, ktorá ako vstup očakáva verejný klúč SE, zvolený šifrovaný algoritmus (v našom prípade AES), nezašifrovanú správu (JSON štruktúra s Passer položkami) a smerník na objekt riadenia chýb, ktorý kontroluje chyby a oznámi ich do konzoly v prípade zlyhania.

Pri najbližšom spustení aplikácie je najprv od používateľa vyžiadaná biometrická autentizácia. Po úspešnom verifikovaní sa zavolá metóda `vaultDecrypt()`, ktorá dešifruje dátu a vráti ich:

```
return SecKeyCreateDecryptedData(privateKey!, algorithm, dataToDecrypt!
                                as CFData, &error) as Data?
```

Metóda `SecKeyCreateDecryptedData()` očakáva súkromný klúč SE, zvolený šifrovaný algoritmus (v našom prípade AES), zašifrovanú správu a smerník na objekt riadenia chýb.

Dáta sú dešifrované. JSON objekt vieme deserializovať a naplniť z neho `Vault`. Úvodná inicializácia je hotová. Používateľovi sa zobrazí úvodná obrazovka a vidí svoje Passer položky.

Pri úvodnej biometrickej autentizácii nešlo o šifrovanie/dešifrovanie/výmenu klúčov. Je to len podmienka k tomu, aby mohlo dešifrovanie vôbec začať. Ak používateľ nemá žiadne položky, tento krok je preskočený. Je zbytočné autentizovať vstup do niečoho, kde nič nie je.

3.6.2 Server a webstránka

Server a webstránka sú komponenty, ktoré sú neustále v sieti internet, takže sú kedykoľvek dosiahnuteľné. No najmä, dostať k nim dátu znamená prenášať ich po sieti. Preto je dôležité, aby bol protokol HTTP zabezpečný a Eva⁵ nevedela kanál čítať. Z toho dôvodu sú oba komponenty zabezpečené HTTPS protokolom, ktorý zaobalauje vrstvu HTTP do TLS vrstvy. Tá šifruje informácie, ktoré cestujú z Passera na server, zo servera na webstránku, z webstránky na server.

Server nemá žiadne dátá vo filesystéme. Teda, narozenie od Passera nemusíme riešiť problematiku ukladania a výmeny klúčov. Jediné dátá, ktoré server uchováva sú v cache. Lenže tie musia byť v plaintexte, pretože sa používajú v reálnom čase pri rôznych operáciách.

Webstránka je na tom podobne. Prijaté dátá od servera po úspešnej verifikácii používateľom sú rovnako v plaintexte, pretože ich potrebujeme zobraziť na obrazovke. Taktiež

⁵pri spomínanom Bobovi a Alici je Eva chápana ako útočník, ktorý chce spojenie narušiť alebo čerpaať z neho citlivé informácie

musí byť používateľ schopný ich skopírovať. Keby skopíroval šifrovaný text, používanie jeho údajov by bolo zbytočné, nakoľko by boli šifrované.

Aby sme po sebe nezanechali stopy, vždy po získaní dát od servera vyčistíme local storage. Inak by boli dátá dohľadateľné v každej session, pokiaľ je otvorený prehliadač na cudzom zariadení.

4 Hrozby

V minulej kapitole sme do hĺbky ukázali implementáciu nášho riešenia. Máme aplikáciu Passer, ktorá dokáže uchovávať citlivé informácie používateľa a sprístupniť ich bezpečne na cudzom zariadení.

To, či je systém naozaj bezpečný sa ukáže až po tom, ako vnikne do terénu [31]. Existuje mnoho moderných šifrovacích algoritmov, ktoré môžu byť oveľa lepšie než tie súčasné. Avšak napriek tomu sa masovo nepoužívajú, pretože to, či je niečo bezpečné sa vždy ukáže až časom. Preto sú dnešným svetovým štandardom algoritmy ako AES, či SHA. Pritom AES bolo akceptované už pred 19-timi rokmi a SHA (256) pred 18-timi. Ale napriek tomu sa tieto algoritmy používajú. Boli overené časom. Sú bezpečné, ak sú v správnych rukách a správne použité.

V našej implementácii sa spoliehame na Secure Enclave. Ak by sa útočník vedel nabúrať do SE, dostal by prístup ku všetkým klúčom vo vnútri. A keby útočník prelomil SE iPhone-u, ktorý má nainštalovaný Passer, útočník by vedel dešifrovať všetky Passer dátá používateľa.

Ak by chcel útočník ukradnúť dátá v cache servera, musel by nájsť stroj, na ktorom je spustený skript. Z neho by musel vybrať pamäť RAM a zamraziť ju tekutým dusíkom (Cold boot attack [35]). Tak by dátá ostali v cache po ľubovoľne dlhú dobu.

Iný spôsob, ako ukradnúť dátá je brute-force útok. Na webstránke má útočník dve minúty na to, aby spomedzi milión možností trafil konkrétny, aktívny šestciferný kód. Avšak naša webstránka vytvára oneskorenie pred ďalším pokusom písania. Väčšie bezpečenstvo predstavuje botnet. Botnet je skupina počítačov, ktoré plnia príkazy ich „veliteľa“ (v našom prípade útočník). Uvažujme, že by mal botnet 1000 strojov. Každý z nich by nepretržite, automatizované zadával rôzne šestciferné kódy. Mal by otvorených 10 okien prehliadača, takže by paralelne skúšal 10 pokusov naraz. S našim oneskorením na webstránke za 2 minúty stíha jeden stroj v botnete 400 pokusov. Ak je v botnete 1000 takýchto strojov, za 2 minúty stihnuť neuveriteľných 400 000 pokusov. To je pokrytie takmer polovice priestoru, do ktorého sa útočník snaží trafiť.

4.1 Kopírovanie do schránky

Celkom závažný problém je moment, kedy sa používateľ na webstránke dostane k svojim údajom. Potrebuje si ich skopírovať, aby ich vedel použiť. Pointa je, že tento úkon je cielený na cudzie zariadenia. To znamená, že ak používateľ dokončí svoju prácu a odíde od cudzieho zariadenia, je viac než pravdepodobné, že jeden zo svojich citlivých údajov

nechá v schránke.

Potom stačí, aby útočník počkal, kým obet odíde. Po jej odchode príde k zariadeniu. Získa retazec uložený v schránke a ukradne ho. Na tento problém sme nenašli riešenie.

Záver

Počas celej práce sme sa snažili nájsť bezpečný, pohodlný a efektívny spôsob prístupu používateľa k heslám na cudzom zariadení. Teda na takom zariadení, kde password manager s citlivými údajmi používateľa nie je prítomný (mimo ekosystému).

Nami nájdený spôsob je pohodlný. Používateľ nemusí na cudzom zariadení nič inštalovať. Stránka nemá vysoké nároky na výkon. Má minimum elementov, ktoré sa rýchlo načítajú aj na zariadeniach so slabším výkonom/pomalým pripojením na internet.

Najpohodlnnejší spôsob je verifikácia QR kódom. Ak má používateľ funkčnú kameru na svojom smartfóne, celý proces verifikácie sa vykoná do sekundy.

V opačnom prípade je pre neho k dispozícii šestciferný kód. Pamäťanie čísel nie je náročné, ich zadávanie je rýchle. Po ich zadaní je verifikácia taká rýchla ako pri QR kóde.

Snažili sme sa o to, aby bola aplikácia Passer prehľadná a „user-friendly“. Pri jednotlivých položkách má používateľ k dispozícii tlačidlo *Outsider*, takže sa rýchlo dostane k prenosu dát. Z hľadiska zložitosti používania, či časovej zložitosti sme splnili požiadavky: riešenie je lepšie ako manuálne prepisovanie z password managera, či inštalácia podpornej aplikácie na cudzom zariadení, keď ide o niečo jednorázové.

Týmto sme zodpovedali aj otázku efektivity. Úkony (verifikácia) sú rýchle. Na serveri ukladáme dáta do cache, čo je najrýchlejší spôsob, ako k nim pristúpiť. Používame symetrickú kryptografiu. Tá je oveľa rýchlejšia ako asymetrická.

Ako sme písali v Hrozby, to, či je naše riešenie bezpečné, sa zistí časom. Ale z teoretického hľadiska sa nám podarilo ochrániť naše riešenie tak, aby uspokojilo dnešné bezpečnostné štandardy. AES, HTTPS, SHA-256, ECDH sú overené protokoly, algoritmy a štandardy, ktoré sú bezpečné a spoľahlivé. Preto sme ich implementovali do finálneho riešenia.

Zoznam použitej literatúry

1. *Heslo* [online]. Wikimedia Foundation, 2020 [cit. 2020-02-04]. Dostupné z: <https://cs.wikipedia.org/wiki/Heslo>.
2. *Bezpečnosť údajov* [online]. Wikimedia Foundation, 2019 [cit. 2020-02-04]. Dostupné z: https://sk.wikipedia.org/wiki/Bezpe%C4%8Dnos%C5%A5_%C3%BAdajov.
3. BAKA, Tomáš. *Kryptografická ochrana hesiel* [online]. Bratislava: Slovenská technická univerzita v Bratislave, 2017 [cit. 2020-02-04].
4. *Sila hesla* [online]. Wikimedia Foundation, 2016 [cit. 2020-02-04]. Dostupné z: https://sk.wikipedia.org/wiki/Sila_hesla.
5. BALLA, Martin. *Autentifikácia a bezpečnosť hesiel* [online]. Bratislava: Slovenská technická univerzita v Bratislave, 2015 [cit. 2020-02-04].
6. *Software pro správu hesel* [online]. Wikimedia Foundation, 2019 [cit. 2020-02-14]. Dostupné z: https://cs.wikipedia.org/wiki/Software_pro_spr%C3%A1vu_hesel.
7. *What is a Master Password?* [online] [cit. 2020-02-14]. Dostupné z: <https://simplicable.com/new/master-password-definition>.
8. GROŠEK, Otokar, VOJVODA, Milan a ZAJAC, Pavol. *Klasické šifry*. Bratislava: Vydavatelstvo STU, 2007.
9. *Create website and app passwords on iPhone* [online] [cit. 2020-02-14]. Dostupné z: <https://support.apple.com/en-gb/guide/iphone/iphf9219d8c9/ios>.
10. *Celebrating 10 Years of LastPass* [online]. 2018 [cit. 2020-02-14]. Dostupné z: <https://blog.lastpass.com/2018/07/celebrating-10-years-lastpass.html/>.
11. *The best way to manage passwords*. [online] [cit. 2020-02-14]. Dostupné z: <https://www.lastpass.com/how-lastpass-works>.
12. *LastPass* [online]. Wikimedia Foundation, 2020 [cit. 2020-02-14]. Dostupné z: <https://en.wikipedia.org/wiki/LastPass#Features>.
13. *Transport Layer Security* [online]. Wikimedia Foundation, 2017 [cit. 2020-02-14]. Dostupné z: https://sk.wikipedia.org/wiki/Transport_Layer_Security.
14. *10 Million People Globally Use Dashlane to Manage Passwords* [online]. 2020 [cit. 2020-02-14]. Dostupné z: <https://blog.dashlane.com/10-million-users/>.

15. *How to use Password Changer* [online] [cit. 2020-02-14]. Dostupné z: <https://support.dashlane.com/hc/en-us/articles/202699281-How-to-use-Password-Changer>.
16. ROUSE, Margaret. *What is Two-Factor Authentication (2FA) and How Does It Work?* [online]. TechTarget, 2020 [cit. 2020-02-14]. Dostupné z: <https://searchsecurity.techtarget.com/definition/two-factor-authentication>.
17. *Myki Password Manager & Authenticator Review* [online] [cit. 2020-02-14]. Dostupné z: <https://www.pcmag.com/reviews/myki-password-manager-authenticator>.
18. RUBENKING, Neil J. *Survey: Hardly Anybody Uses a Password Manager* [online]. PCMag, 2015 [cit. 2020-02-25]. Dostupné z: <https://www.pcmag.com/news/survey-hardly-anybody-uses-a-password-manager>.
19. *Co je malware a jak ho odstranit: Ochrana proti malware* [online] [cit. 2020-02-25]. Dostupné z: <https://www.avast.com/cs-cz/c-malware>.
20. BOTT, Ed. *Do you save passwords in Chrome? Maybe you should reconsider* [online]. ZDNet, 2013 [cit. 2020-02-25]. Dostupné z: <https://www.zdnet.com/article/do-you-save-passwords-in-chrome-maybe-you-should-reconsider/>.
21. MOSCARITOLO, Angela. *35 Percent of People Never Change Their Passwords* [online]. PCMag, 2018 [cit. 2020-02-25]. Dostupné z: <https://www.pcmag.com/news/35-percent-of-people-never-change-their-passwords>.
22. GROSSE, E. et al. *An investigation into users' considerations towards using password managers* [online]. SpringerOpen, 1970 [cit. 2020-02-25]. Dostupné z: <https://link.springer.com/article/10.1186/s13673-017-0093-6>.
23. *TextField* [online] [cit. 2020-05-21]. Dostupné z: <https://developer.apple.com/documentation/swiftui/textfield>.
24. *SecureField* [online] [cit. 2020-05-21]. Dostupné z: <https://developer.apple.com/documentation/swiftui/securefield>.
25. *identifierForVendor* [online] [cit. 2020-05-21]. Dostupné z: <https://developer.apple.com/documentation/documentation/uikit/uidevice/1620059-identifierforvendor>.
26. *Quickstart* [online] [cit. 2020-05-27]. Dostupné z: <https://flask.palletsprojects.com/en/1.1.x/quickstart/>.
27. *What is Memory Caching? How Memory Caching Works.* [online] [cit. 2020-05-27]. Dostupné z: <https://hazelcast.com/glossary/memory-caching/>.

28. *Cache* [online] [cit. 2020-05-27]. Dostupné z: <https://werkzeug.palletsprojects.com/en/0.16.x/contrib/cache/>.
29. *Status Codes* [online] [cit. 2020-05-27]. Dostupné z: <https://www.flaskapi.org/api-guide/status-codes/>.
30. *Window.localStorage* [online] [cit. 2020-05-27]. Dostupné z: <https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage>.
31. FERGUSON, Niels a SCHNEIER, Bruce. *Practical Cryptography* [online]. United States of America: Wiley Publishing, Inc., 2003 [cit. 2020-03-14].
32. *Encrypting for Apple's Secure Enclave* [online]. 2018 [cit. 2020-05-12]. Dostupné z: <https://darthnull.org/security/2018/05/31/secure-enclave-ecies/>.
33. *Storing Keys in the Secure Enclave* [online] [cit. 2020-05-12]. Dostupné z: https://developer.apple.com/documentation/security/certificate_key_and_trust_services/keys/storing_keys_in_the_secure_enclave.
34. *Integrated Encryption Scheme* [online]. Wikimedia Foundation, 2019 [cit. 2020-05-28]. Dostupné z: https://en.wikipedia.org/wiki/Integrated_Encryption_Scheme.
35. *Cold boot attack* [online]. Wikimedia Foundation, 2020 [cit. 2020-05-28]. Dostupné z: https://en.wikipedia.org/wiki/Cold_boot_attack.