

Aplikace vodoznaku a digitálního podpisu do QR kódu pro zajištění autenticity

Bc. Petr Nagy



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2024/2025

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	Bc. Petr Nagy
Osobní číslo:	A23323
Studijní program:	N0613A140022 Informační technologie
Specializace:	Softwarové inženýrství
Forma studia:	Prezenční
Téma práce:	Aplikace vodoznaku a digitálního podpisu do QR kódu pro zajištění autenticity
Téma práce anglicky:	Application of Watermarking and Digital Signature in QR Codes for Ensuring Authenticity

Zásady pro vypracování

- Nastudujte a popište metody pro zabezpečení QR kódů pomocí vodoznaku a digitálního podpisu.
- Zpracujte srovnání možností integrace těchto bezpečnostních prvků do QR kódů.
- Sepište výhody a nevýhody jednotlivých přístupů, s ohledem na čitelnost a spolehlivost QR kódů.
- Zvolte vhodnou metodu a implementujte ukázkový QR kód s vodoznakem a digitálním podpisem.
- Výsledky vhodně popište a prezentujte.

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. KATZENBEISSER, Stefan. *Information Hiding Techniques for Steganography and Digital Watermarking*. Boston: Artech House Publishers, 1999. ISBN 978-1580530354
2. KATZ, Jonathan a YEHUDA, Lindell. *INTRODUCTION TO MODERN CRYPTOGRAPHY*. Second edition. Boca Raton (Florida): CRC Press, 2015. ISBN 978-1-4665-7026-9
3. HASSABALLAH, Mahmoud. *Digital Media Steganography*. Qena (Egypt): Stacy Masucci, 2020. ISBN 978-0-12-819438-6
4. NEMATOLLAHI, Mohammad Ali; VORAKULPIPAT, Chalee a GAMBOA ROSALES, Hamurabi. *Digital watermarking: techniques and trends*. Springer topics in signal processing. Singapore: Springer, 2017. ISBN 978-981-10-2095-7
5. KATZ, Jonathan. *Digital Signatures*. London: Sprienger, 2010. ISBN 978-0-387-27712-7.

Vedoucí diplomové práce: **Ing. Petr Žáček, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **27. října 2024**
Termín odevzdání diplomové práce: **2. června 2025**



doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

L.S.

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 17. listopadu 2024

PROHLÁŠENÍ AUTORA DIPLOMOVÉ PRÁCE

Beru na vědomí, že

- odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb., v platném znění bez ohledu na výsledek obhajoby;
- diplomová práce bude uložena v elektronické podobě v univerzitním informačním systému a bude dostupná k nahlédnutí;
- jedno vyhotovení diplomové práce v listinné podobě bude ponecháno Univerzitě Tomáše Bati ve Zlíně k uložení;
- na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- podle § 60 odst. 1 autorského zákona má Univerzita Tomáše Bati ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- podle § 60 odst. 2 a 3 mohu užít své dílo – diplomovou práci – nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tj. k nekomerčnímu využití), nelze výsledky diplomové práce využít ke komerčním účelům;
- pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá; neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji, že

- jsem na diplomové práci pracovala samostatně a použitou literaturu jsem řádně citoval; v případě publikace výsledků budu uveden jako spoluautor;
- odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou obsahově totožné.
- při tvorbě této práce jsem použil nástroje generativních modelů AI ChatGPT; <https://chatgpt.com> a Grok; <https://grok.com> za účelem kontroly gramatiky, stylistiky a stavby vět a pro hledání zdrojů informací. Po použití tohoto nástroje jsem provedl/a kontrolu obsahu a přebírám za něj plnou zodpovědnost.

Ve Zlíně, dne

.....

podpis autora

ABSTRAKT

Diplomová práce se zaměřuje na aplikování vodoznaku a digitálního podpisu do QR kódů za účelem zajištění jejich autenticity. Zkoumá možnosti, jak přidat tyto bezpečnostní prvky, aniž by došlo k narušení jejich čitelnosti. Práce se zabývá teoretickými i praktickými přístupy k zabezpečení QR kódů, včetně stávajících řešení. Důraz je kladen na ochranu dat a ověřitelnost informací zakódovaných v QR kódech. Výstupem bude aplikace, která bude pro tyto účely sloužit.

Klíčová slova:

QR kód, Steganografie, Vodoznak, Digitální podpis, Python, PyScript, Autenticita, Zabezpečení dat, Kryptografie, GitHub Pages, Quishing, Integrita dat, Verifikace

ABSTRACT

This Master's thesis focuses on applying watermarking and digital signatures to QR codes to ensure their authenticity. It explores ways to integrate these security features without compromising readability. The work examines both theoretical and practical approaches to securing QR codes, including existing solutions. Emphasis is placed on data protection and the verifiability of information encoded in QR codes. The outcome will be an application designed for this purpose.

Keywords:

QR code, Steganography, Watermark, Digital signature, Python, PyScript, Authenticity, Data security, Cryptography, GitHub Pages, Quishing, Data integrity, Verification

Tímto bych chtěl velmi poděkovat vedoucímu práce Ing. Petru Žáčkovi, Ph.D. za cenné rady, konzultace a pevné nervy ohledně vypracovávání diplomové práce.

OBSAH

ÚVOD.....	13
I TEORETICKÁ ČÁST	14
1 QR KÓDY	15
1.1 HISTORIE A VÝVOJ QR KÓDŮ	15
1.2 STRUKTURA A PRINCIP FUNGOVÁNÍ QR KÓDŮ	17
1.2.1 Kódování dat v QR kódech	19
1.3 TYPY QR KÓDŮ A ÚROVNĚ KOREKCE CHYB ECL	19
1.4 MOŽNOSTI ČTENÍ A GENEROVÁNÍ QR KÓDŮ	20
1.5 VÝHODY A NEVÝHODY QR KÓDŮ	21
1.6 PRAKTICKÉ VYUŽITÍ QR KÓDŮ	22
2 BEZPEČNOSTNÍ RIZIKA QR KÓDŮ	23
2.1 QUISHING A PODVODNÉ QR KÓDY	23
2.1.1 Příklady reálného zneužití	24
2.2 TYPY ÚTOKŮ NA QR KÓDY	24
2.2.1 Přesměrování na škodlivý web	25
2.2.2 Instalace malwaru nebo spywaru	25
2.2.3 Automatizované akce (volání, SMS, Wi-Fi připojení)	25
2.2.4 Zneužití platebních údajů	25
2.2.5 Sociální inženýrství pomocí QR kódů	26
2.3 METODY OCHRANY PROTI ZNEUŽITÍ	26
2.3.1 Softwarové ochrany	26
2.3.2 Edukace uživatelů	27
2.3.3 Technická opatření na straně tvůrce	27
2.4 DŮSLEDKY KOMPROMITACE QR KÓDU	27
2.4.1 Dopady na uživatele	28
2.4.2 Dopady na organizace	28
3 KRYPTOGRAFICKÉ ALGORITMY PRO ZABEZPEČENÍ QR KÓDŮ	29
3.1 HASH FUNKCE	29
3.1.1 SHA-256	30
3.1.2 Další využívané hash funkce	30
3.2 DIGITÁLNÍ PODPIS	31
3.2.1 RSA	32
3.2.2 EdDSA	32
3.2.3 Další algoritmy digitálních podpisů	32
4 METODY OCHRANY OBSAHU V QR KÓDECH	33
4.1 VYUŽITÍ DIGITÁLNÍHO PODPISU	33
4.2 TYPY VODOZNAKŮ	33

4.3	STEGANOGRAFIE A JEJÍ VYUŽITÍ	34
4.3.1	Metoda LSB (Least Significant Bit).....	34
4.4	SROVNÁNÍ METOD OCHRANY	35
5	EXISTUJÍCÍ ŘEŠENÍ A PŘÍSTUPY.....	36
5.1	APLIKACE VODOZNAKŮ V QR KÓDECH	36
5.2	DIGITÁLNÍ PODPISY A OVĚŘOVÁNÍ AUTENTICITY QR KÓDŮ	37
5.3	STEGANOGRAFIE V QR KÓDECH	37
5.4	SHRNUTÍ KOMERČNÍCH A BEZPLATNÝCH NÁSTROJŮ	37
II	PRAKTICKÁ ČÁST	39
6	POUŽITÉ TECHNOLOGIE.....	40
6.1	PROGRAMOVACÍ JAZYK PYTHON	40
6.2	PYSCRIPT A WEBOVÉ TECHNOLOGIE	41
6.3	GITHUB PAGES PRO NASAZENÍ APLIKACE.....	42
7	SROVNÁNÍ METOD ZABEZPEČENÍ QR KÓDŮ	43
7.1	METODIKA SROVNÁNÍ	43
7.2	PŘÍPADY UŽITÍ.....	43
7.3	VÝSLEDKY TESTOVÁNÍ.....	44
7.3.1	Digitální podpis přímo v QR kódu	44
7.3.2	Digitální podpis uložen pomocí LSB steganografie.....	45
7.3.3	Digitální podpis jako samostatný soubor	46
7.3.4	Aplikace vodoznaku	46
7.4	ANALÝZA VÝHOD A NEVÝHOD	48
7.4.1	Digitální podpis přímo v QR kódu	48
7.4.2	Digitální podpis uložen pomocí LSB steganografie.....	49
7.4.3	Digitální podpis jako samostatný soubor	49
7.4.4	Aplikace vodoznaku	49
7.4.5	Přínos vůči hrozbě Quishingu	49
8	NÁVRHY APLIKACÍ.....	50
8.1	NÁVRH WEBOVÉ APLIKACE	50
8.1.1	Požadavky na aplikaci	51
8.1.2	Cíle webové aplikace.....	52
8.1.3	Prostředí GitHub Pages	53
8.1.4	Rozdělení aplikace do funkčních sekcí	53
8.2	NÁVRH DESKTOPOVÉ APLIKACE QREBUILD	54
9	IMPLEMENTACE WEBOVÉ APLIKACE.....	56
9.1	UŽIVATELSKÉ ROZHRAŇÍ.....	56
9.1.1	Webové rozhraní	57
9.1.2	Způsob ovládání	61

9.2	ARCHITEKTURA APLIKACE	62
9.2.1	Modulární struktura.....	62
9.2.2	Integrace s PyScript.....	64
9.3	IMPLEMENTACE DIGITÁLNÍHO PODPISU	65
9.3.1	Integrace digitálního podpisu pomocí LSB steganografie	66
9.3.2	Distribuce digitálního podpisu pomocí textového souboru	67
9.3.3	Algoritmus digitálního podpisu.....	67
9.4	IMPLEMENTACE VODOZNAKU A ODZNAKU DO QR KÓDU.....	70
9.4.1	Integrace vodoznaku do QR kódu.....	70
9.4.2	Integrace odznaku do QR kódu.....	71
9.4.3	Algoritmus aplikace vodoznaku a odznaku	71
9.5	FUNKČNÍ TOK APLIKACE.....	74
9.5.1	Generování QR kódu.....	74
9.5.2	Dekódování QR kódu.....	78
9.5.3	Ověření QR kódu	79
9.5.4	Detekování QR kódu.....	81
9.6	AUTOMATICKÉ NASAZENÍ WEBOVÉ APLIKACE.....	81
10	DESKTOPOVÁ APLIKACE QREBUILD.....	82
10.1	DŮVOD VZNIKU A ÚČEL APLIKACE	82
10.1.1	Uživatelské rozhraní.....	83
10.2	ARCHITEKTURA A IMPLEMENTACE	84
10.2.1	Modulární struktura.....	84
10.3	FUNKCIONALITA APLIKACE	87
10.3.1	Nahrání a detekce QR kódu z fotografie.....	87
10.3.2	Rekonstrukce QR kódu do digitální podoby	87
10.3.3	Zobrazení výsledků a export	88
10.4	AUTOMATICKÉ SESTAVENÍ A DISTRIBUCE.....	90
10.4.1	Sestavení aplikace pomocí GitHub Actions.....	90
10.4.2	Distribuce prostřednictvím GitHub Releases.....	92
	ZÁVĚR	94
	SEZNAM POUŽITÉ LITERATURY.....	95
	SEZNAM OBRÁZKŮ	101
	SEZNAM TABULEK.....	102
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....	103
	SEZNAM PŘÍLOH.....	105

ÚVOD

QR kódy se staly běžnou součástí každodenního života díky své schopnosti rychle a efektivně přenášet informace mezi fyzickým a digitálním světem. V současnosti se využívají v celé řadě oblastí, například při platbách, v marketingových kampaních nebo ve zdravotnictví. S rostoucí popularitou však přicházejí i bezpečnostní rizika, jako je Quishing, manipulace s daty nebo zneužití citlivých informací. Tato diplomová práce se zaměřuje na analýzu a řešení těchto rizik prostřednictvím návrhu a implementace bezpečnostních prvků, konkrétně digitálního podpisu a vodoznaku, které zvyšují důvěryhodnost a autenticitu QR kódů.

Technologie neustále pokračují ve svém vývoji a přinášejí nové možnosti, ale zároveň i výzvy. Bezpečnost QR kódů je jednou z oblastí, která si žádá pozornost, protože jejich otevřená povaha je činí zranitelnými vůči zneužití. Ačkoliv některé aspekty zabezpečení QR kódů, jako kryptografické algoritmy a steganografie, jsou již známy, jejich praktická implementace a optimalizace pro reálné použití zůstávají složitým úkolem.

Cílem diplomové práce je analyzovat stávající metody ochrany QR kódů, navrhnout a implementovat řešení pro generování a ověřování QR kódů s bezpečnostními prvky a otestovat různé přístupy k jejich integraci. Teoretická část zkoumá základy technologie QR kódů a jejich bezpečnostní aspekty. Kapitola 1 popisuje strukturu, typy a principy fungování QR kódů. Kapitola 2 se věnuje bezpečnostním rizikům, jako je Quishing nebo manipulace s daty. Kapitola 3 analyzuje kryptografické algoritmy RSA, ECDSA, EdDSA, SHA-256, a jejich využití pro zabezpečení QR kódů. Kapitola 4 rozebírá metody ochrany, včetně vodoznaků a steganografie. Kapitola 5 hodnotí existující nástroje a přístupy k zabezpečení QR kódů. Praktická část se zaměřuje na návrh a vývoj dvou aplikací. Kapitola 6 popisuje použité technologie a nástroje při vývoji aplikací. Kapitola 7 detailně popisuje srovnání metod použitých pro zabezpečení QR kódů, výsledky testování a analýzu jejich výhod a nevýhod. Kapitola 8 detailně rozebírá vývoj webové aplikace postavené na technologiích HTML, CSS, JavaScript, PyScript a GitHub Pages, která slouží k generování, podepisování a ověřování QR kódů. Kapitola 9 se věnuje desktopové aplikaci QRebuild pro rekonstrukci QR kódů z fotografií na systémech Windows a Linux, nahrazující nedostatek webové aplikace.

I TEORETICKÁ ČÁST

1 QR KÓDY

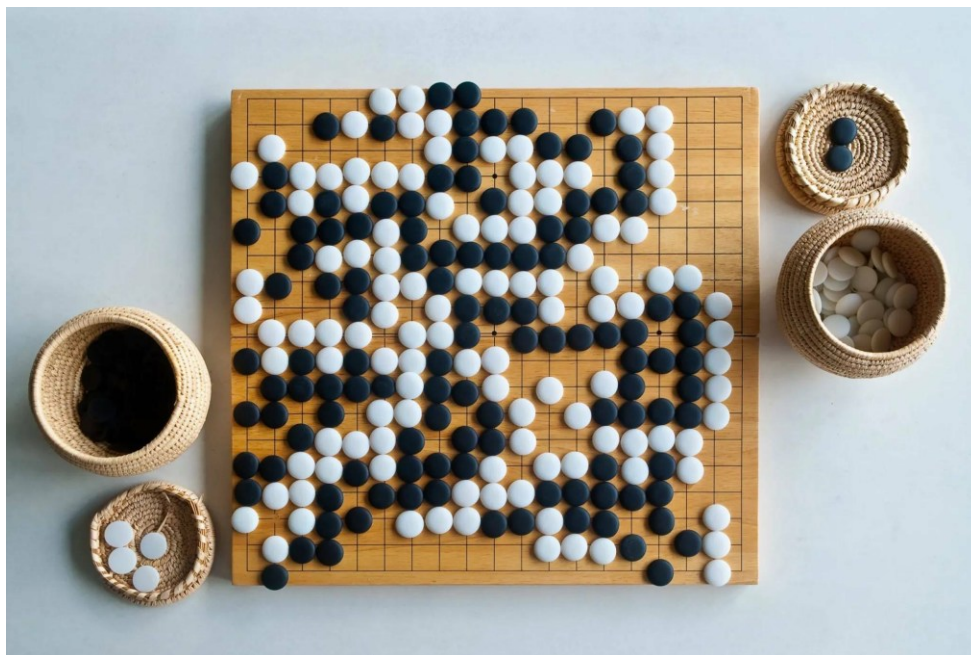
QR (Quick Response) kódy představují významný prvek moderní digitální komunikace. Jedná se o dvourozměrné maticové kódy, které umožňují rychlé a spolehlivé zakódování a dekodování různých typů dat, od jednoduchého textu až po komplexní strukturované informace. Díky své flexibilitě, snadné čitelnosti a vysoké odolnosti proti poškození se staly nepostradatelnou součástí mnoha technologických a komerčních oblastí. [1] [2]

Od svého vzniku v 90. letech 20. století prošly QR kódy výrazným vývojem. I přestože byly původně navrženy pro průmyslové využití, dnes nacházejí uplatnění v širokém spektru aplikací, od marketingu a elektronických plateb, přes veřejnou dopravu až po zdravotnictví. Jejich význam vzrostl zejména s nástupem chytrých telefonů, které umožňují pohodlné skenování kódů bez potřeby speciálního vybavení, neboť téměř každý telefon je vybaven aplikací pro čtení QR kódů, případně si ji uživatel může snadno nainstalovat z obchodu s aplikacemi. [2] [3] [5]

Velkým milníkem v rozšíření QR kódů bylo jejich začlenění do mobilních zařízení, obzvláště chytrých telefonů, které umožnily jejich skenování bez specializovaného vybavení. Dlouhodobě se QR kódy uplatňují také v bankovníctví a online platbách, kde usnadňují rychlé a bezpečné transakce. Další významný impuls k masovému využití QR kódů přinesla pandemie COVID-19, která umožnila bezkontaktní přístup k informacím například prostřednictvím očkovacích certifikátů, jídelních lístků či rezervačních systémů. [1] [4] [5]

1.1 Historie a vývoj QR kódů

QR kódy byly vyvinuty v roce 1994 pod vedením inženýra Masahira Hary japonskou společností Denso Wave, dceřinou firmou Toyoty. Hlavním cílem bylo vytvořit efektivnější alternativu k tradičním jednorozměrným čárovým kódům, jako jsou EAN nebo UPC, které měly kapacitu pouhých 20 alfanumerických znaků. Zároveň disponovali omezením v podobě nutnosti přesného zarovnání jednorozměrných kódů při skenování, které razantně zpomalovaly procesy ve výrobě. Samotný vývoj QR kódů započal již v roce 1992 hledáním řešení, které by umožnilo rychlé čtení dat z jakéhokoli směru a možnosti uložení více dat do jednoho kódu. Inspirací se stala japonská hra Go, jejíž dvourozměrná mřížka sloužila jako základ pro strukturu QR kódu. K vyřešení druhého klíčového prvku, skenování kódu z libovolného směru a úhlu, posloužily čtvercové značky v rozích kódu. [3] [5]



Obrázek 1 – Japonská stolní hra Go

(zdroj: https://usgo.org/content.aspx?page_id=5&club_id=454497&item_id=111091)

Po dokončení vývoje v roce 1994 se společnost Denso Wave rozhodla specifikaci QR kódů uvolnit jako veřejného doménu. Tento krok byl zásadní pro popularizaci technologie nejen v Japonsku, ale i globálně. Standardizace technologie probíhala postupně. Nejprve v roce 1997 byly QR kódy přijaty do standardu AIM, následně v roce 1999 do japonského JIS a na závěr v roce 2000 do mezinárodního standardu ISO/IEC 18004. Tyto kroky zajistily jejich konzistentní použití a kompatibilitu napříč různými platformami. [3] [5]

První vlna popularity QR kódů přišla v Japonsku kolem roku 2002, kdy se mobilní telefony s kamerami a čtecím softwarem staly běžnou součástí v každodenním životě. QR kódy začaly sloužit k propojení fyzického a digitálního světa, kde jejich využití sahalo od odkazů na webové stránky přes slevové kupóny až po elektronické vstupenky. S příchodem chytrých telefonů se využití QR kódů rozšířilo i mimo Japonsko, přičemž zlomový okamžik nastal v roce 2017, kdy operační systémy iOS a Android integrovaly nativní skenery QR kódů, čímž odpadla potřeba stahovat samostatné aplikace. Další výrazný posun přinesla pandemie COVID-19 v roce 2020, kdy QR kódy našly uplatnění v bezkontaktních řešeních. Restaurace je využívaly pro digitální menu, dopravní společnosti pro elektronické jízdenky a státní instituce pro očkovací certifikáty či sledování kontaktů. [3] [5] [6]

Tabulka 1 – Klíčové události ve vývoji QR kódů

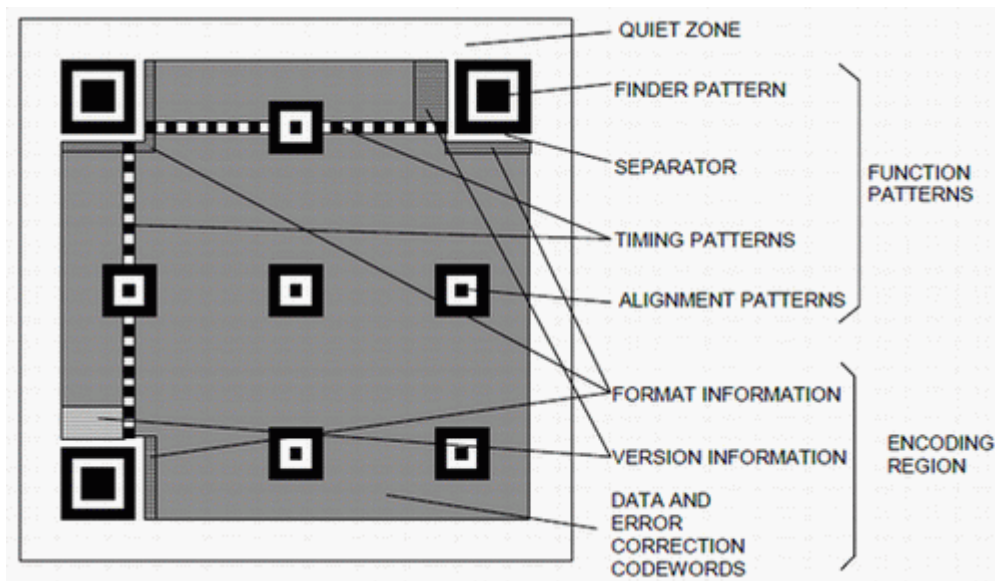
Rok	Událost
1992	Začátek vývoje QR kódů Denso Wave
1994	Vynález QR kódů, primární použití v automobilovém průmyslu.
1994	Specifikace uvolněna jako veřejná doména
1997	Standard AIM
1999	Standard JIS
2000	Standard ISO/IEC
2002	První vlna popularity kvůli mobilním telefonům v Japonsku
2004	Představení Micro QR Code
2008	Představení iQR Code
2014	Představení FrameQR
2017	Druhá vlna popularity kvůli vestavěným skenerům QR kódu v iOS a Android
2020	Třetí vlna popularity kvůli pandemii COVID-19

(zdroj: [3] [5] [6])

V průběhu několika let, vlivem různých potřeb využití QR kódů, vznikly různé varianty QR kódů. Mezi tyto varianty patří například Micro QR Code z roku 2004, určený pro menší plochy, nebo iQR Code z roku 2008 s větší kapacitou a obdélníkovým tvarem. V roce 2014 přibyl FrameQR, který umožňuje začlenit grafické prvky přímo do QR kódu. [3]

1.2 Struktura a princip fungování QR kódů

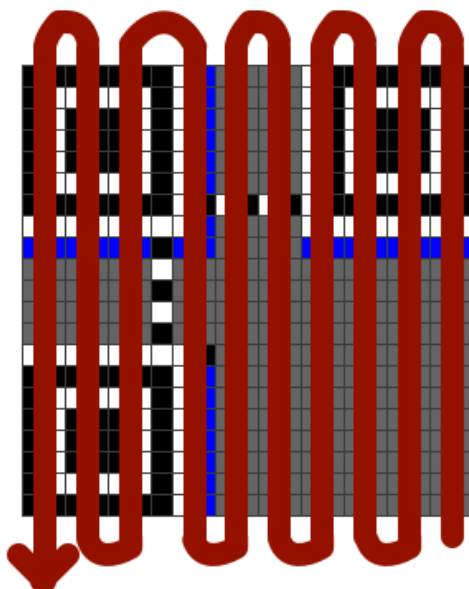
QR kódy představují dvourozměrné maticové kódy, které umožňují rychlé ukládání a čtení dat, jako jsou webové odkazy, text nebo kontaktní údaje. Oproti jednorozměrným čárovým kódům, které pojmulý pouze 20 alfanumerických znaků, QR kódy nabízejí kapacitu až 7 089 číslic nebo 4 296 alfanumerických znaků, díky čemuž jsou ideální pro širší spektrum využití. Struktura QR kódu zahrnuje několik klíčových prvků. Prvním jsou čtvercové značky v rozích kódu, s poměrem 1:1:3:1:1, usnadňují rozpoznání orientace a skenování z libovolného úhlu. Dalšími klíčovými prvky jsou časovací vzory, umístěné v šesté řadě a sloupci, které slouží k synchronizaci čtení, a datová oblast obsahující zakódované informace a kódy pro korekci chyb založené na algoritmu Reed-Solomon, který umožňuje obnovit až 30 % poškozených dat podle zvolené úrovně korekce. [7] [8] [9]



Obrázek 2 – Struktura QR kódu

(zdroj: [7])

Princip fungování QR kódu spočívá v dekódování dat skenerem, který nejprve rozpozná čtvercové značky v rozích kódu, pote se synchronizuje pomocí časovacích vzorů a následně přečte data z datové oblasti od dolního pravého rohu směrem vzhůru. Korekce chyb zajišťuje spolehlivost i při částečném poškození kódu. Velikost QR kódu se liší podle verze, od 21x21 modulů ve verzi 1, po 177x177 modulů ve verzi 40, přičemž kapacita dat, které mohou být uloženy, klesá s vyšší úrovní korekce chyb. [7] [8] [9]



Obrázek 3 – Směr čtení dat z QR kódu

(zdroj: [7])

1.2.1 Kódování dat v QR kódech

Kódování dat v QR kódech je proces převodu informací, například textu, URL adresy nebo kontaktních údajů, do dvourozměrného maticového kódu, který lze rychle dekódovat skenerem. QR kódy podporují čtyři hlavní režimy kódování. Numerický, pro čísla 0–9, alfanumerický, pro velká písmena, číslice a vybrané symboly, binární, pro obecná data včetně textu v UTF-8 a kanji, pro japonské znaky. Každý režim je optimalizován pro konkrétní typ dat, aby se snížil počet potřebných bitů. Proces kódování zahrnuje výběr režimu, který se přidá pomocí 4bitového indikátoru, například 0010 je indikátor pro alfanumerický režim, určením počtu znaků a převod dat na bitový řetězec. [7] [8]

Kódování je ovlivněno verzí QR kódu a úrovní korekce chyb, která určuje, kolik prostoru zůstane pro data. Například Verze 40 s úrovní korekce chyb L pojme maximální kapacitu, zatímco s úrovní H značně méně. Po zakódování dat se přidávají Reed-Solomon kódy pro korekci chyb a maska pro optimalizaci čitelnosti. Tento proces zajišťuje, že QR kód je kompaktní, odolný vůči poškození a snadno čitelný. [7] [8]

1.3 Typy QR kódů a úrovně korekce chyb ECL

QR kódy existují v několika typech, které se liší velikostí, kapacitou a zaměřením na konkrétní aplikace. Standardní QR kódy mají 40 verzí, od 21x21 modulů ve verzi 1 až po 177x177 modulů ve verzi 40, a slouží širokému spektru účelů, od logistiky po marketing. Specializované varianty zahrnují Micro QR, iQR, FrameQR, rMQR a SQRC kódy. Micro QR kódy jsou ve verzích M1–M4, určené pro malé plochy s kapacitou maximálně 35 numerických znaků. iQR kódy nabízejí větší kapacitu a mají obdélníkový tvar. FrameQR kódy se využívají především v reklamě díky možnosti začlenit grafické prvky přímo do QR kódu. rMQR kódy jsou navrženy pro úzké prostory, kde standardní QR kódy nelze použít, a kapacita Micro QR kódů není dostačující. SQRC kódy obsahují funkci omezení čtení pro ochranu soukromých dat, přičemž vypadají jako běžné QR kódy. [7] [10]



Obrázek 4 – Typy QR kódů

(zdroj: [10])

Klíčovou vlastností QR kódů je korekce chyb, založená na algoritmu Reed-Solomon, který umožňuje obnovit poškozená data. Existují čtyři úrovně korekce: L, M, Q a H, kde každá úroveň označuje procento dat, která lze rekonstruovat. Nižší úroveň L je vhodná pro čistá prostředí, jako jsou například digitální displeje, zatímco vyšší úroveň H se používá v náročných podmínkách, například v továrnách nebo místech, kde je vyšší náchylnost na poškození. Tato flexibilita zajišťuje spolehlivost kódu i při částečném zakrytí, zašpinění nebo poškození. [11] [12] [13]

Úroveň korekce chyb ovlivňuje kapacitu dat výsledného QR kódu. Například Verze 1 s úrovní korekce chyb L pojme až 152 bajtů, ale s úrovní H pouze 72 bajtů, protože vyšší korekce vyžaduje více prostoru pro redundantní data. Designérské QR kódy s logy nebo barevnými prvky často využívají úroveň H, aby zůstaly čitelné i při grafických úpravách. Volba vhodného typu a úrovně korekce je tak zásadní pro vyvážení kapacity, odolnosti a estetiky v závislosti na konkrétním použití. [11] [12] [13]

Tabulka 2 – Kapacita QR kódu ve verzi 1 s pouze numerickými daty

Úroveň	Procento rekonstrukce	Kapacita (bajty)
L	7 %	152
M	15 %	128
Q	25 %	104
H	30 %	72

(zdroj: [13])

1.4 Možnosti čtení a generování QR kódů

Čtení QR kódů je díky moderním technologiím jednoduché a široce dostupné. Většina chytrých telefonů s operačními systémy iOS a Android má od roku 2017 vestavěné čtečky, které umožňují skenování QR kódů přímo přes kameru bez nutnosti další aplikace. Pro zařízení bez této funkce existují dedikované aplikace, které nabízejí pokročilé funkce, jako například ukládání historie skenování nebo podporu různých formátů kódů navíc oproti pouhým QR kódům. [5] [14]

Generování QR kódů je přístupné díky online webovým aplikacím, které nevyžadují programátorské znalosti. Nejprve uživatel zadá data pro vybraný typ QR kódu, například webovou adresu, text nebo vCard. Pokud to aplikace umožňuje, může uživatel upravit vzhled generovaného kódu přidáním loga, změněním barvy a úpravou tvaru. Výsledky QR kód si

poté můžou stáhnout nejčastěji ve formátu PNG nebo SVG. Některé aplikace umožňují vytváření i dynamických QR kódů, jejichž obsah lze později měnit, což je ideální pro marketingové kampaně nebo aktualizace informací, například digitálních menu v restauracích. [14] [15]

Standard ISO/IEC 18004:2024 zajišťuje kompatibilitu a kvalitu QR kódů tím, že definuje jejich kódování, strukturu a korekci chyb. Tento standard umožňuje, aby všechny QR kódy vygenerované různými nástroji byly čitelné jakoukoli standardní čtečkou, což je klíčové pro jejich univerzální využití v logistice, marketingu nebo zdravotnictví. [16]

1.5 Výhody a nevýhody QR kódů

QR kódy nabízejí rychlé skenování, vysokou kapacitu dat, nízké náklady na generování a odolnost vůči poškození díky korekci chyb, což je činí univerzálním nástrojem v použití v oblastech pro marketing, logistiku či sdílení informací. Dynamické QR kódy umožňují sledování skenování, což je cenné pro analýzu kampaní. Naopak pro skenování je vyžadovaný chytrý telefon a často internetové připojení, což omezuje přístupnost a zvyšuje bezpečnostní rizika, jako je například snazší šíření malware. [17] [18] [19]

Mezi výhody QR kódů řadíme [17] [18] [19]:

- Rychlé skenování chytrými telefony
- Vysoká kapacita dat až 7 089 znaků
- Nízké náklady na generování
- Odolnost vůči poškození kvůli korekci chyb
- Univerzálnost použití
- Sledování skenování u dynamických QR kódů

Mezi nevýhody QR kódů řadíme [17] [18] [19]:

- Závislost na smartphonu a internetu
- Bezpečnostní rizika, například šíření malware
- Neznalost skenování u některých uživatelů
- Jednosměrná komunikace
- Neznalost obsahu před naskenováním

1.6 Praktické využití QR kódů

QR kódy se díky rychlému a snadnému vytvoření, univerzálnosti a nízkým nákladům uplatňují v mnoha odvětvích. V marketingu propojují fyzické a digitální prostředí, umožňují přístup k webovým stránkám, sociálním sítím nebo formulářům pro sběr kontaktů. V logistice usnadňují sledování produktů a správu dodavatelských řetězců, zatímco ve zdravotnictví poskytují rychlý přístup k informacím, například zdravotním údajům nebo preventivním pokynům. Jejich bezkontaktní povaha získala na významu během pandemie, kdy se využívaly pro platby nebo přístup k digitálním materiálům. [2] [6] [20] [21]

Ve finančním sektoru se QR kódy využívají pro bezkontaktní platby, rychlé transakce v obchodech nebo ve veřejné dopravě. V turismu zjednodušují přístup k informacím a službám, například využitím při platbách nebo získáním navigačním pokynům či informacích ohledně atrakce. Dále se uplatňují při sdílení Wi-Fi přihlašovacích údajů, poskytování detailů o produktech na obalech nebo podpoře zákaznického zapojení prostřednictvím zpětné vazby a propagačních akcí. V bezpečnosti QR kódy pomáhají ověřovat pravost produktů a bojovat proti padělkům, zejména ve farmaceutickém průmyslu. Ve vzdělávání a kultuře obohacují zážitky, například poskytováním dodatečného obsahu v muzejních expozicích. [20] [21]

2 BEZPEČNOSTNÍ RIZIKA QR KÓDŮ

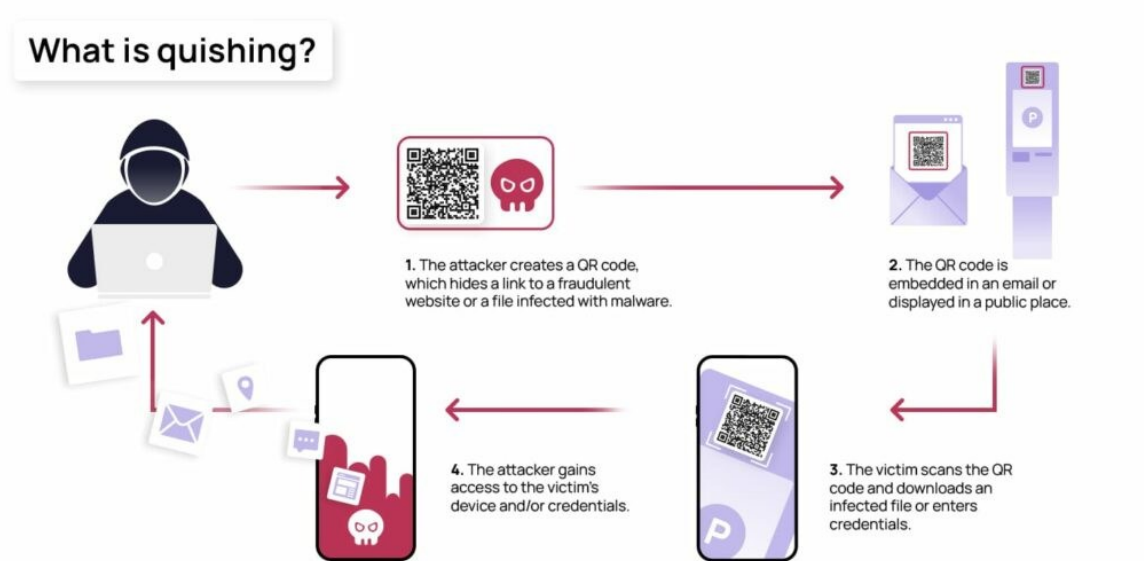
QR kódy představují efektivní nástroj pro rychlý přenos informací, avšak jejich široké využití v marketingu, elektronických platbách či logistice přináší bezpečnostní rizika, která mohou ohrozit nejen uživatele, ale i samotné organizace. Mezi hlavní hrozby patří zneužití kódů k phishingovým útokům, šíření škodlivého softwaru nebo manipulace s daty, často kvůli nedostatečné obezřetnosti uživatelů či nedostatkům v zabezpečení generování a skenování kódů. [22] [23]

2.1 Quishing a podvodné QR kódy

Quishing, neboli QR phishing, je forma kybernetického útoku, při kterém útočníci využívají QR kódy k oklamání uživatelů a přesměrování na škodlivé webové stránky nebo k instalaci malware. Na rozdíl od tradičního phishingu, který využívá nejčastěji e-maily nebo SMS, Quishing láká oběti ke skenování QR kódů umístěných na veřejných místech, jako jsou plakáty, parkoviště nebo restaurace, případně těch poslaných digitálně. Tyto kódy často vedou na falešné přihlašovací stránky, které kradou osobní údaje, nejčastěji přihlašovací jména, hesla nebo bankovní informace, nebo spouštějí automatické stažení škodlivého softwaru. Quishing je obzvláště nebezpečný díky obtížnosti rozpoznání škodlivého kódu pouhým okem a rostoucí důvěře uživatelů v QR kódy, čemuž obzvláště prospěla pandemie COVID-19, kdy se využití QR kódů pro bezkontaktní služby rozšířilo. [23] [24]

Statické QR kódy obsahují pevně zakódovaná data, která nelze po vytvoření měnit, což z nich činí méně rizikové z hlediska manipulace. Naopak dynamické QR kódy umožňují aktualizaci cílového obsahu i po jejich vygenerování, což sice přináší větší flexibilitu, ale také otevírá prostor pro zneužití, kdy útočníci mohou směřovat uživatele na škodlivé stránky bez viditelných změn v samotném kódu. [23]

Úspěch Quishingu spočívá v jeho nenápadnosti a schopnosti obejít běžné bezpečnostní filtry, jako jsou e-mailové detektory spamu. Útočníci často napodobují důvěryhodné QR kódy, například banky nebo doručovací služby. Dalším faktorem je absence vizuálních indikátorů škodlivosti. Na rozdíl od podezřelých URL v e-mailech obsah QR kódu nelze dekódovat před skenováním. Tento nedostatek je jedním z hlavních bezpečnostních rizik QR kódů. [23] [24]



Obrázek 5 – Proces získání dat využitím Quishingu

(zdroj: [23])

2.1.1 Příklady reálného zneužití

Quishingové útoky se v praxi projevují různými způsoby, často s vážnými důsledky. Jedním z běžných scénářů je přelepení legitimních QR kódů na parkovištích, kde škodlivý kód přesměruje uživatele na falešnou platební stránku, která krade bankovní údaje. Dalším příkladem je zneužití v restauracích, kde falešné QR kódy na jídelních lístcích odkazují na škodlivé weby požadující ověření identity, což umožní útočníkům získat citlivé údaje. Dalším příkladem jsou podvodné QR kódy zaslané v e-mailech tak, aby napodobovaly doručovací služby a lákaly uživatele na stažení malwaru pod záminkou sledování zásilky nebo doplnění informací o zásilce pro její úspěšné doručení. Další formou zneužití je distribuce QR kódů prostřednictvím sociálních sítí nebo plakátů, které slibují slevy nebo soutěže. [23] [25]

2.2 Typy útoků na QR kódy

QR kódy představují efektivní nástroj pro rychlý přenos informací, avšak jejich široké rozšíření a jednoduchost použití je činí atraktivním cílem pro kybernetické útočníky. Quishing je jedním z hlavních rizik, kde útočníci oklamají uživatele skenováním falešných QR kódů, což často vede k přesměrování na škodlivé weby. Každý typ útoku využívá specifické techniky, od manipulace s obsahem QR kódu po psychickou manipulaci uživatelů, a může vést k závažným důsledkům, jako jsou finanční ztráty, únik dat nebo

narušení bezpečnosti zařízení. Klíčovým faktorem úspěchu těchto útoků je jejich nenápadnost, protože QR kódy neprozrazují svůj obsah před naskenováním, což zvyšuje riziko pro neopatrné uživatele. [23] [24] [25]

2.2.1 Přesměrování na škodlivý web

Přesměrování na škodlivý web je jedním z nejčastějších útoků na QR kódy, úzce spojený s Quishingem. Útočníci vytvoří QR kód, který po naskenování přesměruje uživatele na falešnou webovou stránku, které napodobují důvěryhodné služby, jako jsou například bankovní portály, e-shopy nebo doručovací služby, s cílem ukrást přihlašovací údaje nebo osobní informace. Tyto stránky často využívají techniky, které mají za účel vyvolat pocit nutné a bezodkladné akce. [23] [25]

2.2.2 Instalace malwaru nebo spywaru

Dalším typem útoku je instalace malwaru nebo spywaru, kdy QR kód po naskenování spustí stažení a instalaci škodlivého softwaru bez vědomí uživatele. Tento útok je obzvláště nebezpečný, protože uživatelé často nekontrolují obsah QR kódu před naskenováním. Útočníci často distribuují tyto kódy prostřednictvím e-mailů, sociálních sítí nebo plakátů na veřejně dostupných místech, což umožňuje šíření malwaru na širokou škálu zařízení. [23] [24] [25]

2.2.3 Automatizované akce (volání, SMS, Wi-Fi připojení)

Automatizované akce zahrnují útoky, při kterých QR kód po naskenování spustí nechtěné akce, jako například automatické volání na nežádoucí telefonní čísla, odesílání SMS zpráv nebo připojení k nechráněným Wi-Fi sítím. Tyto akce mohou vést k finančním ztrátám nebo k narušení bezpečnosti zařízení, pokud se připojí ke škodlivé a špatně zabezpečené Wi-Fi, která umožňuje útočníkům odposlouchávat komunikaci. Tento typ útoku je zvláště efektivní v prostředích, kde uživatelé automaticky skenují kódy pro rychlý přístup ke službám, například pro připojení k Wi-Fi v kavárnách nebo obchodních centrech. [26]

2.2.4 Zneužití platebních údajů

Zneužití platebních údajů je dalším vážným typem útoku, zejména v kontextu rostoucí popularity bezkontaktních plateb prostřednictvím QR kódů. Útočníci mohou přelepit originální QR kódy na platebních terminálech nebo vytvořit falešné kódy, které po

naskenování přesměrují platbu na účet útočníka nebo přímo ukradnou platební údaje, jako je číslo platební karty nebo přihlašovací údaje k bankovnímu účtu. Tento typ útoku, známý jako QRLjacking, je obzvláště nebezpečný v prostředích, kde uživatelé automaticky skenují kódy pro platby. [23] [25]

2.2.5 Sociální inženýrství pomocí QR kódů

Sociální inženýrství je klíčovou součástí mnoha útoků pomocí QR kódů, včetně Quishingu. Útočníci využívají důvěryhodný vzhled QR kódů a jejich široké rozšíření v používání mezi uživateli, aby je přiměli skenovat škodlivé QR kódy. Příklady využití sociálního inženýrství jsou například rozesílání e-mailů nebo zpráv na sociálních sítích s QR kódy slibujícími slevy, soutěžemi nebo urgentními aktualizacemi, což vede uživatele k naskenování kódu. Tento typ útoku je zvláště efektivní, protože využívá psychologických faktorů, jako je zvědavost nebo naléhavost. [23] [27]

2.3 Metody ochrany proti zneužití

QR kódy představují významná bezpečnostní rizika, včetně Quishingu, instalace malwaru, automatizovaných akcí, zneužití platebních údajů a sociálního inženýrství. Tyto hrozby lze však minimalizovat kombinací softwarových ochran, edukace uživatelů a technických opatření na straně tvůrců QR kódů. Efektivní prevence vyžaduje zvýšení povědomí o nebezpečích a implementaci robustních bezpečnostních mechanismů, aby se minimalizovala pravděpodobnost úspěšného útoku. Úspěch ochrany závisí na proaktivním přístupu, který zahrnuje jak technické řešení, tak vzdělávání. Například softwarové nástroje pro čtení QR kódů mohou detekovat škodlivé kódy ještě před skenováním. [23] [24]

2.3.1 Softwarové ochrany

Softwarové ochrany hrají klíčovou roli v prevenci útoků pomocí QR kódů, jako je Quishing nebo automatická instalace malwaru. Moderní antivirové programy a bezpečnostní aplikace nabízejí funkce pro ověření QR kódů před jejich naskenováním, čímž detekují škodlivé URL nebo podezřelé akce. Některé prohlížeče, například Google Chrome, integrují varování před phishingovými weby, které se uživatel může snažit navštívit. Další softwarová řešení zahrnují aplikace pro bezpečné skenování QR kódů, které ověřují legitimnost odkazů a blokují přístup k nebezpečným stránkám. Softwarové ochrany jsou obzvláště efektivní proti útokům typu přesměrování na škodlivé weby nebo instalace malwaru, protože mohou analyzovat obsah QR kódu v reálném čase a reagovat na něj. [23] [24]

2.3.2 Edukace uživatelů

Edukace uživatelů je hlavním prvkem v oblasti prevence proti útokům založených na sociálním inženýrství, jako je Quishing nebo falešných QR kódů. Před skenováním je potřeba ověřit zdroj QR kódu, například zkontrolováním, zda kód nebyl přelepen jiným kódem nebo zda pochází od důvěryhodné organizace. Dalším klíčovým prvkem je správné rozpoznání podezřelé situace, jako jsou například naléhavé výzvy k provedení nějaké akce nebo QR kódy obsažené v neočekávaných e-mailech či ve zprávách na sociálních sítích. Nedostatečná obezřetnost uživatelů je totiž jedním z hlavních faktorů úspěchu Quishingových útoků. Edukace by měla zahrnovat praktické tipy, jako je používání bezpečných skenerů QR kódů, kontrola webového odkazu po naskenování QR kódu a vyhýbání se skenování kódů na veřejných místech bez předchozího ověření obsahu. [23] [25] [28]

2.3.3 Technická opatření na straně tvůrce

Technická opatření na straně tvůrců QR kódů jsou nezbytná pro zajištění jejich bezpečnosti a prevenci útoků, jako je například QRLjacking nebo přelepení kódů. Mezi klíčová opatření patří použití bezpečných generátorů QR kódů, které umožňují šifrování obsahu nebo vkládání digitálních podpisů pro ověření autenticity. Například dynamické QR kódy umožňují aktualizaci obsahu a sledování skenování, což ztěžuje manipulaci útočníky. Dále, v případě QR kódu s webovým odkazem, je doporučeno používat HTTPS odkazy v QR kódech, aby se zajistilo šifrované připojení a snížilo riziko odposlechu. Dále je potřeba pravidelně kontrolovat integritu QR kódu dostupných na veřejných místech, například na platebních terminálech nebo plakátech. [23] [28] [29]

2.4 Důsledky kompromitace QR kódu

Kompromitace QR kódů je vážným bezpečnostním rizikem, které může mít významné důsledky jak pro jednotlivé uživatele, tak pro organizace. Následky naskenování komprimovaného QR kódu mohou být od reputační škody přes finanční škody až po právní následky. [23] [24]

2.4.1 Dopady na uživatele

Uživatelé, kteří naskenují kompromitovaný QR kód, mohou čelit řadě negativních důsledků, které ovlivňují jejich finanční bezpečnost, soukromí a bezpečnost zařízení. Mezi hlavní dopady patří: [23] [24] [25]

- Finanční ztráty
 - Platby pomocí QR kódu, zadání přihlašovacích údajů od bankovníctví
- Krádež a ztráta dat
 - Přihlašovací údaje, kontaktní údaje, fotografie a videa, dokumenty
- Krádež identity
 - Citlivé osobní údaje, bankovní identita
- Infekce zařízení malwarem
 - Automatické stažení a spuštění škodlivého softwaru

2.4.2 Dopady na organizace

Organizace, které využívají QR kódy ve svých procesech, mohou čelit významným důsledkům, pokud jejich kódy budou kompromitovány. Mezi hlavní dopady patří: [23] [24] [25]

- Reputační škody
 - Využití QR kódů organizace pro distribuci malware
- Finanční ztráty
 - Převedení plateb za služby na jiný účet než organizace
- Operační narušení
 - Ztráta přístupu ke kritickým systémům, downtime
- Právní důsledky
 - GDPR, získání citlivých údajů zaměstnanců nebo organizace

3 KRYPTOGRAFICKÉ ALGORITMY PRO ZABEZPEČENÍ QR KÓDŮ

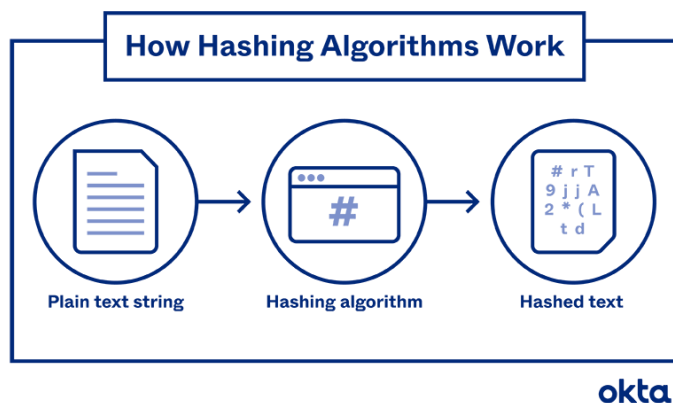
S rostoucím počtem využívání QR kódů v citlivých oblastech, jako je online bankovníctví, dvoufázové ověření či elektronické vstupenky, nabývá jejich zabezpečení zásadního významu. Kryptografie hraje klíčovou roli při ochraně integrity a autenticity dat zakódovaných v QR kódech, přičemž nejčastěji využívanými nástroji jsou hashovací funkce a digitální podpisy. [30]

3.1 Hash funkce

Hash funkce jsou matematické algoritmy, které převádějí vstupní data libovolné délky na výstup pevné délky, který se nazývá hash. Jedná se o jednosměrné funkce, což znamená, že z hash hodnoty nelze zpětně odvodit původní data a slouží primárně k ověření integrity informací. V QR kódech mohou hash funkce vytvořit digitální otisk dat, který při dekodování potvrzuje, že obsah nebyl pozměněn, protože i malá změna ve vstupních datech vede k výrazné změně hash hodnoty na výstupu. [31] [32]

Moderní kryptografické hash funkce, jako je například SHA-256, mají následující klíčové vlastnosti: [31] [32]

- Jednosměrnost – Nelze získat vstupní data z hash hodnoty
- Odolnost vůči kolizím – Neexistují dva stejné vstupy, které mají stejnou hash hodnotu na výstupu
- Determinističnost – Stejný vstup vždy vygeneruje stejnou hash hodnotu na výstupu



Obrázek 6 – Znázornění hashovací funkce

(zdroj: [31])

3.1.1 SHA-256

SHA-256 (Secure Hash Algorithm) je široce používaná kryptografická hash funkce z rodiny SHA-2. Vytváří 256bitový, neboli 32bajtový, hash, obvykle vyjádřený 64 šestnáctkovými číslicemi. Díky své odolnosti vůči kolizím a útokům nalezením vzorů, tedy nalezení vstupu pro daný výstupní hash, je považována za bezpečnou. V QR kódech se SHA-256 využívá k vytvoření hash hodnoty obsahu, která může být uložena spolu s daty nebo na jiném bezpečném místě. Při skenování se hash znovu vypočítá a porovná s uloženou hodnotou, čímž se ověří neporušenost dat. [31] [32] [33]

3.1.2 Další využívané hash funkce

Kromě algoritmu SHA-256 existuje řada dalších hashovacích funkcí, které se liší svými vlastnostmi, bezpečností a vhodností pro různé aplikace. Prvním algoritmem je MD5, který byl vyvinut v roce 1991 a produkuje 128bitový hash. Ačkoli byl dlouho standardem pro kontrolu integrity dat, dnes je považován za kryptograficky nezabezpečený kvůli známým kolizím, které umožňují dvěma různým vstupům mít stejný hash. Proto se jeho použití v bezpečnostních aplikacích nedoporučuje. Dalším algoritmem je SHA-1, který produkuje 160bitový hash a byl široce používán v různých bezpečnostních protokolech. Nicméně, podobně jako MD5, byl SHA-1 prolomen a již není považován za bezpečný pro kryptografické účely. Algoritmus SHA-3 je nejnovější člen rodiny SHA algoritmů, vyvinutý jako alternativa k SHA-2 s odlišnou vnitřní strukturou založenou na algoritmu Keccak a nabízí vysokou úroveň bezpečnosti. Na závěr moderní hashovací funkce BLAKE3 je navržena pro vysokou rychlost a bezpečnost. Je rychlejší než mnoho starších algoritmů, včetně MD5 a SHA-1, a zároveň nabízí silnou kryptografickou odolnost. Díky své efektivitě a bezpečnosti je vhodná pro široké spektrum aplikací, včetně těch, které vyžadují rychlé zpracování dat. [32] [34]

Tabulka 3 – Srovnání hashovacích algoritmů

Algoritmus	Výstupní délka (bitů)	Bezpečnostní status
MD5	128	Nebezpečný
SHA-1	160	Nebezpečný
SHA-256	256	Bezpečný
SHA-3-256	256	Bezpečný
BLAKE3	Variabilní	Bezpečný

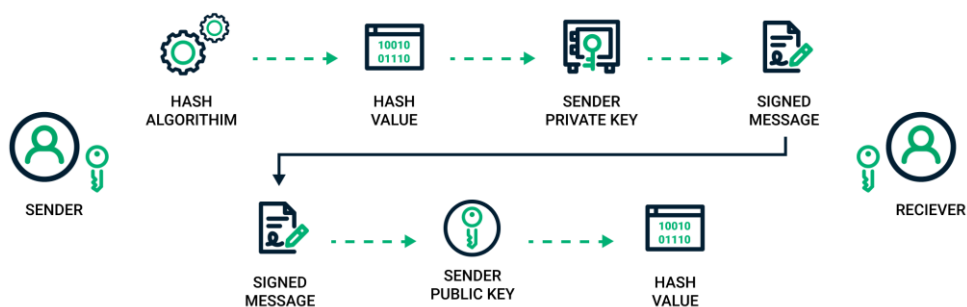
(zdroj: [32] [34])

3.2 Digitální podpis

Digitální podpisy jsou kryptografické mechanismy, které ověřují autenticitu a integritu dat, zpráv nebo softwaru. Jsou to digitální ekvivalenty ručních podpisů nebo razítek, které nabízí vyšší úroveň bezpečnosti. Digitální podpis řeší problém manipulace a falšování v digitální komunikaci a zvyšují transparentnost online interakcí a budují důvěru mezi uživateli, partnery a dodavateli. V kontextu QR kódů jsou digitální podpisy klíčové pro zajištění, že data obsažená v QR kódu pocházejí z důvěryhodného zdroje a nebyla pozměněna. Účelem digitálního podpisu je zajistit autenticitu, že data pochází od odesílatele, integritu, že data nebyla změněna a zabránění popírání autorství. Tyto vlastnosti jsou dosaženy pomocí veřejného klíčového kryptografického systému, kde odesílatel použije svůj soukromý klíč k podepsání dat, zatímco příjemce použije veřejný klíč k ověření podpisu. [35] [36]

Digitální podpisy se dělí podle použitých kryptografických algoritmů, které určují jejich bezpečnost, efektivitu a kompatibilitu. Mezi hlavní algoritmy patří RSA (Rivest-Shamir-Adleman), založený na problému faktorizace velkých čísel, EdDSA (Edwards-curve Digital Signature Algorithm), moderní algoritmus založený na Edwardsových křivkách, a další, jako DSA (Digital Signature Algorithm) nebo ECDSA (Elliptic Curve Digital Signature Algorithm), které nabízejí alternativní přístupy k zabezpečení. Výběr algoritmu závisí na specifických potřebách aplikace, včetně bezpečnostních úrovní, výpočetních požadavků a kompatibility s existujícími systémy. [36] [37] [38]

How Does a Digital Signature Work?



Obrázek 7 – Princip digitálního podpisu a jeho ověření

(zdroj: [36])

3.2.1 RSA

RSA je jeden z nejznámějších algoritmů digitálních podpisů, založený na problému faktorizace velkých čísel. Data jsou nejprve zahashována a výsledný hash je zašifrován soukromým klíčem, čímž vznikne podpis. Příjemce dešifruje podpis veřejným klíčem a porovná ho s hashem originálních dat. RSA je vhodný pro zabezpečení citlivých informací v QR kódech, například bankovních údajů nebo oficiálních dokumentů. RSA je tradiční algoritmus, který je široce přijatý, ale může být náročný na výpočetní zdroje, zejména pro velké klíče. [39]

3.2.2 EdDSA

EdDSA je moderní algoritmus digitálních podpisů založený na Edwardsových křivkách. Nabízí vysokou bezpečnost a jednoduchost implementace, což ho činí ideálním pro moderní aplikace, včetně digitálních měn a zabezpečení QR kódů. EdDSA je odolný vůči útokům a efektivní v pohledu výpočetních nároků, což ho činí vhodným pro mobilní zařízení, kde je důležitá nízká spotřeba zdrojů. Proces zahrnuje hashování dat a generování podpisu pomocí soukromého klíče, přičemž příjemce ověří podpis veřejným klíčem. [39] [40]

3.2.3 Další algoritmy digitálních podpisů

Kromě RSA a EdDSA existují i jiné algoritmy digitálních podpisů, které mohou být použitelné v závislosti na specifických požadavcích. Mezi tyto algoritmy patří DSA standardizovaný podle FIPS 186-4, který je založený na diskretním logaritmickém problému a je široce používán v bezpečnostních systémech. Zároveň je ale méně flexibilní než RSA nebo EdDSA. Dalším algoritmem je ECDSA, který je založený na eliptických křivkách. ECDSA nabízí stejnou bezpečnost jako RSA, ale s menšími klíči, což ho činí efektivnějším pro mobilní zařízení. Tyto algoritmy poskytují alternativní přístupy k zabezpečení dat v QR kódech, přičemž výběr závisí na potřebách aplikace. [39]

4 METODY OCHRANY OBSAHU V QR KÓDECH

Vzhledem k rostoucímu využití QR kódů v různých oblastech je zajištění integrity a autenticity jejich obsahu stále důležitější. QR kódy totiž mohou obsahovat citlivá nebo důvěrná data, která je třeba chránit proti neoprávněným zásahům, falšování či manipulaci. Existuje několik přístupů, jak tento obsah účinně zabezpečit. Mezi nejčastěji využívané metody patří digitální podpisy, které umožňují ověřit původ dat a jejich nezměnitelnost. Dále pak aplikace viditelných i neviditelných vodoznaků, které slouží jako dodatečný identifikátor či ochranný vizuální prvek. Při práci s digitálními obrázky QR kódů najde také hojně využití i steganografie, která umožňuje ukrytí informací přímo do obrázku, například pomocí techniky LSB (Least Significant Bit). [30]

4.1 Využití digitálního podpisu

Digitální podpisy zajišťují autenticitu a integritu dat v QR kódech. Podepsaná data lze ověřit, že pocházejí z legitimního zdroje a nebyla změněna, což je klíčové pro citlivé informace, například finanční dokumenty nebo akademické záznamy. Ve veřejném prostoru můžou podepsané QR kódy pomoci k nižšímu procentu napadených uživatelů prostřednictvím skenování QR kódů. V kontextu QR kódů se digitální podpisy používají například v digitálních COVID certifikátech, kde ověřují pravost a nezměněnost informací. Hlavní výhodou je možnost ověření původu dat a detekce jakýchkoli změn. Mezi omezení můžeme zařadit potřebu specializovaných aplikací pro ověření podpisu a složitost implementace. [30] [41] [42]

4.2 Typy vodoznaků

Vodoznaky představují důležitý nástroj pro ochranu digitálního obsahu, včetně QR kódů, před neoprávněným použitím, paděláním a porušováním autorských práv. Obecně lze vodoznaky rozdělit na viditelné a neviditelné. Viditelné vodoznaky jsou grafické prvky, jako jsou loga, texty nebo QR kódy, které jsou záměrně umístěny do obsahu a slouží k identifikaci vlastníka nebo zdroje. Neviditelné vodoznaky jsou skryté informace vložené do obsahu tak, aby nebyly běžným způsobem detekovatelné, ale mohly být později extrahovány pomocí speciálních nástrojů. Tyto vodoznaky jsou často využívány pro sledování šíření obsahu a prokazování vlastnictví, aniž by narušily vizuální integritu původního materiálu. [14] [15] [43] [44]

Integrace vodoznaků do QR kódů může být realizována různými technikami. Nejznámějším přístupem je vkládání loga přímo do QR kódu, obvykle do jeho středu nebo rohu. Tato technika využívá schopnosti QR kódů tolerovat určitou míru poškození kvůli implementaci ECL, což umožňuje zachovat čitelnost kódu i při překrytí části jeho plochy logem. Vkládání loga do QR kódu nejen zvyšuje estetickou hodnotu a vizuální rozpoznatelnost značky, ale také může posílit důvěru uživatelů. Při implementaci této techniky je však nezbytné dbát na správné umístění a velikost loga, aby nedošlo k narušení klíčových prvků QR kódu, jako jsou detekční vzory v rozích. Dalším přístupem je vložení QR kódu jako neviditelného vodoznaku do digitálního obrazu pomocí transformací v prostorové nebo frekvenční doméně, například pomocí diskrétní vlnkové transformace (DWT) nebo diskrétní kosinové transformace (DCT). Tato technika umožňuje zachovat vizuální kvalitu obrazu a zároveň poskytuje robustnost vůči běžným útokům, jako je komprese nebo ořezávání. [14] [15] [43] [45] [46]

4.3 Steganografie a její využití

Steganografie je technika skrývání informací tak, aby nebyly viditelné pouhým okem uživatele. Na rozdíl od kryptografie, která chrání obsah zprávy, steganografie zajišťuje, že přítomnost zprávy zůstane skrytá v médiu. V digitálním prostředí se steganografie často využívá k ukrytí dat v multimediálních souborech, jako jsou obrázky, zvukové nahrávky nebo videa. V kontextu QR kódů může být steganografie použita k zakódování citlivých informací přímo do vizuální struktury kódu nebo k vložení QR kódu do jiného média, například do obrázku, čímž se zvyšuje úroveň zabezpečení a diskrétnosti. [47] [48]

4.3.1 Metoda LSB (Least Significant Bit)

Jednou z nejběžnějších metod digitální steganografie je technika LSB, která spočívá ve změně nejméně významných bitů v digitálním médiu. Tato metoda umožňuje vložení skryté zprávy do média, aniž by došlo k viditelným změnám pro lidské oko. Při aplikaci LSB na vygenerované obrázky QR kódů, lze například vložit šifrovanou zprávu do obrázku tak, že orázek zůstane pro lidské oko nezměněný, ale zpráva bude stále dekodovatelná pomocí specializovaného softwaru. Využitím kombinace steganografie a kryptografie zvyšuje bezpečnost přenášených informací. [47] [49]

4.4 Srovnání metod ochrany

Pro výběr vhodné metody ochrany obsahu v QR kódech je důležité zvážit několik faktorů: [42] [44] [45] [47]

- Digitální podpisy – Poskytují vysokou úroveň důvěryhodnosti a integrity dat, ale mohou být náročné na implementaci a vyžadují správu klíčů.
- Vodoznaky – Viditelné vodoznaky, jako je vložení loga do QR kódu, zvyšují rozpoznatelnost kódu, zatímco neviditelné vodoznaky poskytují diskrétní ochranu proti neoprávněnému kopírování.
- Steganografie (LSB) – Umožňuje skryté přenášení informací bez zjevné přítomnosti dat, což je výhodné pro citlivé informace, ale může být náchylná k detekci při použití pokročilých analytických nástrojů.

Každá z těchto metod má své specifické výhody a omezení. Kombinace těchto metod může poskytnout vyšší úroveň zabezpečení v závislosti na konkrétních požadavcích a kontextu použití.

5 EXISTUJÍCÍ ŘEŠENÍ A PŘÍSTUPY

V posledních letech se QR kódy staly běžnou součástí digitální komunikace a marketingu. S jejich rostoucím využitím se zvyšují i požadavky na zajištění autenticity, integrity a bezpečnosti těchto kódů. V reakci na to vznikla celá řada nástrojů a metod, které do QR kódů integrují prvky jako vodoznaky, digitální podpisy nebo steganografii. Tyto přístupy se liší podle účelu použití, dostupnosti funkcí, technické náročnosti a finančních nákladů.

5.1 Aplikace vodoznaků v QR kódech

Vodoznaky slouží k ověření autenticity a integrity QR kódů a mohou být viditelné nebo neviditelné. Existující nástroje nabízejí různé způsoby jejich aplikace: [14] [15] [49] [50]

- Viditelné vodoznaky – Tyto vodoznaky, například loga nebo text, jsou umístěny přímo do QR kódu, nejčastěji do středu nebo tichých zón na prázdných okrajích kódu, aby nebránily jeho čitelnosti. Nástroje jako QRCode Monkey a QR.io umožňují snadné přidání loga s možností nastavení průhlednosti a barev. Tyto nástroje jsou zdarma a oblíbené pro jednoduché na použití. Umístění vodoznaku do středu vygenerovaného QR kódu je preferovaný, protože většina čteček QR kódů dokáže data dekodovat i přes takovou úpravu.
- Neviditelné vodoznaky – Tyto vodoznaky nejsou okem viditelné a vyžadují speciální software pro jejich detekci. Například společnost Digimarc používá pokročilé techniky, jako DWT-DCT, k integraci autentizačních dat do QR kódu bez změny jeho vzhledu. Tento nástroj je placený a zaměřený na profesionální použití, například v průmyslových aplikacích.

Aplikace vodoznaků do QR kódů je využita v širokém spektru aplikací, od jednoduchých řešení zdarma, například QRCode Monkey, až po složitější placené nástroje, jako například od společnosti Digimarc. Bezplatné nástroje jsou široce dostupné a snadno použitelné, ale neviditelné vodoznaky zůstávají doménou placených řešení nebo vlastních implementací. [14] [15] [49] [50]

5.2 Digitální podpisy a ověřování autenticity QR kódů

Digitální podpisy jsou důležité pro zajištění toho, že data v QR kódu pocházejí z důvěryhodného zdroje a nebyla změněna. Běžné bezplatné nástroje, jako QRCode Monkey nebo QR.io, digitální podpisy nepodporují, což omezuje jejich využití pro bezpečnostní účely. Jedním z možných řešení je vlastní rozšíření open-source projektu, například nayuki/QR-Code-generator, o digitální podpisy. Použití digitálních podpisů není příliš časté u běžných uživatelů, protože je dostupné hlavně v placených nástrojích, jako je například Qrptal. Tyto nástroje cílí na specifické profesionální aplikace, jako je ověřování dokumentů. [14] [30] [49] [51]

5.3 Steganografie v QR kódech

Steganografie umožňuje skrýt dodatečná data v QR kódu, což může sloužit k přenosu citlivých informací nebo autentizaci. Žádný z běžných nástrojů, mezi které řadíme například QRCode Monkey nebo QR.io, steganografii přímo nepodporuje. Pro její implementaci je nutné použít vlastní implementaci nebo rozšíření nějakého open-source řešení. V reálných aplikacích pro generování QR kódů je využití steganografie velmi vzácné, protože chybí přímá podpora v komerčních i bezplatných nástrojích a její využití je omezeno primárně na specializované projekty. [14] [49]

5.4 Shrnutí komerčních a bezplatných nástrojů

Komerční a bezplatné nástroje se liší v přístupu, ceně a dostupných funkcích: [14] [30] [49] [50] [51]

- Komerční nástroje – Jsou placené nástroje s pokročilými funkcemi, jako jsou digitální podpisy nebo neviditelné vodoznaky. Příklady těchto řešení zahrnují Qrptal a Digimarc.
- Bezplatné nástroje – Jsou dostupné zdarma a nabízejí většinou základní funkce pro úpravu vygenerovaných QR kódů. Mezi bezplatné nástroje se řadí i open-source řešení, která navíc umožňují i možnost vlastní úpravy kódu přidáním vlastních funkcí. Příklady těchto řešení jsou nayuki/QR-Code-generator, QRCode Monkey a QR.io.

Tabulka 4 – Shrnutí dostupných nástrojů pro generování QR kódů

Nástroj	Typ	Vodoznaky	Digitální podpisy	Steganografie
Qryptal	Komerční	Ne	Ano	Ne
Digimarc	Komerční	Ano	Ne	Ne
QRCode Monkey	Zdarma	Ano	Ne	Ne
QR.io	Zdarma	Ano	Ne	Ne
nayuki/QR-Code-generator	Open-Source	Ne	Ne	Ne
Zint	Open-Source	Ne	Ne	Ne

(zdroj: [14] [30] [49] [50] [51] [52])

II PRAKTICKÁ ČÁST

6 POUŽITÉ TECHNOLOGIE

Pro vypracování praktické části byla zvolena kombinace programovacího jazyka Python, webových technologií HTML+CSS+JS a PyScript a platformy GitHub Pages. Webové technologie byly vybrány kvůli možnosti spuštění aplikace na většině zařízení s přístupem k internetu a možností procházet, zobrazovat a interagovat s webovými aplikacemi. Použití těchto technologií umožnilo vytvořit webovou aplikaci pro generování a ověřování QR kódů. Kvůli omezení platformy GitHub Pages byla vytvořena i multiplatformní desktopová aplikace QRebuild, která slouží pro detekování QR kódů z fotek a vygenerování jejich přesné kopie do digitální podoby.

6.1 Programovací jazyk Python

Programovací jazyk Python byl zvolen díky své jednoduchosti a všestrannosti, které umožňují rychlý a pohodlný vývoj. Vybrán byl také primárně pro svou širokou podporu knihoven pro zpracování obrázků, kryptografii a generování a čtení QR kódů. Využití Pythonu usnadňuje také jeho multiplatformnost a velmi aktivní komunita, která zajišťuje jednodušší řešení problémů a již zmíněné velké množství knihoven. Python byl použit v případě webové aplikace i desktopové aplikace QRebuild.

Ve webové aplikaci byl Python nasazen prostřednictvím open source platformy PyScript, která umožňuje spouštět Python kód přímo v prohlížeči. Pro generování QR kódů byla využita knihovna qrcode a pro aplikaci vodoznaku do vygenerovaného QR kódu se používá knihovna Pillow. Pro zpracování digitálního podpisu slouží knihovna cryptography.

Desktopová aplikace je rozdělena na dva základní typy, CLI a GUI. Jak již název napovídá, CLI aplikace je vytvořena za účelem spouštění pomocí příkazové řádky nebo terminálu a GUI aplikace disponuje klasickým uživatelským rozhraním. Oba typy používají sdílenou knihovnu, protože jejich účel je totožný. Využité jsou zde taktéž knihovny qrcode a Pillow. Kromě těchto knihoven jsou zde použité i knihovny pro detekci QR kódu z fotografie a vytvoření uživatelského rozhraní. Pro detekci QR kódu z fotografie je využita knihovna pyzbar, která je pouze Python nástavbou pro C knihovnu zbar a umožňuje využití této knihovny ve verzi Python 3. Pro vytvoření uživatelského rozhraní byla využita knihovna PyQt6.

6.2 PyScript a webové technologie

PyScript je technologie, která umožňuje spouštět Python kód přímo v prohlížeči pomocí WebAssembly. V kombinaci s ostatními webovými technologiemi (HTML, CSS a JavaScript) se jedná o výkonný nástroj, který umožňuje tvořit interaktivní a webové aplikace pomocí efektivního řešení i bez nutnosti použití serverové logiky. PyScript byl zvolen hlavně díky kompatibilitě s platformou GitHub Pages, která je využívána pro nasazení webové aplikace a byla veřejně dostupná. [53] [54]



Obrázek 8 – Infrastruktura technologie PyScript

(zdroj: [54])

Jelikož PyScript instaluje potřebné Python knihovny při každém načtení stránky, byla zvolena cesta jednoho HTML souboru, který obsahuje více záložek pro jednotlivé sekce (generování, dekodování, ověřování a detekování QR kódu). Záložky jsou napsané pomocí HTML kontejnerových elementů div, kde každý obsahuje unikátní ID. V jeden moment je zobrazená pouze jedna záložka, kterou uživatel vybere z navigační lišty. JavaScript slouží primárně pro zajištění interaktivity s webem samotným.

Komunikace mezi PyScriptem, Pythonem, HTML a JavaScriptem je zajištěna pomocí modulu js, který je potřeba importovat v Python skriptu, kde bude prováděná modifikace HTML souboru za běhu aplikace. Tento modul zajišťuje přístup k webovým rozhraním a umožňuje manipulovat s DOM, například upravovat nebo číst elementy v HTML. Typickým příkladem je vygenerování QR kódu pomocí qrcode knihovny v Python skriptu a následně jej zobrazit na webové stránce v předpřipraveném elementu.

6.3 GitHub Pages pro nasazení aplikace

GitHub Pages je platforma od GitHubu, která umožňuje bezplatné hostování webových stránek přímo z daného repozitáře. Přestože GitHub Pages umožňuje hostování pouze statických webových stránek, dokážeme web obohatit o dynamické funkce právě použitím nástrojů, které se spouští kód přímo ve webovém prohlížeči. [55]

Proces nasazení webové aplikace je plně zautomatizován využitím CI/CD nástroje GitHub Actions. Publikování webové aplikace probíhá při každém pushi do hlavní větve main v repozitáři, čímž se spustí vytvořený YAML workflow. Proces nasazení prvně stáhne kód repozitáře a následně připraví prostředí pro nasazení. Následně se vytvoří artefakt, který obsahuje všechny nezbytné soubory pro správnou funkčnost webové aplikace a ta se posléze nasadí. Výhodou zde je to, že není potřeba spravovat žádný server.

Hlavním omezením GitHub Pages je jednoznačně možnost hostování pouze statických webů bez podpory serverové logiky. Tyto nedostatky znemožňují detekovat QR kódy z fotografií přímo ve webové aplikaci. Kvůli těmto omezením jsem vytvořil desktopovou aplikaci QRebuild, která nahrazuje nemožnost detekování QR kódů z fotografií přímo ve webové aplikaci.

7 SROVNÁNÍ METOD ZABEZPEČENÍ QR KÓDŮ

V rámci vývoje webové aplikace bylo potřeba ověřit, jaké možnosti existují pro zajištění autenticity a integrity vygenerovaných QR kódů a aby možné tyto možnosti aplikovat a použít v již omezené platformě GitHub Pages. Z tohoto důvodu byla provedena analýza a praktické testy několika přístupů, jak zabezpečit QR kód pomocí digitálního podpisu a vodoznaku. Jednotlivé testy byly porovnány z hlediska bezpečnosti, kompatibility s platformou GitHub Pages a čitelnosti pomocí čtečky QR kódů.

Testované metody zahrnují vložení digitálního podpisu přímo do dat QR kódu ať do viditelné části či mimo viditelnou část, uložení podpisu pomocí LSB steganografie, vygenerování digitálního podpisu jako separátní dokument a poskytnutý ke stažení a možné kombinace. Výsledky těchto testů sloužily jako podklad pro implementaci finálního řešení.

Aplikace vodoznaku do QR kódu byla primárně testována tak, aby bylo možné vygenerovaný QR kód přečíst i s aplikovaným vodoznakem. Testovalo se uspořádání vrstev, nastavení průhlednosti, barev a umístění vodoznaku ve vygenerovaném obrázku QR kódu.

7.1 Metodika srovnání

Pro srovnání jednotlivých metod zabezpečení QR kódů byly zvoleny hodnotící kritéria, kterými jsou bezpečnost, možnost přečtení QR kódu pomocí běžných čteček i po aplikaci vodoznaku, použitelnost na platformě GitHub Pages. Každá metoda byla otestována a zhodnocena na základě těchto kritérií.

7.2 Případy užití

Testování jednotlivých metod a jejich kombinací zabezpečení QR kódu a aplikace vodoznaku probíhalo primárně na reálných případech, které odpovídají nejčastějším typům využívaných u QR kódů. Testované typy zahrnovali webový odkaz, klasický text, e-mail, Wi-Fi připojení, zeměpisné souřadnice, platební údaje pro Bitcoin, textová zpráva SMS, telefonní číslo, telefonní kontakt (elektronická vizitka vCard) a data ve formátu JSON.

Všechny metody zabezpečení, digitální podpis, steganografie a vodoznak, byly testovány na všech výše uvedených typech QR kódů. Cílem testování bylo zjistit odpovědi na následující otázky:

- Jak se mění čitelnost QR kódu běžnými čtečkami
- Dochází ke změně významu nebo typu QR kódu
- Změní se velikost QR kódu
- Je metoda kompatibilní s omezeními platformy GitHub Pages
- Je možné dekódovat QR kód přímo v prohlížeči

7.3 Výsledky testování

Výsledky testování byly rozděleny do čtyř částí, z nichž každá se zaměřuje na jednu konkrétní metodu zabezpečení QR kódu. První část se věnuje vložení digitálního podpisu přímo do dat QR kódu. Druhá část zkoumá možnost uložení podpisu pomocí LSB steganografie do obrázku samotného. Třetí část se zabývá vystavením digitálního podpisu jako samostatného souboru, který je poskytován ke stažení společně s QR kódem. Čtvrtá část se zaměřuje na aplikaci vodoznaku a její vliv na čitelnost a funkčnost QR kódů.

7.3.1 Digitální podpis přímo v QR kódu

Vložení digitálního podpisu přímo do dat QR kódu přineslo smíšené výsledky. V některých případech zůstaly vygenerované QR kódy čitelné a šli úspěšně ověřit, v jiných případech, zvláště u typů webového odkazu docházelo k nesprávné funkci QR kódu. Přidáním podpisu se změnilo chování QR kódu tak, že při použití čtečky se zobrazil pouze textový obsah QR kódu namísto otevření webové stránky. Podobná chyba se vyskytovala i v případě typů klasického textu a JSON formátu, kde QR kód byl správně přečten a rozpoznán, ale data podpisu výrazně zhoršovala čitelnost, protože byla čitelná společně s ostatními daty.

Lepší výsledky byly dosaženy u určitých typů QR kódů tím, že se podpis vložil mimo hlavní datovou část. Tento přístup fungoval velmi dobře například v případě vCard, kde se chování QR kódu po načtení nezměnilo. Tento přístup ale nevyřešil chyby u typů textový obsah a formátu JSON. Společnou nevýhodou obou přístupů bylo to, že některé podpisy mohli zabírat velké množství datového prostoru, který mohl být použitý pro uživatelská data.

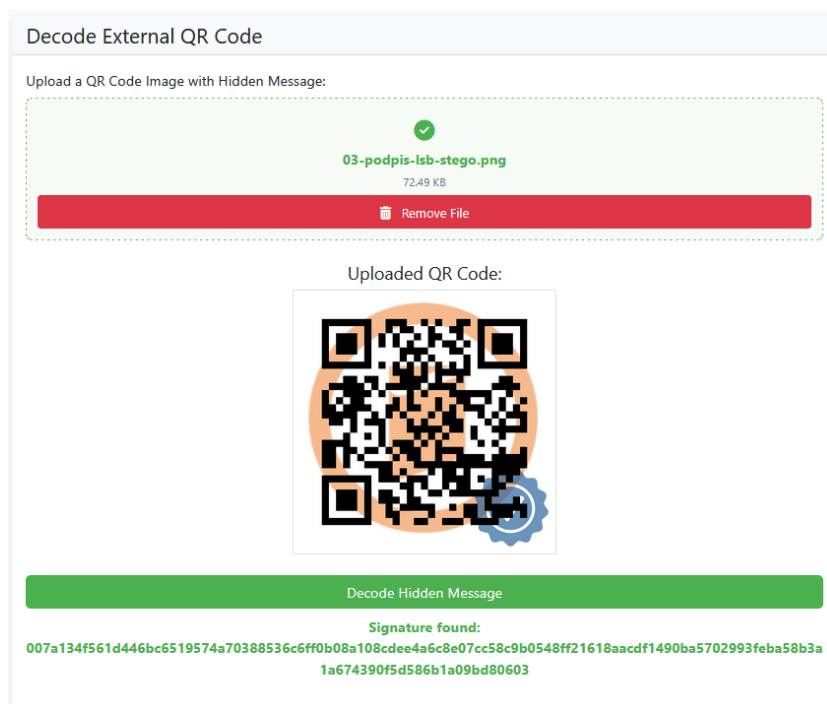


Obrázek 9 – Změna velikosti QR kódu při vložení digitálního podpisu

(zdroj: vlastní)

7.3.2 Digitální podpis uložen pomocí LSB steganografie

Uložení digitálního podpisu pomocí LSB steganografie do samotného obrázku se ukázalo jako účinná metoda, protože neovlivňuje data QR kódu ani jeho čitelnost. Výsledný QR kód lze normálně naskenovat a chování všech typů se také nezměnilo. Při použití speciálního nástroje je možné extrahovat skrytý podpis přímo z originálního obrázku.



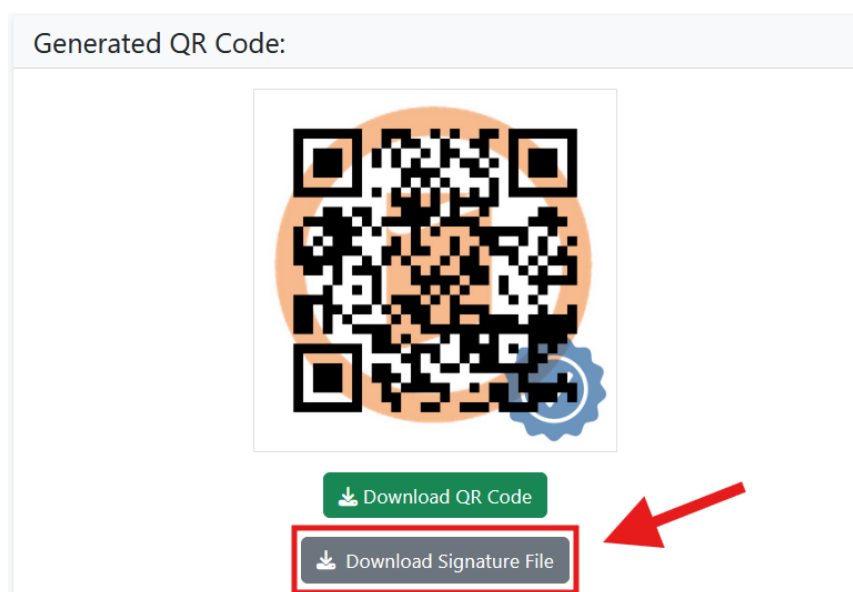
Obrázek 10 – Detekovaný podpis z QR kódu použitím LSB steganografie

(zdroj: vlastní)

Nevýhodou této metody je ztráta dat uložených pomocí steganografie při manipulaci s obrázkem, například vytisknutí. Jelikož tato metoda ukládá data do posledních bitů jednotlivých pixelů, tak o všechny přijdeme, pokud vygenerovaný QR kód vytiskneme.

7.3.3 Digitální podpis jako samostatný soubor

Univerzálně použitelnou metodou se osvědčilo poskytnutí digitálního podpisu ke stažení jako samostatného souboru s příponou .TXT. Jelikož je podpis oddělen do samostatného souboru, zůstává vygenerovaný QR kód zcela nedotčený a jeho data nejsou nijak upravena. Nevýhodou tohoto přístupu je nutnost poskytnutí společně s QR kódem také soubor s digitálním podpisem. Tímto se zvyšuje riziko, že dojde ke ztrátě souboru s podpisem.



Obrázek 11 – Vystavený soubor s podpisem ke stažení

(zdroj: vlastní)

7.3.4 Aplikace vodoznaku

Nejlepší metodou aplikace vodoznaku do QR kódu se prokázalo zachováním barvy vodoznaku a vložení jej mezi bílé pozadí a černé části QR kódu. Tento přístup umožňuje nejvariabilnější nastavení vodoznaku, například nastavení libovolné průhlednosti, a to bez ztráty čitelnosti QR kódu.

Při aplikování vodoznaku přímo na vygenerovaný QR kód, dochází při nižší průhlednosti nebo více vrstvách vodoznaku překrývající se k výraznému snížení čitelnosti až do stavu, kdy QR kód nebylo možné přečíst vůbec.

V rámci testování bylo dále ověřeno, zda má barva vodoznaku vliv na čitelnost. Výsledky ukázaly, že na barvě nezáleží, pokud jsou aplikovány s vyšší průhledností. Naopak při nižší průhlednosti se zachování původní barvy ukázalo jako výhodnější, protože použití barev odstínu šedé rušilo v určitých případech čitelnost QR kódů.



Obrázek 12 – Ukázky dvou přístupů aplikace vodoznaku

(zdroj: vlastní)

Na obrázku 5 je prakticky ukázáno, že volba umístění a barva vodoznaku přímo ovlivňuje čitelnost vygenerovaného QR kódu klasickými čtečkami. Světlejší barva vodoznaku umožňuje použití většího rozsahu nastavení průhlednosti. Všechny ukázky, kromě QR kódu vpravo nahoře, jsou normálně čitelné a lze je naskenovat. V případě vložení vodoznaku přímo na vygenerovaný QR kód je nutné použít vyšší průhlednost, protože obrázek přímo koliduje s černými částmi QR kódu.

7.4 Analýza výhod a nevýhod

Na základě výsledků testování metod zabezpečení QR kódů byly identifikovány hlavní výhody a nevýhody jednotlivých metod. Toto shrnutí sloužilo k následné implementaci řešení ve webové aplikaci.

Tabulka 5 – Analýza výhod a nevýhod metod zabezpečení QR kódů

Metoda	Výhody	Nevýhody
Digitální podpis přímo v QR kódu	• Rychlé ověření bez nutnosti dalších souborů	• Omezený prostor pro uživatelské data • U některých typů ovlivňuje čitelnost
Digitální podpis uložen pomocí LSB steganografie	• Nezabírá místo pro uživatelské data • Neovlivňuje čitelnost	• Funguje pouze v digitální podobě • Ztráta podpisu při tisku nebo zhoršení kvality
Digitální podpis jako samostatný soubor	• Nezabírá místo pro uživatelská data • Neovlivňuje čitelnost	• Nutnost poskytnout dva soubory • Riziko ztráty dokumentu s podpisem
Vodoznak přímo přes QR kód	• Jednoduchá implementace	• Při nižší průhlednosti ovlivňuje čitelnost
Vodoznak mezi vrstvami (pozadí – vodoznak – QR)	• Neovlivňuje čitelnost • Více možností stylizace	• Složitější implementace

(zdroj: vlastní)

7.4.1 Digitální podpis přímo v QR kódu

Hlavní výhodou je možnost ověření pravost dat bez nutnosti dalších souborů. Mezi další výhody se řadí jednoduchost implementace, protože stačí přidat digitální podpis přímo do uživatelských dat.

Nevýhodou, která velmi ovlivnila nevybrání této metody, je, že digitální podpis zabírá nemalou část dat, která může QR kód obsahovat. Navíc přidáním podpisu do dat QR kódu způsobil u některých typů nesprávnou čitelnost čtečkami.

7.4.2 Digitální podpis uložen pomocí LSB steganografie

Využitím LSB steganografie eliminujeme obě nevýhody metody vložení podpisu přímo do QR kódu. Uložený podpis tedy nijak neovlivňuje čitelnost a ani množství uživatelských dat, které mohou být uloženy do QR kódu. Jedinou nevýhodou, kterou tato metoda disponuje je, že uložený digitální podpis se ztratí při vytištění nebo ztrátě kvality obrázku. Vygenerovaný QR kód musí tedy zůstat v digitální podobě v originální kvalitě.

7.4.3 Digitální podpis jako samostatný soubor

Vystavit digitální podpis jako samostatný soubor kombinuje výhody prvních dvou metod, kdy QR kód zůstává čitelný a nezměněný. Ověření je tedy jednoduché a QR kód může být klidně i v tištěné podobě. Použitím této metody je nutné posílat místo jediného souboru, QR kódu, i textový soubor s digitálním podpisem, což zvyšuje riziko ztráty jednoho z nich.

7.4.4 Aplikace vodoznaku

Nejlépe se osvědčil přístup, kdy je vodoznak vložen mezi bílé pozadí a černé prvky QR kódu, protože není nijak ovlivněna čitelnost při libovolném nastavení průhlednosti. Jedinou výjimkou je, pokud vodoznak obsahuje černou barvu. Tehdy může vodoznak splývat s černými prvky QR kódu, a tak znemožnit jeho čitelnost. Naopak při aplikaci vodoznaku přímo na vygenerovaný QR kód dochází k nečitelnosti, obzvláště při nastavení nižší průhlednosti. Nečitelnost QR kódu může být dosažena i při vyšší průhlednosti vodoznaků, pokud použijeme více vrstev, které budeme skládat postupně na sebe.

7.4.5 Přínos vůči hrozbě Quishingu

Prvním nástrojem je aplikace digitálního podpisu, který umožňuje ověřit daný QR kód tak, že pokud kód neobsahuje platný podpis, může být považován za nedůvěryhodný. Podpis může být buď vložen přímo do obrázku QR kódu pomocí LSB steganografie, nebo distribuován jako samostatný soubor. Dalším bezpečnostním prvkem je vizuální vodoznak nebo odznak, který umožňuje uživateli snadno rozpoznat důvěryhodný QR kód již na první pohled. V případech, kdy si uživatel není jistý původem QR kódu, může využít verifikační sekci aplikace, která umožňuje nahrát digitální obrázek kódu a ověřit jeho pravost. Kombinace těchto bezpečnostních prvků poskytuje aplikaci účinnou obranu proti Quishingu.

8 NÁVRHY APLIKACÍ

Tato kapitola se věnuje návrhu a vývoji webové a desktopové aplikace pro generování, dekodování, ověřování, rekonstrukci a zabezpečení QR kódů. Cílem bylo vytvořit uživatelsky přívětivá řešení, která splňují požadavky na jednoduchost, přístupnost a bezpečnost, přičemž respektují technická omezení zvolených platforem. Webová aplikace, nasazená na GitHub Pages, využívá technologie PyScript pro klientské zpracování a je navržena jako responzivní jednosouborové řešení s důrazem na přehlednost a minimalizaci inicializačního času technologie PyScript při startu aplikace. Zároveň webová aplikace klade důraz na bezpečnost, zejména při práci se soukromými klíči. Desktopová aplikace QRebuild doplňuje webovou verzi funkcionalitou rekonstrukce QR kódů z fotografií a je optimalizována pro jednoduché použití i automatizaci. Obě aplikace nabízejí intuitivní rozhraní pro široké spektrum uživatelů.

8.1 Návrh webové aplikace

Hlavními důvody, proč byla aplikace vyvinuta jako webová aplikace, byly snadná dostupnost bez nutnosti instalace jakéhokoliv softwaru a jednoduchost použití. Aplikace musí jednoduché generování, dekodování, ověřování a zabezpečení QR kódů. Webové technologie zajišťují kompatibilitu s většinou běžných zařízení a operačních systémů. Důraz byl kladen také na responzivní design, aby byla aplikace použitelná na libovolně velkém zařízení.

Významnou roli při návrhu aplikace sehráli také omezení platformy GitHub Pages, na kterých lze vystavit pouze statické weby bez možnosti spouštění jakýchkoliv operací na serveru. Z tohoto důvodu musela být všechna logika aplikace řešena na uživatelské straně přímo ve webovém prohlížeči. Proto byla zvolena technologie PyScript, která umožňuje spouštění Python scriptů přímo v prohlížeči. Jelikož webová aplikace pracuje i s privátními uživatelskými klíči, které slouží pro digitální podepsání vygenerovaného QR kódu, tak se v tomto případě jedná i o výhodu, jelikož se všechny akce provádí na uživatelské straně, takže klíč při použití neopustí uživatelské zařízení a není nikde nahrán a uchován.

Protože aplikace disponuje více funkcionalitami a technologie PyScript se musí inicializovat při každém načtení stránky, byla zvolena metoda jednosouborového HTML souboru rozděleného do sekcí, které odpovídají jednotlivým funkcionalitám aplikace. Tímto redukuje načítací a inicializační čas PyScriptu, protože se inicializace provádí pouze

jednou, a to při prvním načtení webové stránky. Sekce nám navíc přehledně rozdělí webovou aplikaci na jednotlivé funkční části, takže web zůstává přehledný.

8.1.1 Požadavky na aplikaci

Před samotným návrhem a implementací webové a desktopové aplikace bylo nutné stanovit jasné požadavky, které musí výsledná řešení splňovat. Tyto požadavky byly definovány na základě uživatelské přívětivosti, technických omezení platformy GitHub Pages a bezpečnostních požadavků. Požadavky byly rozděleny na funkcionální a nefunkcionální, které napomáhají pokrýt všechny klíčové aspekty návrhu a vývoje.

Funkcionální požadavky určují konkrétní funkcionality, které musí aplikace poskytovat uživatelům, tedy jaké akce musí být možné provádět, jaké vstupy systém očekává a jaké výstupy poskytuje. Níže uvedená tabulka shrnuje všechny hlavní funkční požadavky definované pro tuto aplikaci.

Tabulka 6 – Funkcionální požadavky

ID	Funkcionální požadavek
F1	Uživatel může vygenerovat QR kód pro různé typy dat
F2	Uživatel si může zvolit, zda bude aplikován digitální podpis při generování QR kódu pomocí LSB steganografie
F3	Uživatel si může zvolit úroveň korekce chyb při generování QR kódu
F4	Webová aplikace umožňuje dekodování podpisu ukrytého pomocí LSB steganografie v QR kódu
F5	Webová aplikace umožňuje ověření pravosti QR kódu na základě podpisu a veřejného klíče
F6	Webová aplikace umožňuje stažení souborů samostatného podpisu a vygenerovaného QR kódu
F7	Desktopová aplikace umožňuje rekonstrukci QR kódu z nahrané fotografie
F8	Uživatelské rozhraní se mění podle zvolené sekce funkcionality webové aplikace

(zdroj: vlastní)

Nefunkcionální požadavky popisují omezení a očekávání od chování aplikace, která nesouvisí přímo s jednotlivými funkcemi. Týkají se například použitelnosti, bezpečnosti, výkonu nebo platformních omezení. Tyto požadavky byly rovněž zásadní při návrhu řešení obzvláště s ohledem na platformu GitHub Pages.

Tabulka 7 – Nefunkcionální požadavky

ID	Nefunkcionální požadavek
N1	Webová aplikace nesmí odesílat privátní klíče mimo uživatelské zařízení
N2	Webová aplikace musí být přístupná bez instalace tedy pouze přes webový prohlížeč
N3	Webová aplikace musí být plně funkční na uživatelské straně bez serveru
N4	Rozhraní webové aplikace musí být responzivní a použitelné na různých zařízeních
N5	Inicializace PyScriptu musí proběhnout pouze jednou při načtení webové stránky
N6	Desktopová aplikace musí být plně funkční na obou operačních systémech
N7	Uživatelské rozhraní aplikací musí být jednoduché, intuitivní a srozumitelné

(zdroj: vlastní)

8.1.2 Cíle webové aplikace

Hlavním cílem webové aplikace bylo umožnit uživatelům jednoduché generování QR kódů o mnoha typech s možností zabezpečení a následné ověření autenticity. Aplikace by měla být přístupná na co nejvíce zařízeních bez nutnosti instalace dodatečného softwaru, proto byla zvolena volba webových technologií. Zvolením této metody nezáleží na operačním systému ani na typu zařízení, ale na tom, jestli má zařízení přístup k internetu.

Mezi zásadní požadavky patřila aplikace dvou metod zabezpečení, digitálního podpisu a vodoznaku. Obě dvě metody byly navrženy na základě výsledků z testování tak, aby zachovali vysokou míru čitelnosti QR kódů čtečkami i v případě vytištění a nezabírali místo v uživatelském datovém prostoru.

Dalšími neméně důležitými cíli bylo vytvořit webovou aplikaci tak, aby byla jednoduchá a intuitivní, podporovala široké spektrum typů QR kódů a responzivní, aby aplikace byla použitelná na různě velkých zařízeních. Všechny zmíněné prvky museli být napsané a vytvořené tak, aby respektovali omezení platformy GitHub Pages, na které bude

celá webová aplikace nasazená. Tyto omezení kladou důraz na to, aby všechny operace byly vykonávány na uživatelské straně.

8.1.3 Prostředí GitHub Pages

Pro nasazení webové aplikace bylo zvolena platforma GitHub Pages, která umožňuje bezplatné publikování statických webových stránek přímo z repozitáře na GitHubu. Výhodami hostování webové aplikace na GitHub Pages je jednoduchá správa webu a celý proces nasazení, automatické poskytnutí HTTPS certifikátu a bezplatné hostování.

Platforma GitHub Pages podporuje pouze statické webové stránky, což znamená, že nelze provádět operace na serveru ani využívat databázi. Tohle omezení mělo zásadní vliv na návrh a celý vývoj aplikace, která musela být ve výsledku navržena a zrealizována tak, aby se všechny operace vykonávali na uživatelské straně. Tenhle fakt není tak moc omezující, jelikož aplikace může pracovat s privátním klíčem uživatele, kdy je výhodnější provádět operace na straně uživatele. Tímto přístupem klíč nikdy neopustí uživatelské zařízení. Pro tento účel byl vybrána technologie PyScript, která umožňuje vyžadované vykonávání operací na uživatelské straně, kdy spouští Python skripty přímo v prohlížeči.

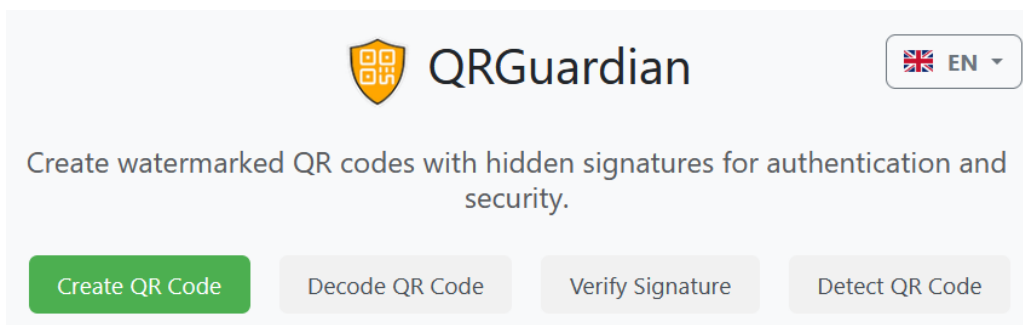
Výsledná webová aplikace pracuje s omezeními platformy GitHub Pages a současně využívá jejich výhod. V závislosti na omezeních platformy nebylo možné realizovat funkcionalitu detekování QR kódů z fotografií. Proto byla vytvořena a vydána samostatná desktopová aplikace QRebuild, která tuhle funkci nahrazuje. Webová aplikace odkazuje automaticky vždy na nejnovější vydanou verzi aplikace, kterou lze z webu stáhnout a použít bez nutnosti instalace.

8.1.4 Rozdělení aplikace do funkčních sekcí

Pro vytvoření webové aplikace tak, aby byla co nejvíce přehledná a uživatelsky přívětivá, byla rozdělena do čtyř základních sekcí, které odpovídají funkčním prvkům aplikace. Každá sekce obsahuje pouze ovládací prvky, které k dané sekci náleží a sekce je využije.

První sekcí je generování QR kódu, která slouží uživateli k vygenerování různých typů QR kódů s možností aplikace digitálního podpisu a vodoznaku. Druhá sekce slouží pro dekodování podpisu, který byl uložen pomocí LSB steganografie do QR kódu. Ve třetí sekci si může uživatel ověřit pravost QR kódu, kde je potřeba nahrát QR kód s aplikovaným podpisem pomocí steganografie nebo separátní textový soubor s podpisem a veřejný klíč

uživatele, který nahraný QR kód podepsal. Čtvrtou a poslední sekci je detekování QR kódu z fotografie, která obsahuje pouze odkazy na desktopovou aplikaci QRebuild, která tuhle funkci nahrazuje. Tímto rozdělením zajistíme přehlednější prostředí a snazší orientaci v aplikaci a zároveň umožňuje snadné rozšíření aplikace o další funkce.



Obrázek 13 – Sekce ve webové aplikaci

(zdroj: vlastní)

8.2 Návrh desktopové aplikace QRebuild

Uživatelské rozhraní desktopové aplikace QRebuild bylo navrženo s důrazem na jednoduchost, přehlednost a hlavně funkčnost. Hlavním cílem vytvoření aplikace bylo omezení platformy GitHub Pages a umožnit rekonstrukci QR kódu z fotografií do digitální podoby jak běžným uživatelům pomocí verze s uživatelským rozhraním tak i pokročilejším uživatelům preferující práci s příkazovou řádkou. Zároveň byl kladen důraz na umožnění použití aplikace v automatizovaných skriptech.

Pro návrh a realizaci aplikace byl využitý programovací jazyk Python. GUI verze aplikace byla postavena na frameworku PyQt6 a CLI verze využívá knihovnu argparse, pro jednoduché zpracování parametrů příkazu.



Obrázek 14 – Wireframe aplikace QRebuild

(zdroj: vlastní, zhotoveno pomocí <https://wireframe.cc>)

9 IMPLEMENTACE WEBOVÉ APLIKACE

Tato kapitola se věnuje implementaci webové aplikace, která vychází ze získaných poznatků a zkušeností z výsledků testování v kapitole 7. Výsledná aplikace byla vyvinuta pomocí webových technologií v kombinaci s technologií PyScript, který umožňuje spouštění Python scriptů přímo ve webovém prohlížeči. Aplikace byla nasazená prostřednictvím platformy GitHub Pages, která umožňuje bezplatné nasazení aplikace na úkor určitých omezení. Webová aplikace je volně dostupná na adrese: <https://pipousek.github.io/QRGuardian>.



Obrázek 15 – Odkaz na webovou aplikaci

(zdroj: vlastní)

V důsledku těchto omezení byla vytvořena desktopová aplikace QRebuild, která nahrazuje funkcionalitu detekce QR kódu z fotografií, na kterou je potřeba vyššího serverového výkonu, který platforma GitHub Pages neposkytuje. Popis aplikace včetně automatického procesu sestavení a distribuce je popsán v následující kapitole.

9.1 Uživatelské rozhraní

Uživatelské rozhraní bylo navrženo a vytvořeno tak, aby umožnilo uživatelům snadné generování, dekodování a ověřování QR kódů bez nutnosti pokročilých znalostí. Webová aplikace je rozdělena do čtyř přehledných sekcí, kde každá sekce je vytvořena tak, aby byla intuitivní a bylo jednoduché náležitou funkci použít. Uživatelé mohou v jednotlivých sekcích zadat potřebná data, nahrát soubory pomocí funkce Drag & Drop nebo procházením souborového systému a vybráním požadovaného souboru nebo si stáhnout vygenerovaný QR kód a digitální podpis. Hlavní důraz byl kladen na jednoduchost ovládání a přístupnost z různých typů zařízení.

9.1.1 Webové rozhraní

Webové rozhraní aplikace je navrženo jako jednostránková aplikace (SPA) primárně kvůli omezením technologie PyScript, kde využitím SPA zajistíme, že se prostředí bude inicializovat pouze jednou při startu aplikace. Při použití metody vícestránkové aplikace (MPA) by se prostředí PyScriptu inicializovalo při každé změně stránky. Aby webová aplikace zůstala přehledná, tak jednotlivé sekce byly rozdělené do sekcí. V jednu chvíli může být zobrazená pouze jedna sekce, kterou si vybere uživatel z horního menu. Všechny sekce obsahují totožné záhlaví a zápatí. V záhlaví se nachází logo a název aplikace, výběr jazyku aplikace a menu pro vývěr zobrazené sekce. V zápatí jsou odkazy na GitHub repozitáře a copyright text.

Sekce pro generování QR kódů je rozdělena do dvou bloků. První blok obsahuje rozbalovací seznam pro výběr typu generovaného QR kódů, kde každý typ obsahuje několik textových polí pro potřebné informace k vybranému typu QR kódu. Následují dvě vstupní pole v režimu Drag & Drop, kde stačí daný soubor přetáhnout na dané pole, nebo po stisknutí se otevře souborový systém pro výběr souboru. První vstupní pole je pro soukromý klíč, který slouží pro podepsání obsahu QR kódu. Pod tímto vstupním polem se ještě nachází zaškrtnávací pole, které slouží pro určení, jestli se má QR kód podepsat. Druhé vstupní pole slouží pro nahrání obrázku, který bude použitý jako vodoznak. Po nahrání obrázku se pod vstupním polem zobrazí náhled nahraného obrázku a posuvník pro zvolení průhlednosti vodoznaku. Další částí této sekce je řada čtyř přepínačů, které slouží pro nastavení úrovně korekce chyb (ECL), která se použije u generování. Na konci prvního bloku je tlačítko, pomocí kterého vygenerujeme QR kód z vyplněných informací. Druhý blok slouží pro zobrazení vygenerovaného QR kódu a sadu tří tlačítek pro dekodování digitálního podpisu aplikovaného pomocí LSB steganografie a stažení vygenerovaného QR kódu jako PNG obrázek a podpisu v textovém souboru.

 QRGuardian

EN

Create watermarked QR codes with hidden signatures for authentication and security.

Create QR Code

Decode QR Code

Verify Signature

Detect QR Code

Create QR Code

QR Code Type:

Website URL

Website URL:

https://pyscript.com

Digital Signature

Upload Private Key (ED25519 or RSA):

Drag & drop your file here or click to browse

No key loaded (supports ED25519 and RSA)

☒ Use steganography (hide signature in QR code)

Watermark Image (optional):

Drag & drop your file here or click to browse

Error Correction Level:

☐ Low (7%)

☐ Medium (15%)

☐ Quartile (25%)

☒ High (30%)

Higher levels allow QR codes to remain readable even when partially damaged but reduce data capacity.

Create QR Code

Ready to generate QR code

Obrázek 16 – Sekce pro generování QR kódu

(zdroj: vlastní)

Sekce pro dekodování QR kódu obsahuje pouze jedno vstupní pole, ve formátu Drag & Drop, pro nahrání obrázku QR kódu a tlačítko pro dekodování podpisu uloženého v obrázku pomocí LSB steganografie. Nahráný obrázek se zobrazí mezi vstupním polem a tlačítkem. Pokud nahráný obrázek obsahuje podpis, zobrazí se pod tlačítkem.

The screenshot shows the QRGuardian web application interface. At the top, there is a logo and the text "QRGuardian". Below it, a subtitle reads "Create watermarked QR codes with hidden signatures for authentication and security." There are four main buttons: "Create QR Code", "Decode QR Code" (highlighted in green), "Verify Signature", and "Detect QR Code". The "Decode QR Code" section is active, showing a "Decode External QR Code" header. Below this, there is a prompt "Upload a QR Code Image with Hidden Message:" followed by a large dashed box for file upload. Inside the box, there is a cloud icon and the text "Drag & drop your file here or click to browse". Below the upload area, there is a section labeled "Uploaded QR Code:" with a text input field containing "Uploaded QR Code". At the bottom of this section, there is a green button labeled "Decode Hidden Message" and a status message "No QR code to decode" in green text.

Obrázek 17 – Sekce pro dekodování podpisu z obrázku QR kódu

(zdroj: vlastní)

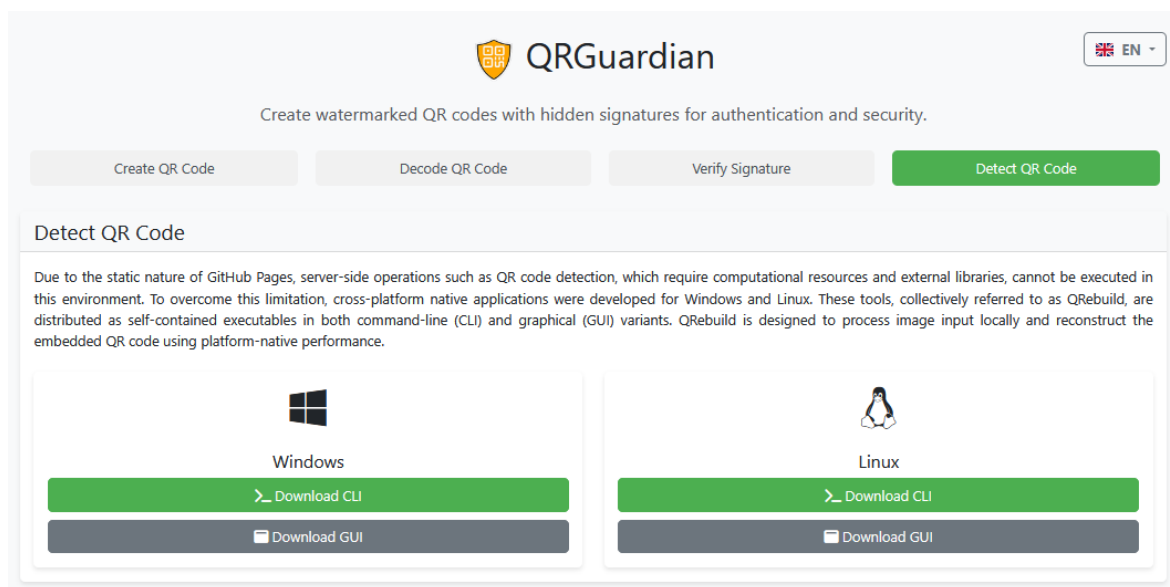
Sekce pro ověření QR kódu obsahuje tři vstupní pole ve formátu Drag & Drop. První pole slouží pro nahrání obrázku QR kódu, druhé pro textový soubor s digitálním podpisem ve formátu TXT a třetí pro veřejný klíč, který slouží k ověření podpisu. Pod vstupními poli je tlačítko, které na základě nahraných souborů ověří platnost digitálního podpisu, který najde buď přímo v obrázku pomocí LSB steganografie, nebo z nahraného textového souboru. Pokud podpis obsahuje jak obrázek i soubor, ověří oba dva podpisy. Výsledek ověření se zobrazí pod tlačítkem.

The screenshot shows the QRGuardian web application interface. At the top, there is a logo and the text "Create watermarked QR codes with hidden signatures for authentication and security." Below this, there are four buttons: "Create QR Code", "Decode QR Code", "Verify Signature" (highlighted in green), and "Detect QR Code". The "Verify QR Code Signature" section is active, showing three upload areas: "Upload a QR Code Image to Verify:", "Upload Signature File (.txt):", and "Upload Public Key (ED25519 or RSA):". Each area has a dashed border and a cloud icon with an upward arrow, and the text "Drag & drop your file here or click to browse". Below these areas, it says "No key loaded (supports ED25519 and RSA)". At the bottom, there is a "Verify Signature" button and a green message "No QR code to verify".

Obrázek 18 – Sekce ověření QR kódu

(zdroj: vlastní)

Sekce pro detekování QR kódů z fotografií obsahuje pouze vysvětlující text, proč nebylo možné vytvořit tuto funkcionalitu přímo ve webové aplikaci a jako náhrada byla vytvořena multiplatformní aplikace QRebuild. Pod tímto textem se nachází dvě karty, každá pro jeden operační systém, kde každá karta obsahuje dvě tlačítka pro stažení aplikace ve formátu CLI nebo GUI. Tlačítka odkazují přímo na soubor, který je dostupný pomocí GitHub Releases, a po stisknutí se nejnovější verze aplikace začne automaticky stahovat.



Obrázek 19 – Sekce detekování QR kódu

(zdroj: vlastní)

9.1.2 Způsob ovládání

Při vytváření webové aplikace byl kladen důraz na jednoduché, intuitivní a přehledné ovládání. Při generování QR kódu je prvním krokem výběr typu QR kódu. Po výběru se zobrazí příslušné textové pole, které obsahují zástupné texty, aby bylo jednodušší je vyplnit. Po vyplnění známých informací, může uživatel nahrát soukromý klíč a obrázek pro vodoznak. V případě, že uživatel ani jedno nemá, nebo jej nechce použít, aplikace funguje i bez těchto ochranných prvků a QR kód se vygeneruje.

Ověření QR kódu funguje pouze na digitální obrázky, nikoliv na fotografie. Pro ověření je potřeba mít obrázek s QR kódem, kde je aplikovaný digitální podpis pomocí LSB steganografie a veřejný klíč autora QR kódu nebo samostatný textový soubor s podpisem. Pokud obrázek obsahuje digitální podpis textový soubor není potřeba, ale pokud bude také nahraný, proběhne ověření podpisů z obou zdrojů.

9.2 Architektura aplikace

Architektura webové aplikace byla navržena a vytvořena na základě zachování modularity a snadného rozšíření a vylepšování jednotlivých sekcí. Jednotlivé funkcionality jsou rozdělené do samostatných modulů, což usnadňuje snadné testování, provádění úprav a vývoj. Vzhledem k omezením platformy GitHub Pages je veškerá logika aplikace vykonávána přímo na straně uživatele za pomoci technologie PyScript, která umožňuje spouštění Python skriptů přímo v prohlížeči. Pro dynamickou úpravu uživatelského rozhraní je využitý JavaScript, který je použitý například pro překlady aplikace nebo změnu stylu po nahrání souboru do vstupních polí.

9.2.1 Modulární struktura

Modulární struktura zjednodušuje udržitelnost a snadnou rozšiřitelnost aplikace. Tato struktura využívá toho, že funkcionality aplikace jsou rozdělené do samostatných skriptů.

Frontendová část, využívající JavaScript, slouží pro veškerou interakci s uživatelem a vizuální práci s aplikací.

- `main.js` – Inicializace aplikace
 - Hlavní inicializační skript frontendové části, kde se po načtení dokumentu `DOMContentLoaded` vyčistí všechny vstupní pole, aktualizují se odkazy pro stažení QRbuild aplikace, nastaví se tlačítko pro stažení vygenerovaného QR kódu, inicializuje se navigace mezi sekcemi aplikace a spustí se inicializace pro vylepšení UI pro zařízení s dotykovými displeji.
- `navigation.js` – Navigace mezi sekcemi
 - Skript starající se o správné přepínání mezi jednotlivými sekcemi aplikace. Při stisknutí na libovolné tlačítko navigace se zobrazí příslušná sekce.
- `downloadLinks.js` – Nastavení odkazů a tlačítek pro stahování
 - Skript zajišťuje aktualizaci stávajících odkazů v sekci pro detekci na nejnovější vydanou verzi, kde se nachází odkazy na aplikaci QRbuild. Dále slouží i pro generování odkazu pro stažení vygenerovaného QR kódu.

- fileUpload.js – Nastavení vstupních polí pro Drag & Drop
 - Skript zajišťuje veškerou správu vstupních polí ve formátu Drag & Drop. Zajišťuje vizuální odezvu při přetahování i nahrání souboru, přidání tlačítka pro odebrání nahraného souboru a zobrazení názvu a velikosti souboru.
- translations.js – Vícejazyčná podpora aplikace
 - Skript starající se o vícejazyčnost aplikace, kde překlady načítá z předem definovaných JSON souborů s překlady. Pro aplikování překladů hledá elementy s atributem data-i18n, který obsahuje cestu k překladu, která je uvedena v JSON souborech.
- ui.js – Vylepšení UI
 - Script aplikuje drobná vylepšení pro zařízení s dotykovým zařízením, kde upravuje vzhled elementů s atributem py-2, kde přidává větší vertikální odsazení.

Backendová část, využívající Python skripty spouštěné pomocí PyScript, slouží pro logiku aplikace.

- main.py – Inicializační Python skript
 - Hlavní inicializační skript, který po spuštění webové aplikace nastavuje počáteční stav UI, jako je skrytí načítací animace, inicializace výchozích textů a obsluha událostí click a change u prvků rozhraní aplikace.
- event_handlers.py – Obsluha událostí v PyScriptu
 - Script přímo navazuje na skript main.py, kdy tento skript obsahuje funkce, které se volají při obsluze událostí. Obsahuje obslužné funkce pro generování, dekodování, ověřování a podepisování QR kódů a náhled vodoznaku.
- crypto.py – Správa klíčů, podepisování a ověřování
 - Skript zajišťující zpracování kryptografických klíčů a práci s digitálním podpisem. Skript umožňuje nahrání soukromého i veřejného klíče a detekci jeho typu. Dále obsahuje funkce pro podepsání a ověření QR kódů.

- qr_content.py – Obsah generovaných QR kódů
 - Skript obsahuje seznam podporovaných typů QR kódu. Každý typ přečte potřebné informace z příslušných vstupních textových polí a vytvoří obsah QR kódu daného typu.
- qr_generator.py – Generování QR kódu a aplikace vodoznaku
 - Skript obsahuje funkce pro generování QR kódu s možností aplikace vodoznaku a ikony, která signalizuje, že QR kód byl digitálně podepsán. Dále zpracovává výběr úrovně korekce chyb, kterou aplikuje při generování.
- stego.py – LSB steganografie
 - Skript zajišťuje kompletní práci se steganografií. Skript obsahuje funkce pro zakódování podpisu do obrázku a dekodování z něj pomocí techniky LSB steganografie.
- ui_handlers.py – Ovládání UI
 - Skript zajišťuje aktualizaci prvků UI, které slouží pro komunikaci s uživatelem a dávají mu informaci o stavu jednotlivých akcí. Obsahuje funkce pro aktualizaci stavových hlášek, načtení a zpracování nahraných souborů a zobrazení výsledků ověření QR kódu.
- base64_signed_badge.py – Odznak při správném vytvoření digitálního podpisu
 - Odznak ve formátu Base64, který je aplikován pouze v případě, když je QR kód digitálně podepsán. Jedná se o jednoznačný grafický ukazatel.

9.2.2 Integrace s PyScript

Aby bylo možné spouštět Python skripty přímo v prohlížeči, je nutné integrovat technologii PyScript do HTML souboru. Pro integrování je potřeba v <head> elementu nalinkovat PyScript CSS a PyScript JS jako typ modul a následným použitím <py-script> elementů nebo externích Python skriptů. Pro využití více skriptů je potřeba vytvořit konfigurační soubor ve formátu TOML (Tom's Obvious Minimal Language). Níže je uvedená ukázka kódu integrace technologie PyScript v HTML souboru.

```
<head>
  <link rel="stylesheet"
    href="https://pyscript.net/releases/2025.3.1/core.css" />
  <script type="module"
    src="https://pyscript.net/releases/2025.3.1/core.js"></script>
  <!-- Rest of the code -->
</head>

<body>
  <py-config src="pyscript.toml"></py-config>
  <py-script src="scripts/main.py"></py-script>
  <!-- Rest of the code -->
</body>
```

Konfigurační soubor obsahuje seznam všech balíčků, které má PyScript načíst při startu z prostředí Pyodide. V případě platformy GitHub Pages je potřeba, aby všechny Python skripty byly v kořenovém adresáři běhového prostředí PyScriptu, jinak PyScript skripty nevidí a nedokáže s nimi pracovat. Tohoto dosáhneme pomocí konfiguračního souboru PyScriptu, kde do vícenásobného bloku fetch uvedeme názvy skriptů, cestu ke složce se skripty a cílovou cestu kam se mají nakopírovat. Názvy skriptů napíšeme do proměnné `files`, cestu ke složce se skripty napíšeme do proměnné `from` a cílovou cestu do proměnné `to_folder`. Pro načtení konfiguračního skriptu a hlavního Python skriptu je nutné na začátku elementu `<body>` načíst oba dva soubory. Konfigurační skript načteme elementem `<py-config>` a hlavní Python script elementem `<py-script>`. Následující ukázka ukazuje příklad zápisu konfiguračního souboru, který je využitý pro tuto webovou aplikaci.

```
packages = ['pillow', 'numpy', 'pyodide-py', 'qrcode', 'cryptography']

[[fetch]]
files = ["main.py", "event_handlers.py", "crypto.py",
        "qr_generator.py", "stego.py", "ui_handlers.py",
        "base64_signed_badge.py", "qr_content.py"]
from = 'scripts'
to_folder = '.'
```

9.3 Implementace digitálního podpisu

Hlavní klíčovou součástí zabezpečení vygenerovaných QR kódů ve vytvořené aplikaci je digitální podepsání obsahu QR kódu. Digitální podpis slouží pro ověření pravosti vygenerovaného QR kódu a zajišťuje další úroveň zabezpečení proti neoprávněným změnám po vygenerování. Zároveň ujišťuje, že QR kód byl vygenerovaný oprávněnou osobou nebo aplikací. Vytvořená webová aplikace umožňuje aplikování digitálního podpisu přímo do

obrázku vygenerovaného QR kódu použitím techniky LSB steganografie a zároveň vytváří textový soubor s digitálním podpisem ke stažení společně s vygenerovaným QR kódem. Tyto dvě možnosti dávají uživateli možnost zvolit si způsob aplikace digitálního podpisu na základě požadavků na distribuci a čitelnosti. Aplikace dále nabízí i ověření vygenerovaných QR kódů na základě obsahu nahraného QR kódu a digitálního podpisu.



Obrázek 20 – Podepsaný QR kód bez vodoznaku

(zdroj: vlastní)

K implementaci digitálního podpisu byla využita asymetrická kryptografie, specificky algoritmy RSA nebo Ed25519, které jsou považovány za bezpečné a jsou běžně používány. Výběr algoritmu záleží na typu soukromého klíče, který uživatel nahraje ve webové aplikaci za účelem podepsání vygenerovaného QR kódu. Všechny operace s nahraným soukromým i veřejným klíčem aplikace zpracovává pomocí knihovny cryptography. Všechny kryptografické operace probíhají přímo ve webovém prohlížeči uživatele v prostředí PyScript, takže žádné data nejsou posílány na server. Tento přístup zajišťuje vyšší ochranu při práci se soukromým klíčem.

9.3.1 Integrace digitálního podpisu pomocí LSB steganografie

Jedním ze způsobů, jak distribuovat digitální podpis společně s vygenerovaným QR kódem bez nutnosti dalšího souboru, je jeho vložení přímo do obrázku QR kódu použitím techniky LSB steganografie. Využitím této techniky aplikujeme digitální podpis přímo do vygenerovaného obrázku tak, že není čitelný pouhým okem, ale je potřeba aplikace, která z obrázku podpis získá. Problém nastává, když vygenerovaný QR kód projde ztrátovou kompresí nebo je vytisknutý na papír. V tomto případě již nemáme přístup k digitálním datům jednotlivých pixelů a podpis se tedy ztratí.

Aplikování podpisu touto metodou se provádí jako poslední krok ve fázi generování QR kódu. Funkce nejprve zkontroluje, jestli je zaškrtnuté pole pro využití steganografie v uživatelském rozhraní. Digitální podpis je nejprve transformován do binární podoby, ke které je přidána univerzální koncová řada bitů, která označuje konec dat digitálního podpisu. Jednotlivé bity digitálního podpisu jsou postupně vkládány do jednotlivých kanálů RGB pixelů vygenerovaného obrázku QR kódu. Pro implementaci je využita knihovna numpy a Pillow, kde Pillow slouží pro načtení dat obrázku a numpy pro reprezentaci obrázku jako 2D pole. Celý tento proces obstarává funkce ve skriptu stego.py.

9.3.2 Distribuce digitálního podpisu pomocí textového souboru

Nezávisle na zvolení použití techniky LSB steganografie pro distribuci digitálního podpisu, je podpis ke QR kódu distribuován i jako samostatný textový soubor ve formátu .txt. Tento přístup nám umožní širší spektrum distribuce, protože vygenerovaný QR kód můžeme i vytisknout nebo na něm provést i ztrátovou kompresi. Nevýhodou je nutná distribuce textového souboru společně s QR kódem.

Vytvoření textového souboru s digitálním podpisem je prováděno obdobně jak aplikování podpisu pomocí LSB steganografie v posledním kroku ve fázi generování QR kódu. Vytvoření souboru obstarává funkce ze skriptu crypto.py, které se předá jen data digitálního podpisu. Po vytvoření souboru je soubor vystaven ve webové aplikaci přímo pod vygenerovaným QR kódem a je dostupný ke stažení.

9.3.3 Algoritmus digitálního podpisu

Digitální podpis je vytvořen a aplikován ve fázi generování QR kódu pouze, pokud uživatel při generování nahrál soukromý klíč ve formátu RSA nebo Ed25519. Proces podepisování spočívá ve vytvoření HASHe z dat obsažených ve QR kódu a následné zašifrování vytvořeného HASHe pomocí nahraného soukromého klíče. Výsledkem je digitální podpis ve formě bajtového řetězce, který je před aplikací převeden do hexadecimálního formátu. Jestli má být vygenerovaný QR kód podepsán rozhoduje rozhodovací logika uvedena v následující ukázce kódu.

```
# Check if private key is loaded
private_key_input = document.querySelector("#private-key-input")
if private_key_input.files.length > 0:
    # Key is loaded - sign the content
    sign_and_generate_qr(content)
else:
    # No key - generate without signature
    generate_qr_with_content(content, None)
```

Pokud byl při generování nahraný soukromý klíč, zavolá se funkce, která obsah vygenerovaného QR kódu podepíše. Tato funkce je definována ve skriptu `qr_generator.py`. Funkce si nejprve načte nahraný soukromý klíč zjistí obsah a typ nahraného klíče. Následně se zavolá funkce definována ve skriptu `crypto.py`, která vytvoří digitální podpis na základě obsahu vygenerovaného QR kódu a nahraného klíče. Pokud nahraný soukromý klíč je typu Ed25519, provede se jednoduché a moderní podepsání přímo na originálních datech. V případě, že byl nahraný klíč typu RSA, je při podepsání použité schéma PSS (Probabilistic Signature Scheme) s interním použitím HASH funkce SHA-256. Pokud podepsání proběhne bez problémů, funkce vrátí digitální podpis ve formě bajtového řetězce. Následující ukázka popisuje vytvoření digitálního podpisu z obsahu QR kódu.

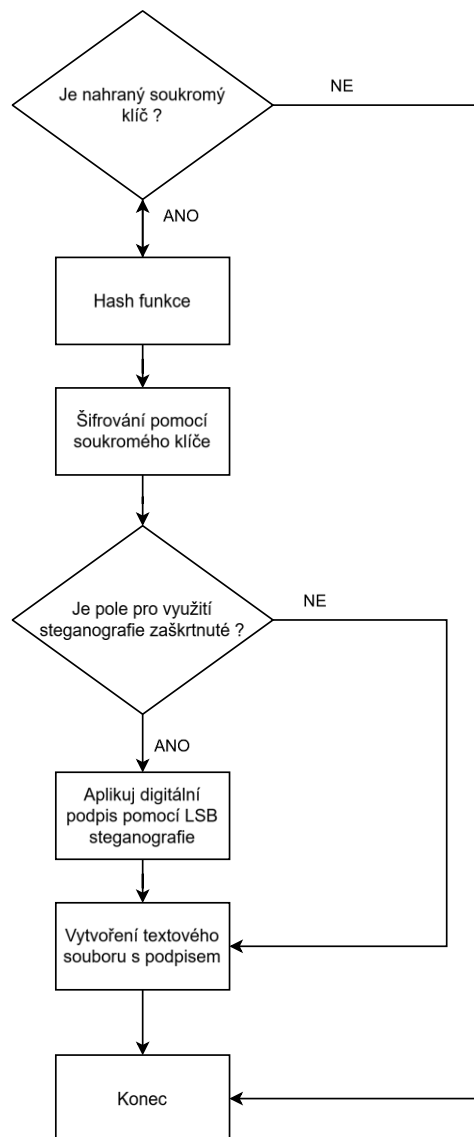
```
if key_type == 'ed25519':
    signature = private_key.sign(content.encode('utf-8'))
elif key_type == 'rsa':
    # Sign the raw content, let library hash internally
    signature = private_key.sign(
        content.encode('utf-8'),
        padding.PSS(
            mgf=padding.MGF1(hashes.SHA256()),
            salt_length=padding.PSS.MAX_LENGTH
        ),
        hashes.SHA256()
    )
```

Na závěr generování QR kódu proběhne vytvoření textového souboru s digitálním podpisem a aplikování podpisu přímo do obrázku QR kódu, pokud byla zaškrtnutá volba použití techniky LSB steganografie. Rozhodování probíhá na základě, jestli funkci byl předán digitální podpis. Tento krok je definovaný ve funkci ve skriptu `qr_generator.py`. Rozhodovací logika je k vidění na úryvku kódu níže.

```

# Encode the signature in the image if it exists
if signature and use_steganography:
    final_image = encode_signature_in_image(qr_image, signature,
    method=steg_method)
    # Create and show signature download link
    create_signature_download(signature)
elif signature and not use_steganography:
    final_image = qr_image
    create_signature_download(signature)
else:
    final_image = qr_image
# Hide signature download link if no signature
document.querySelector("#signature-download-container").style.display =
    "none"

```



Obrázek 21 – Diagram aplikace digitálního podpisu

(zdroj: vlastní, zhotoveno pomocí <https://www.drawio.com>)

9.4 Implementace vodoznaku a odznaku do QR kódu

Jeden z důležitých prvků zabezpečení z pohledu vizuální identifikace správnosti QR kódu je aplikace vodoznaku bez narušení čitelnosti samotného QR kódu. Tato funkcionality má za cíl zvýšit bezpečnost a důvěryhodnost vygenerovaného QR kódu tím, že uživatele vizuálně upozorní, že QR kód byl digitálně podepsaný a vytvořený oficiálním způsobem. Vytvořená webová aplikace používá aplikaci vodoznaku nejen pro aplikování nahraného obrázku od uživatele, ale i pro začlenění speciálního odznaku, který signalizuje, že vygenerovaný QR kód byl digitálně podepsán.



Obrázek 22 – QR kód s vodoznakem bez podpisu

(zdroj: vlastní)

9.4.1 Integrace vodoznaku do QR kódu

Integrace vodoznaku probíhá jako součást generování QR kódu, těsně před jeho vystavením na webové aplikaci. Vodoznak představuje vizuální bezpečnostní prvek, který je umístěn mezi bílé pozadí a černé části QR kódu, aby nebyla narušena jeho čitelnost.

Implementace je realizována pomocí knihovny Pillow, která slouží k načtení, úpravě, konverzi a vytváření obrázků. Základní kroky při práci s obrázkem pro vodoznak je konverze do formátu s průhledností RGBA, úprava velikosti a pozice, změna průhlednosti a následné vložení do výsledného obrázku jako vrstvy. Aplikaci vodoznaku obstarává funkce ve skriptu `qr_generator.py`.

9.4.2 Integrace odznaku do QR kódu

Integraci odznaku do vygenerovaného QR kódu neovládá uživatel přímou volbou, ale je aplikován pokaždé, když je vygenerovaný QR kód digitálně podepsán, tedy je nahrán soukromý klíč při generování. Hlavními účely odznaku je vizuální upozornění, že QR kód byl digitálně podepsán a zvýšení zabezpečení a důvěryhodnosti QR kódu.

Oproti vodoznaku je odznak uložen ve formátu base64 ve skriptu `base64_signed_badge.py`, který obsahuje pouze konstantu obsahující data. Pro aplikaci odznaku je nejprve potřeba dekodovat base64 data využitím standardní Python knihovny `base64` a následně je převést na obrázek ve formátu RGBA pomocí knihovny `Pillow`. Další operace jsou podobné operacím při práci s vodoznakem. Tyto operace zahrnují změnu velikosti a průhlednosti a úpravu pozice. Tyhle operace zajišťuje funkce, která je deklarována ve skriptu `qr_generator.py`.

9.4.3 Algoritmus aplikace vodoznaku a odznaku

Vodoznak a odznak je aplikován ve fázi generování QR kódu. Nejprve je QR kód vytvořen a následně je případně aplikován vodoznak a odznak. Vodoznak je aplikován pouze tehdy, pokud uživatel nahraje obrázek do příslušného vstupního pole a odznak digitálního podpisu je aplikován pouze pokud je vygenerovaný QR kód digitálně podepsán. Aplikaci obou vizuálních prvků obstarává hlavní funkce, která vyhodnotí podle uživatelských vstupů, který prvek aplikovat. Rozhodovací logika je uvedena v následující ukázce kódu.

```
# Add watermark if watermark logo is uploaded
if base64_watermark_data:
    composed_image = add_watermark(qr_image, base64_watermark_data,
    opacity)
else:
    composed_image = Image.new("RGBA", qr_size, (255, 255, 255, 255))

# Add signed badge if key is uploaded
private_key_input = document.querySelector("#private-key-input")
key_type = private_key_input.getAttribute("data-key-type")
if key_type:
    composed_image = add_signed_badge(composed_image)
```

Vodoznak je aplikován jako částečně průhledný obrázek, kterému může uživatel po nahrání nastavit průhlednost mezi hodnotami 0,1 až 0,9 se skokem 0,1. Nejprve se upraví velikost obrázku, aby seděla k velikosti vygenerovaného QR kódu a nastaví se zvolená

průhlednost. Posléze se vezme vrstva bílého pozadí a na ni se vloží vrstva vodoznaku přímo doprostřed. Tento částečně sestavený QR kód je následně vrácen pro možnost aplikování odznaku. Tato technika zajišťuje, že vložený vodoznak je viditelný a nijak nenarušuje čitelnost vygenerovaného QR kódu. Jediný typ obrázku, který při použití jako vodoznaku naruší čitelnost vygenerovaného QR kódu je pokud obrázek obsahuje černou barvu a nastaví se nízká průhlednost. Tohle způsobí, že černá barva z vodoznaku bude splývat s černými částmi QR kódu. Aplikování vodoznaku jako druhou vrstvu ukazuje následující úryvek kódu.

```
# Resize watermark and set opacity
watermark = watermark.resize((new_width, new_height), Image.LANCZOS)
watermark_mask = watermark.getchannel("A").point(lambda p: int(p *
opacity))

# Step 1: White background
base = Image.new("RGBA", qr_size, (255, 255, 255, 255))

# Step 2: Paste watermark in the center
paste_x = (qr_size[0] - watermark.width) // 2
paste_y = (qr_size[1] - watermark.height) // 2
base.paste(watermark, (paste_x, paste_y), watermark_mask)
```

Pokud při generování QR kódu byl nahrán soukromý klíč, tedy QR kód je digitálně podepsaný, přidá se to pravého spodního rohu odznak, který vizuálně signalizuje, že vygenerovaný QR kód byl podepsaný. Tento přidaný odznak upozorňuje uživatele, který chce daný QR kód naskenovat, že daný obrázek obsahuje digitální podpis přímo v obrázku nebo je doprovázen textovým souborem s podpisem. Obrázek odznaku je uložen ve formátu base64 v samostatném Python skriptu. Nejprve se upraví velikost odznaku, aby odpovídala třetině velikosti vygenerovaného QR kódu a nastaví se průhlednost na 66%. Následně se nastaví pozice odznaku na spodní pravý roh a vloží se na částečně sestavený obrázek QR kódu na vrstvu s vodoznakem. Jednotlivé kroky úpravy a aplikování odznaku jsou ukázané v úryvku kódu níže.

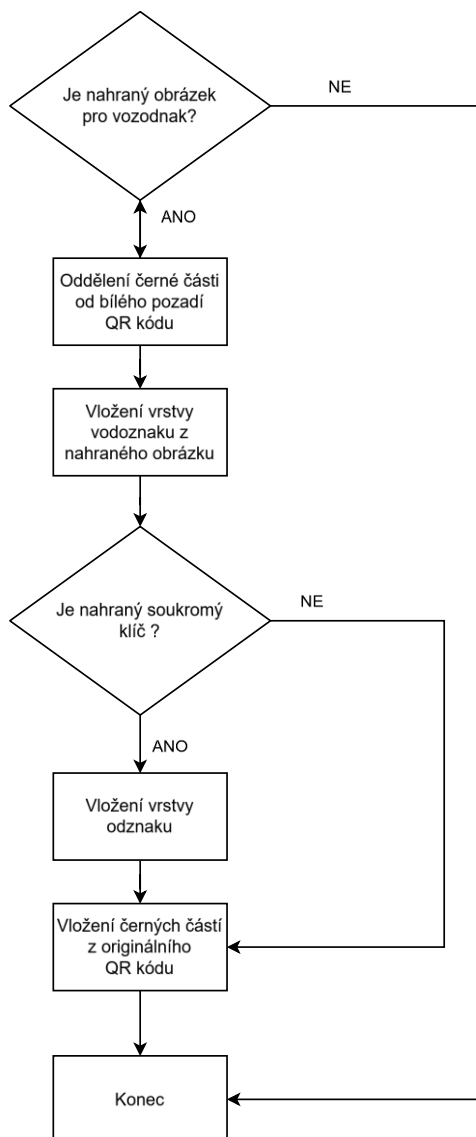
```
# Resize badge to 33% of size of generated QR code
qr_width, qr_height = image.size
badge_size = int(min(qr_width, qr_height) * 0.33)
signed_img = signed_img.resize((badge_size, badge_size), Image.LANCZOS)

# Set opacity to 66%
opacity = 0.66
signed_mask = signed_img.getchannel("A").point(lambda p: int(p *
opacity))

# Paste at bottom-right
position = (qr_width - badge_size - 5, qr_height - badge_size - 5)
image.paste(signed_img, position, signed_mask)
```

Po vložení vrstev vodoznaku a odznaku je na závěr vložena poslední vrstva černých částí z originálně vytvořeného QR kódu. Tímto zajistíme, že celý QR kód zůstane dobře čitelný a neporušený pro čtečky QR kódů. Vložení poslední vrstvy do finálního obrázku je popsána v následující ukázce z kódu.

```
# Add black QR pixels from original QR
qr_bw = qr_image.convert("L")
qr_mask = qr_bw.point(lambda p: 255 if p < 128 else 0).convert("1")
black_layer = Image.new("RGBA", qr_size, (0, 0, 0, 255))
composed_image.paste(black_layer, mask=qr_mask)
```



Obrázek 23 – Diagram aplikace vodoznaku a odznaku

(zdroj: vlastní, zhotoveno pomocí <https://www.drawio.com>)

9.5 Funkční tok aplikace

Základním vstupním bodem pro vygenerování, detekci nebo ověření QR kódu je zvolení příslušné sekce ve webovém rozhraní a následné vyplnění nebo nahrání požadovaných vstupních dat nebo souborů. Každá sekce obsluhuje jinou funkcionalitu, proto každá sekce obsahuje vstupní pole, která slouží pro obsluhu zvolené funkce aplikace.

9.5.1 Generování QR kódu

Při generování QR kódu je první krok výběr typu QR kódu z rozbalovacího seznamu. Po výběru požadovaného typu se zobrazí příslušná vstupní textová pole, které uživatel vyplní daty. Dále má uživatel možnost nahrát soukromý klíč, který slouží pro podepsání

vygenerovaného QR kódu a obrázek, který bude použitý jako vodoznak. Po vyplnění všech informací a nahrání souborů se po stisku tlačítka pro vygenerování QR kódu zavolá funkce pro obsluhu generování QR kódu ze skriptu `event_handlers.py`. Tato funkce získá data ze vstupních textových polí na základě zvoleného typu QR kódu tak, že se zavolá příslušná funkce ze skriptu `qr_content.py`, která vstupní informace naplní do strukturovaného obsahu a předá je zpět. Na základě statusu nahrání soukromého klíče se vygeneruje QR kód s digitálním podpisem nebo bez něj. Tyto funkce jsou obsažené ve skriptu `qr_generator.py`. Výběr postupu při generování QR kódu je ukázán na následujícím úryvku kódu.

```
# Check if private key is loaded
private_key_input = document.querySelector("#private-key-input")
if private_key_input.files.length > 0:
    # Key is loaded - sign the content
    sign_and_generate_qr(content)
else:
    # No key - generate without signature
    generate_qr_with_content(content, None)
```

V případě generování s digitálním podpisem je nejprve získán obsah soukromého klíče, který slouží k vytvoření digitálního podpisu prostřednictvím funkce, která je definována ve skriptu `crypto.py`. Tato funkce vrací obsah klíče a typ daného klíče. Následně se zavolá funkce pro vygenerování QR kódů, které se předá vytvořený digitální podpis a strukturovaný obsah. Tato funkce si na začátku, ještě před generováním QR kódu, získá zvolenou úroveň korekce chyb a následně vygeneruje QR kód. Posléze proběhne ověření, jestli byl nahrán obrázek pro vodoznak. Pokud ano, je zavolána funkce pro aplikaci vodoznaku a odznaku označujícího, že vygenerovaný QR kód byl digitálně podepsán. Proces vygenerování QR kódu lze vidět na ukázce z kódu níže.

```
# Generate basic QR code
qr = qrcode.QRCode(
    version=1,
    error_correction=error_correct,
    box_size=20,
    border=4,
)
qr.add_data(content)
qr.make(fit=True)
qr_image = qr.make_image(fill_color="black",
    back_color="white").convert('RGB')
```

Aplikace vodoznaku probíhá rozdělením částí QR kódu na vrstvy a následným skládáním vrstev na sebe. Prvně se oddělí bílé pozadí a černé části QR kódu a každá část se uloží do samostatné vrstvy. Mezi tyto vrstvy se vloží vrstvy vodoznaku a odznaku. Všechny vrstvy se spojí do obrázku a vytvoří se výsledný QR kód, který už obsahuje aplikovaný vodoznak i odznak, signalizující, že QR kód je digitálně podepsán.

Pokud je zaškrtnuté pole pro použití steganografie, vloží se digitální podpis přímo do obrázku pomocí LSB steganografie. Pro vložení podpisu se zavolá funkce ze skriptu stego.py, která potřebuje obrázek vygenerovaného QR kódu a data podpisu. Na závěr je vygenerován textový soubor obsahující data digitálního podpisu, který je vystavený na webové aplikaci s možností stáhnutí jako samostatného souboru. Proces vytvoření a vystavení textového souboru s digitálním podpisem je popsán v následujícím úryvku kódu.

```
# Convert to PNG format
output = io.BytesIO()
final_image.save(output, format="PNG")

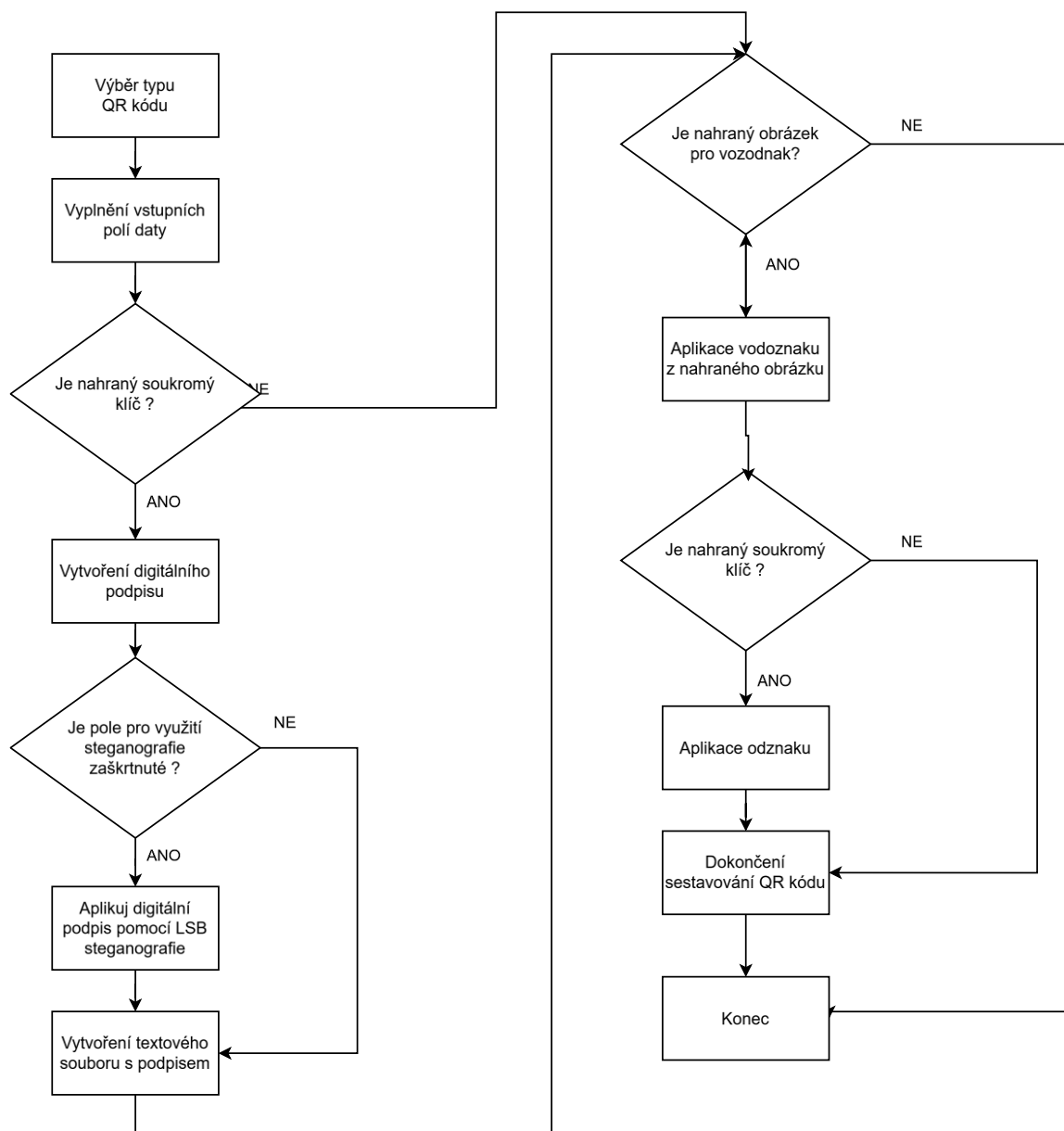
# Create File object and display
bytes_data = output.getvalue()
blob = Uint8Array.new(bytes_data)
image_file = File.new([blob], "qr_code.png", {"type": "image/png"})

# Update the image source
img = document.querySelector("#qr_image")
img.src = URL.createObjectURL(image_file)
```



Obrázek 24 – Digitálně podepsaný QR kód s vodoznakem

(zdroj: vlastní)



Obrázek 25 – Diagram generování QR kódu

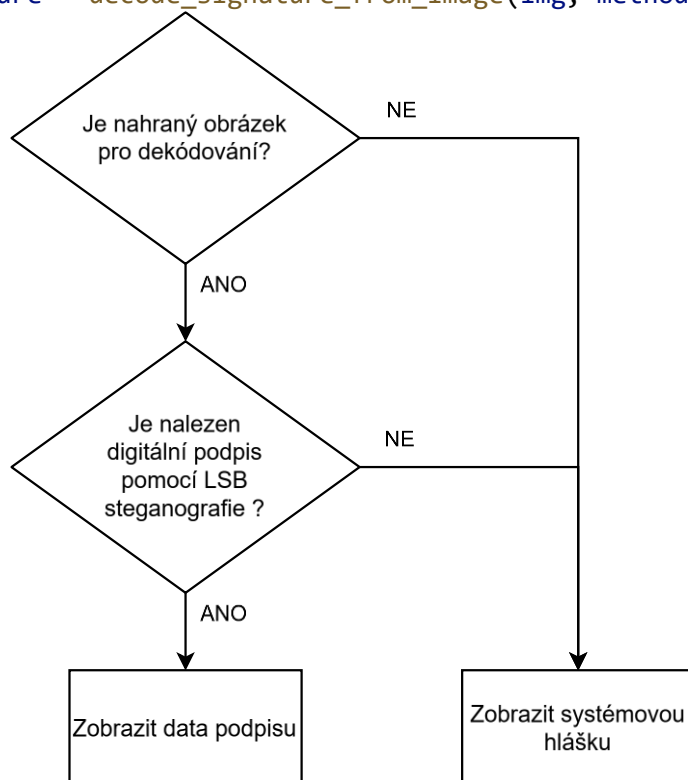
(zdroj: vlastní, zhotoveno pomocí <https://www.drawio.com>)

9.5.2 Dekódování QR kódu

V sekci pro dekódování QR kódu je potřeba nahrát pouze obrázek s QR kódem. Po nahrání obrázku a stisknutí tlačítka pro dekódování se zavolá funkce ze skriptu `event_handlers.py`, která obstarává kontrolu vstupu a obsluhu dekódování. Nejprve proběhne kontrola, jestli byl QR kód nahráný. Pokud ano, zavolá se funkce pro dekódování podpisu ze skriptu `stego.py`, které se jako vstup předá nahráný obrázek. Funkce se posléze pokusí najít skrytý podpis a pokud jej najde, zobrazí jej pod tlačítkem. Pokud funkce nic nenajde, zobrazí se hláška, že digitální podpis nebyl nalezen. Tento proces ukazuje následující ukázka kódu.

```
try:
    response = await window.fetch(img_element.src)
    array_buffer = await response.arrayBuffer()
    byte_array = Uint8Array.new(array_buffer)
    img = Image.open(io.BytesIO(byte_array.to_py()))

    # Decode the signature
    signature = decode_signature_from_image(img, method='lsb')
```



Obrázek 26 – Diagram dekódování QR kódu

(zdroj: vlastní, zhotoveno pomocí <https://www.drawio.com>)

9.5.3 Ověření QR kódu

Pro ověření QR kódu je potřeba mít veřejný klíč osoby, která QR kód vygenerovala a jednu z následujících variant. První variantou je obrázek QR kódu s aplikovaným podpisem pomocí LSB steganografie. Druhou variantou je obrázek QR kódu bez aplikovaného podpisu a textový soubor s digitálním podpisem. Třetí možnou variantou je obrázek QR kódu s aplikovaným podpisem pomocí LSB steganografie a textový soubor s podpisem. V případě třetí varianty aplikace ověří oba dva podpisy. Následující úryvek kódu popisuje pokus o ověření QR kódu pomocí podpisu získaného technikou LSB steganografie.

```
# Try to verify with steganography signature if available
if img_element.src:
    steg_signature = decode_signature_from_image(img, method='lsb')
    if steg_signature:
        steg_verified = verify_signature(public_key, steg_signature,
                                         content_bytes, key_type)
        if not steg_verified:
            result_messages.append("Steganography signature
                                  verification failed")
```

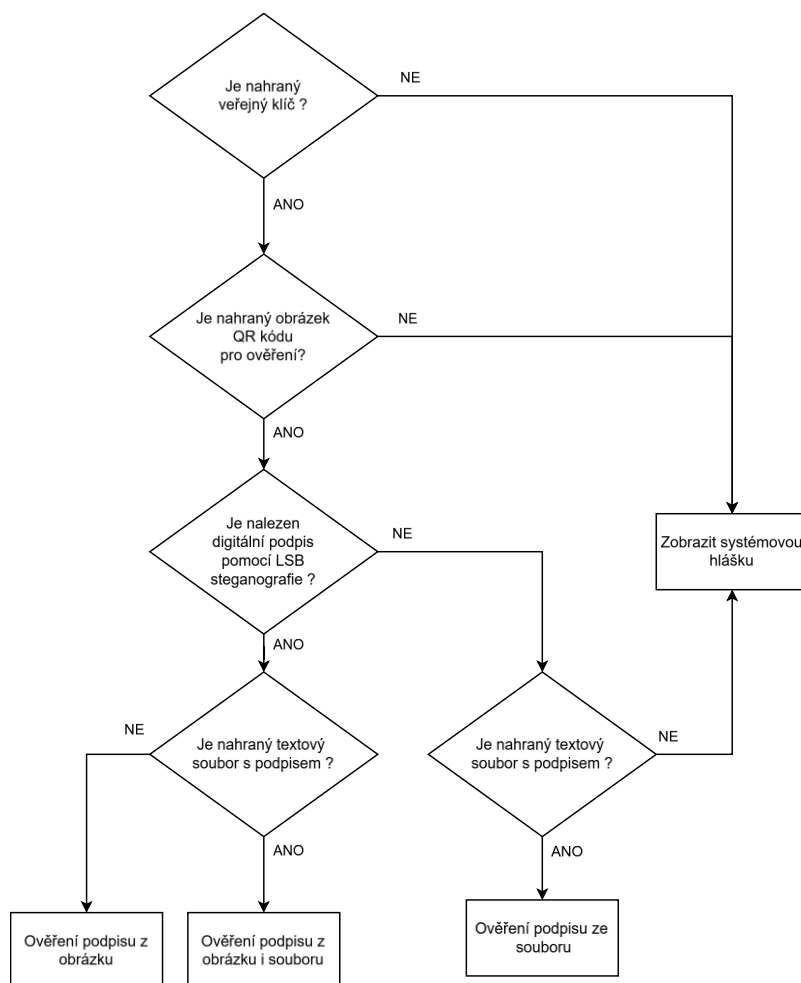
Po nahrání obrázku QR kódu, veřejného klíče a případně textového souboru s podpisem se po stisknutí tlačítka pro ověření zavolá obslužná funkce definována ve skriptu `event_handlers.py`. Tato funkce nejprve ověří, že jsou nahrané soubory v uživatelských vstupech a poté zpracuje nahraný veřejný klíč pomocí funkce ze skriptu `crypto.py`, která vrátí obsah klíče a typ klíče. Po získání obsahu veřejného klíče se přečtou data obsažené v nahraném QR kódu. Následně se zavolá funkce ze skriptu `stego.py`, která se pokusí dekodovat aplikovaný digitální podpis pomocí LSB steganografie z nahraného obrázku QR kódu. Pokud nahraný obrázek obsahuje digitální podpis, zavolá se funkce, která na základě dekodovaného podpisu, obsahu nahraného QR kódu a veřejného klíče ověří pravost obsahu nahraného QR kódu. Tato funkce je deklarována ve skriptu `crypto.py`. V případě, že je nahraný i textový soubor s podpisem, ověří se obsah QR kódu i oproti tomuto podpisu. Na závěr se zavolá funkce ze skriptu `ui_handlers.py`, která nastaví hlášku určující výsledek ověření. Výsledná hláška popisuje, který zdroj digitálního podpisu byl použitý pro ověření QR kódu, jestli ze steganografie, textového souboru nebo obou dvou. Následující úryvek obsahuje logiku ověření obsahu QR kódu.

```

if key_type == 'ed25519':
    public_key.verify(signature, content_bytes)
    return True
elif key_type == 'rsa':
    public_key.verify(
        signature,
        content_bytes,
        padding.PSS(
            mgf=padding.MGF1(hashes.SHA256()),
            salt_length=padding.PSS.MAX_LENGTH
        ),
        hashes.SHA256()
    )

```

Pokud nahraný obrázek QR kódu neobsahuje digitální podpis a textový soubor s podpisem nebyl nahrán, zobrazí se hláška s informací, že nebyl nalezen žádný digitální podpis k ověření.



Obrázek 27 – Diagram ověření QR kódu

(zdroj: vlastní, zhotoveno pomocí <https://www.drawio.com>)

9.5.4 Detekování QR kódu

Sekce pro detekování QR kódu z fotografie obsahuje pouze dvě karty, kde každá obsahuje dvě tlačítka, kterým se dynamicky mění odkazy vždy po načtení webové aplikace. Tlačítka slouží pro stažení CLI nebo GUI verze aplikace a odkazují vždy na nejnovější verzi aplikace. Změnu odkazů obstarává skript `downloadLinks.js`, který při každém načtení stránky zjistí tag nejnovější verze aplikace QRebuild pomocí GitHub API. Ze získaného tagu, typu operačního systému a typu aplikace sestaví finální verzi odkazu, který nastaví na tlačítka. Výsledkem jsou tlačítka, které při stisknutí automaticky začnou stahovat nejnovější verzi aplikace pro zvolený operační systém a typ.

9.6 Automatické nasazení webové aplikace

Pro plně automatizované nasazení webové aplikace, která je hostovaná na platformě GitHub Pages, slouží automatizované workflow, které obstarává publikaci webové aplikace při každé změně v hlavní větvi `main`. Workflow je sepsaný pomocí jazyku YAML a je uložený ve složce `.github/workflows`. Tento workflow se spouští při každém pushi do větve `main`, ale nabízí i možnost ručního spuštění z prostředí GitHub Actions.

Vytvořené workflow obsahuje pouze jednu úlohu s názvem `deploy`, která používá již předpřipravené akce. Workflow nejprve stáhne obsah repozitáře a následně připraví prostředí GitHub Pages spuštěním akce. Následně proběhne nahrání artefaktů a publikace webové aplikace.

```
steps:
  - name: Checkout
    uses: actions/checkout@v4
  - name: Setup Pages
    uses: actions/configure-pages@v5
  - name: Upload artifact
    uses: actions/upload-pages-artifact@v3
    with:
      # Upload entire repository
      path: '.'
  - name: Deploy to GitHub Pages
    id: deployment
    uses: actions/deploy-pages@v4
```

10 DESKTOPOVÁ APLIKACE QREBUILD

Desktopová aplikace s názvem QRebuild byla vytvořena jako doplňkový nástroj k webové aplikaci, která slouží pro znovuvytvoření QR kódu z fotografií do digitální podoby s totožným obsahem. Aplikace je dostupná pro operační systémy Windows a Linux. Pro vytvoření této aplikace byl zvolen programovací jazyk Python a pro každý operační systém byl vytvořen ve formátu CLI a GUI. Verze CLI slouží pro obsluhu pomocí příkazové řádky a GUI obsahuje jednoduché a přehledné uživatelské rozhraní vytvořené pomocí frameworku PyQt6. Oba formáty jsou distribuované jako samostatné spustitelné soubory. Sestavené verze aplikace jsou dostupné ke stažení ve webové aplikaci, nebo přímo ze stránek GitHubu zde: <https://github.com/Pipousek/QRebuild/releases>.



Obrázek 28 – Odkaz na seznam vydaných verzí aplikace QRebuild

(zdroj: vlastní)

10.1 Důvod vzniku a účel aplikace

Desktopová aplikace QRebuild vznikla jako přímá reakce na technickou neschopnost webové aplikace, která kvůli svému provozu na platformě GitHub Pages nedisponuje serverem, tedy ani dostatečným výpočetním výkonem pro detekci. Webová aplikace nedokáže detekovat a číst QR kódy z fotografií nebo naskenovaných dokumentů či listů za účelem ověření nahraného QR kódu. Proto je zapotřebí, aby obrázek QR kódu použitý v aplikaci byl v digitální podobě s dostatečně vysokým rozlišením.

Hlavním účelem aplikace je snadná, rychlá a jednoduchá rekonstrukce QR kódu z fotografie či skenu do digitální podoby s vysokým rozlišením, aby mohl být

rekonstruovaný QR kód použitý ve webové aplikaci pro ověření pravosti. Aplikace detekuje QR kód z přiložené fotografie a vygeneruje nový QR kód se stejným obsahem ve standardizované podobě. QRebuild představuje rozšíření funkcionality webové aplikace o práci s méně kvalitními vstupními daty, která kvůli platformě GitHub Pages není možná.

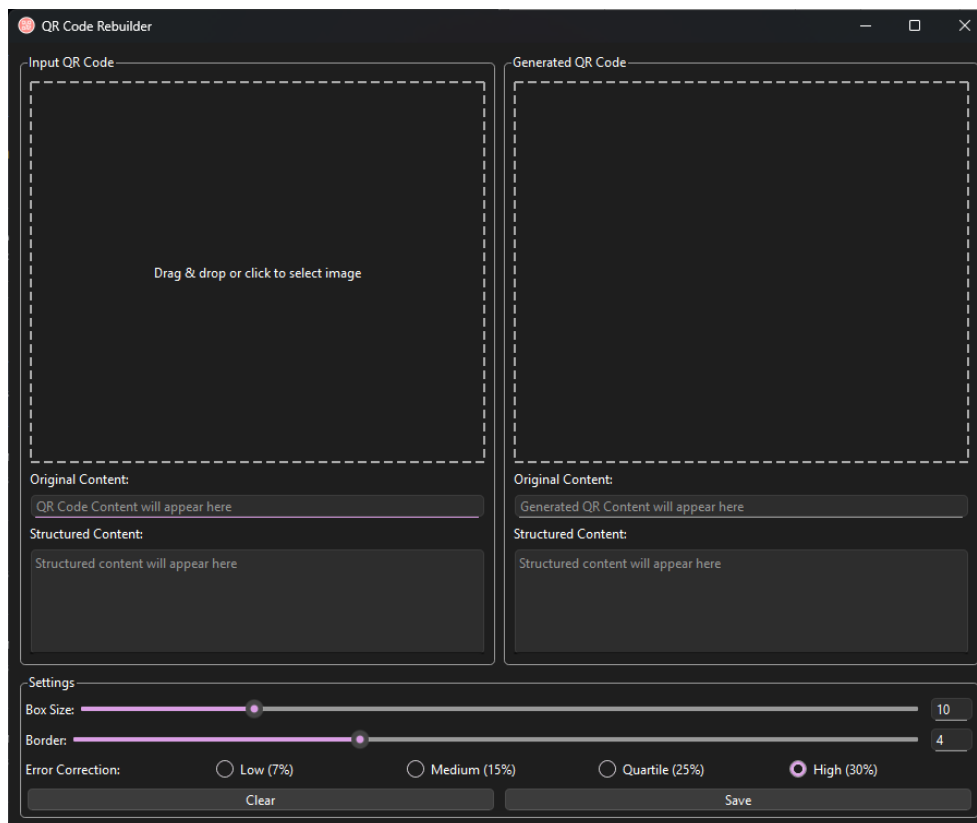
Aplikace je rozdělena do dvou variant. První variantou je CLI verze, která je určena pro pokročilejší uživatele, kteří spíše pracují s příkazovou řádkou. CLI verze umožňuje využití aplikace v automatických nástrojích nebo dávkových skriptech. Druhou variantou je GUI verze, která obsahuje jednoduché a přehledné uživatelské rozhraní, které je vhodné pro běžné uživatele.

10.1.1 Uživatelské rozhraní

Uživatelské rozhraní aplikace je rozděleno na tři základní části. První částí je vstupní pole pro nahrání fotografie obsahující QR kód. Po nahrání se zobrazí nahraná fotografie a pod ní obsah QR kódu ve strukturované a originální podobě. Originální data slouží k rekonstrukci QR kódu do digitální podoby s totožnými daty. Strukturovaná data jsou vhodná pro zjištění typu a snadnou orientaci v datech nahraného QR kódu.

Druhá část zobrazuje rekonstruovaný QR kód a jeho obsah v obou podobách, jak čisté data, tak i ve data ve strukturované podobě. Zobrazená data slouží ke kontrole, jestli rekonstruovaný QR kód obsahuje totožná data jako QR kód originální. Náhled rekonstruovaného QR kódu je zobrazen ihned po nahrání fotografie a je automaticky aktualizován při provádění jakékoliv změny.

Poslední část obsahuje ovládací prvky, které upravují vzhled rekonstruovaného QR kódu v oblasti velikosti černé buňky a bílého rámečku. Poslední nastavitelný prvek slouží pro volbu úrovně korekce chyb. Všechny provedené změny se okamžitě aplikují do rekonstruovaného QR kódu, který je zobrazen. Pod ovládacími prvky se nachází dvě tlačítka, jedno pro vymazání nahrané fotografie a rekonstruovaného QR kódu a druhé pro uložení rekonstruovaného QR kódu jako obrázek. Po stisku tlačítka pro uložení se zobrazí průzkumník souborů pro zvolení destinace pro uložení.



Obrázek 29 – Uživatelské rozhraní aplikace QRebuild

(zdroj: vlastní)

10.2 Architektura a implementace

Architektura aplikace je postavena na základech snadné údržby a jednoduchém rozšíření aplikace o další funkcionality a moduly. Pro implementaci byl zvolen programovací jazyk Python, kvůli velkému množství knihoven pro práci s obrázky a QR kódy. Základem pro obě verze aplikace je společná knihovna, která obstarává veškerou práci s obrázky a QR kódy. Tato knihovna je následně využívána v obou verzích aplikace. GUI verze aplikace využívá navíc framework PyQt6 pro tvorbu uživatelského rozhraní a verze CLI obsahuje navíc knihovnu argparse pro zpracování vstupních parametrů. Obě verze aplikace jsou sestavené využitím Python modulu Pyinstaller.

10.2.1 Modulární struktura

Struktura aplikace QRebuild je rozdělená do tří částí. Společná část s názvem core obsahuje skripty, které zajišťují práci s obrázky, čtení dat a detekci typu QR kódů, rekonstrukci QR kódu do digitální podoby a zobrazení vygenerovaného QR kódu. Seznam a účel skriptů je následující:

- `core/content-analyzer.py`
 - Skript sloužící pro analýzu obsahu nahraného a detekovaného QR kódu. Skript analyzuje obsah a vrátí typ QR kódu a strukturovaná data, aby se uživateli lépe kontrolovali.
- `core/qr_extractor.py`
 - Nejdůležitější část aplikace QRebuild, která slouží pro detekování QR kódu z nahrané fotografie. Skript pro detekci využívá knihovny `pyzbar` a `opencv-python`.
- `core/qr_generator.py`
 - Skript využívající knihovnu `qrcode` pro vygenerování QR kódu s totožným obsahem, jako detekovaný kód z nahrané fotografie.
- `core/utils.py`
 - Tento skript slouží pouze pro zobrazení vygenerovaného obrázku QR kódu. Pro zobrazení obrázku je využita knihovna `Pillow`.

CLI verze obsahuje pouze dva skripty navíc. První obsluhuje práci se vstupními argumenty, které aplikace umí zpracovat. Využitím knihovny `argparse` se vyhneme dlouhému porovnávání a kontrolování předaných argumentů ručně. Pro použití `argparseru` je potřeba nastavit jeden nebo více názvů parametru, datový typ, který je povolený pro daný parametr, výchozí hodnotu a pomocnou zprávu. Všechny tyto nastavení jsou viditelné při zavolání parametru pro nápovědu. Druhý skript je hlavní vstupní skript, který pracuje s ostatními skripty a zajišťuje správný průběh aplikace.

- `cli/argparser.py`
 - Skript obsluhující vstupní parametry a pomocí knihovny `argparse`.
- `cli_main.py`
 - Hlavní vstupní skript aplikace pro verzi CLI. Skript importuje všechny potřebné moduly ze společné části `core` a zajišťuje správné zpracování vstupních parametrů pro rekonstrukci QR kódu.

Tabulka 8 – Parametry CLI verze aplikace QRebuild

Parametr	Popis	Výchozí hodnota
-i INPUT --input INPUT	Cesta ke vstupnímu obrázku obsahujícímu QR kód	qr_code_photo.jpg
-o OUTPUT --output OUTPUT	Cesta pro uložení očištěného QR kódu	clean_qr_output.png
-b BOX_SIZE --box_size BOX_SIZE	Velikost jednotlivých modulů (černých částí) QR kódu v pixelech	10
--border BORDER	Počet modulů tvořících okraj QR kódu	4
-e {L,M,Q,H} --error_correction {L,M,Q,H}	Úroveň opravy chyb (L, M, Q, H)	H
-d --display	Zobrazit vygenerovaný QR kód	Vypnuto
-h --help	Zobrazí nápovědu a ukončí program	---

(zdroj: vlastní)

Třetí sekci je GUI verze aplikace tvořena navíc dvěma hlavními skripty. První z nich obsahuje implementaci uživatelského rozhraní pomocí frameworku PyQt6. Uživatel může nahrát fotografii s QR kódem přetažením do příslušného vstupního pole, nebo pomocí kliknutí otevřít prohlížeč souborů a vybrat soubor z něj. Aplikace obsahuje grafické prvky pro úpravu rekonstruovaného QR kódu. Druhý skript slouží jako hlavní vstupní skript celé aplikace, který inicializuje Qt aplikaci, nastaví ikonu aplikace a zobrazí hlavní okno aplikace.

- gui_app.py
 - Skript obsahující grafické rozhraní aplikace pomocí frameworku PyQt6. Skript importuje požadované moduly ze společné části core, které jsou volané při změně nastavení, vstupních dat nebo kliknutí na tlačítko.

- `gui_main.py`
 - Hlavní vstupní skript aplikace pro verzi GUI, která nejprve inicializuje Qt aplikaci, nastaví ikonu a zobrazí okno aplikace. Zbytek operací obstarává skript `gui_app.py`.

10.3 Funkcionalita aplikace

Hlavní funkcí aplikace QRebuild je rekonstrukce QR kódu z nahrané fotografie či dokumentu do digitální podoby. Celý proces zahrnuje detekci, dekodování, generování a možnost uložení QR kódu. Aplikace je dostupná v provedení s grafickým rozhraním a ve v provedení pro příkazovou řádku. Obě verze obsluhují hlavní funkcionalitu aplikace. Verze s grafickým rozhraním je možné použít i pouze pro získání obsahu QR kódu, který je vypsán po nahrání QR kódu.

10.3.1 Nahrání a detekce QR kódu z fotografie

Prvním krokem v aplikaci QRebuild pro rekonstrukci QR kódu do digitální podoby je nahrání fotografie nebo dokumentu, který obsahuje QR kód. Pro detekci QR kódu z nahrané fotografie jsou využity knihovny `pyzbar` a `opencv-python`. Ve verzi s grafickým uživatelským rozhraním lze soubor nahrát pomocí přetažení do příslušného vstupního pole ve formátu Drag & Drop nebo po kliknutí do pole se otevře průzkumník souborů. Pokud nahraná fotografie neobsahuje QR kód, aplikace o tom upozorní v poli pro výpis obsahu nahraného QR kódu. Verzi pro příkazový řádek předáváme cestu k fotografii a pokud v předané fotografii není nalezený QR kód, aplikace vypíše chybovou hlášku.

```
def extract_qr(image_path):  
    img = cv2.imread(image_path)  
    if img is None:  
        raise ValueError("Image not found")  
    decoded = decode(cv2.cvtColor(img, cv2.COLOR_BGR2GRAY))  
    if not decoded:  
        raise ValueError("No QR code found")  
    return decoded[0].data.decode("utf-8")
```

10.3.2 Rekonstrukce QR kódu do digitální podoby

Ve verzi s GUI se po úspěšném detekování QR kódu z nahrané fotografie spustí dekodování obsahu QR kódu na kterém se spustí analýza. Analýza slouží pro strukturovaný výpis dekodovaných dat, které jsou lépe čitelné a přehledné pro uživatele. Níže je uvedena ukázka analýzy a strukturování dekodovaných dat.

```

# 2. Check for URL
parsed = urlparse(content)
if parsed.scheme and parsed.netloc:
    return {
        'type': 'URL',
        'data': {
            'scheme': parsed.scheme,
            'domain': parsed.netloc,
            'path': parsed.path or '/',
            'query': parsed.query,
            'fragment': parsed.fragment
        }
    }

```

Následně se vygeneruje nový QR kód s totožným detekovaným obsahem za pomoci knihovny qrcode. Originální i strukturovaný obsah a vygenerovaný QR kód se zobrazí v uživatelském rozhraní aplikace. Úryvek kódu ukazuje průběh vygenerování QR kódu.

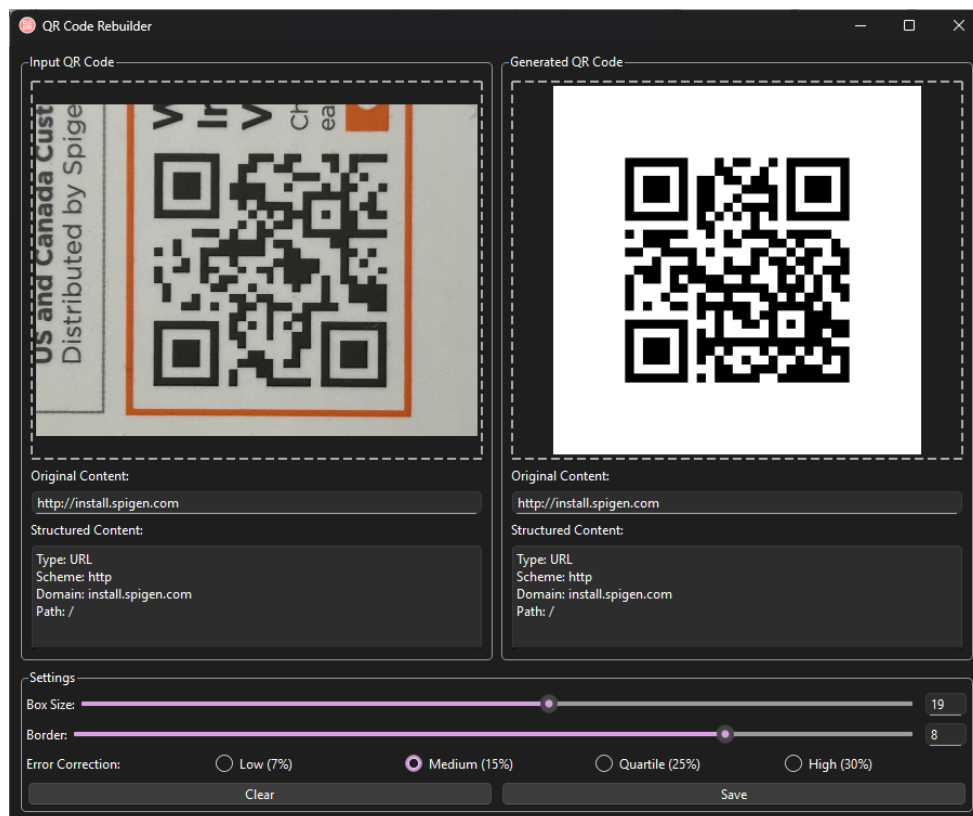
```

def generate_qr(data, box_size=10, border=4, error_correction='H'):
    error_levels = {
        'L': qrcode.constants.ERROR_CORRECT_L,
        'M': qrcode.constants.ERROR_CORRECT_M,
        'Q': qrcode.constants.ERROR_CORRECT_Q,
        'H': qrcode.constants.ERROR_CORRECT_H
    }
    qr = qrcode.QRCode(
        version=1,
        error_correction=error_levels.get(error_correction,
            qrcode.constants.ERROR_CORRECT_H),
        box_size=box_size,
        border=border
    )
    qr.add_data(data)
    qr.make(fit=True)
    return qr.make_image(fill_color="black", back_color="white")

```

10.3.3 Zobrazení výsledků a export

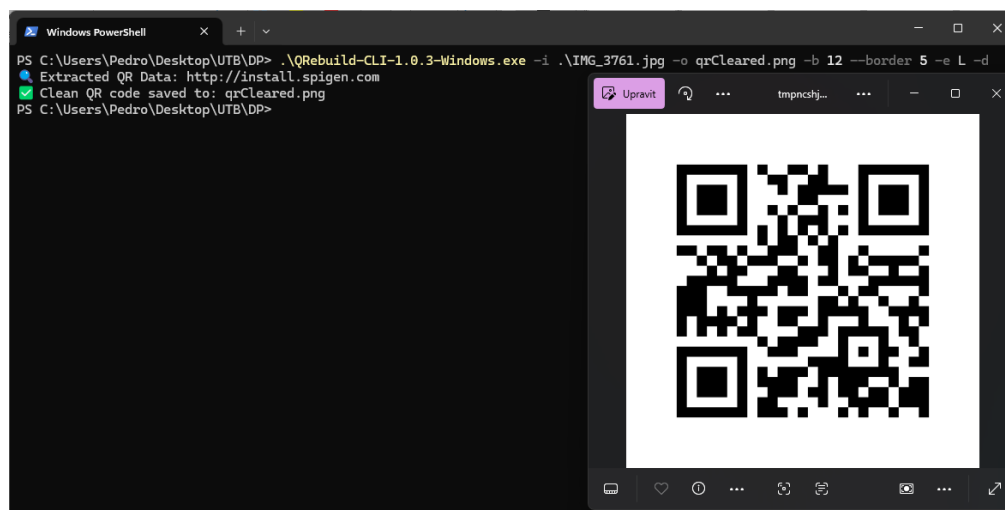
V GUI verze aplikace je rekonstruovaný QR kód zobrazený v příslušném poli a je automaticky aktualizován ihned po provedení změny. Výsledný QR kód může být exportován pomocí tlačítka Save, které otevře průzkumník souborů, kde si uživatel zvolí místo pro uložení obrázku QR kódu.



Obrázek 30 – Zobrazení rekonstruovaného QR kódu v GUI verzi aplikace

(zdroj: vlastní)

CLI verze dokáže zobrazit rekonstruovaný QR kód předáním parametru `-d` nebo `--display`, který otevře vygenerovaný QR kód pomocí výchozího prohlížeče obrázků. Pro tuto funkcionalitu je využita knihovna Pillow. Vygenerovaný QR kód je uložen do cesty uvedené v parametru `-o` nebo `--output`.

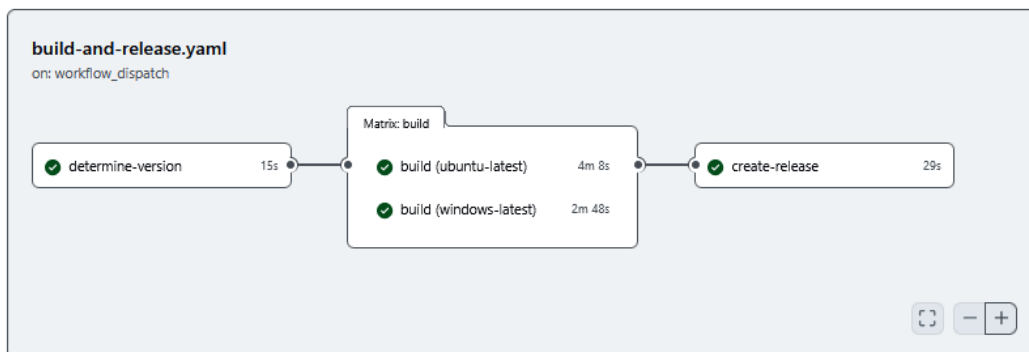


Obrázek 31 – Zobrazení rekonstruovaného QR kódu v CLI verzi aplikace

(zdroj: vlastní)

10.4 Automatické sestavení a distribuce

Pro jednoduchou a automatickou distribuci nových verzí aplikace QRebuild byl vytvořen automatický proces, který obsluhuje všechny potřebné části, pro sestavení aplikace pro operační systémy Windows a Linux. Celý proces probíhá v prostředí GitHub Actions, kde se workflow spustí automaticky při každé změně kódu v repozitáři nebo je možnost spustit workflow ručně, kde se nutně předat požadovanou verzi sestavené aplikace.



Obrázek 32 – Průběh automatického sestavení a distribuce

(zdroj: vlastní)

10.4.1 Sestavení aplikace pomocí GitHub Actions

Automatizovaný sestavovací proces aplikace QRebuild je definován pomocí GitHub Actions v konfiguračním souboru YAML, který je uložen přímo v repozitáři ve složce `.github/workflows`. Tento workflow se automaticky spustí při každém pushi do větve `main`, pokud dojde ke změně v některém ze souborů potřebných pro vytvoření aplikace. Lze jej také spustit ručně prostřednictvím `workflow_dispatch`, přičemž je nutné zadat konkrétní verzi distribuce. Celý workflow je rozdělen do tří hlavních úloh, které se nazývají jobs. Část pro sestavení aplikace využívá první dvě úlohy.

```

on:
  push:
    branches:
      - 'main'
    paths:
      - 'cli/**'
      - 'core/**'
      - '**/*.py'
      - 'requirements.txt'
  workflow_dispatch:
    inputs:
      version:
        description: 'Version number (e.g., 1.0.0)'
        required: true
        default: '0.0.0'

```

Prvním úloha se nazývá `determine-version` a jejím cílem je zjistit číslo verze pro sestavení. Pokud byl workflow spuštěn ručně, použije se verze, kterou zadal uživatel při spuštění workflow. Pokud se workflow spustil automaticky, zjistí se nejnovější existující Git Tag ve formátu `X.Y.Z` a automaticky se poslední číslo verze navýší o jedničku. Tato získaná verze se použije v dalších krocích.

```

- name: Get latest tag and calculate version
  id: set-version
  run: |
    if [ "${{ github.event_name }}" = "workflow_dispatch" ]; then
      echo "Using provided version: ${ inputs.version }"
      echo "version=${{ inputs.version }}" >> $GITHUB_OUTPUT
    else
      # Get the latest semver tag
      latest_tag=$(git tag --sort=-v:refname | grep -E '^[0-9]+\.[0-9]+\.[0-9]+' | head -n 1)
      echo "Latest tag: $latest_tag"
      IFS='.' read -r major minor patch <<< "$latest_tag"
      new_patch=$((patch + 1))
      new_version="${major}.${minor}.${new_patch}"
      echo "Auto-incremented version: $new_version"
      echo "version=$new_version" >> $GITHUB_OUTPUT
    fi

```

Druhá úloha se nazývá `build` a slouží pro sestavení aplikace pro obě platformy (Windows a Linux), kde celý proces běží paralelně pomocí možnosti `matrix`. První kroky slouží pro nastavení Python prostředí na virtuálním stroji a nainstalování závislostí uvedených v souboru `requirements.txt`. Ještě před samotným sestavením je potřeba na stroji

s Linuxem doinstalovat systémové balíčky, potřebné pro úspěšné sestavení a funkci aplikace. Následují kroky samotného sestavení aplikace pro obě verze (CLI i GUI). Sestavení provádí modul Pyinstaller, kterému jsou předány zdrojové kódy a binární soubory, potřebné pro správnou funkci aplikace. Výsledkem je samostatný spustitelný soubor pro každou verzi a platformu. Soubory jsou poté přejmenovány podle získané verze v první úloze a přesunuty do složky release. Na závěr jsou sestavené aplikace nahrané na GitHub jako artefakty k danému workflow.

```
# Sample of app build process
- name: Build application - GUI (Windows)
  if: runner.os == 'Windows'
  run: |
    .venv/Scripts/activate
    pyinstaller .\gui_main.py `
      --onefile `
      --windowed `
      --icon="img/logo.ico" `
      --add-data="img:img" `
      --add-data="core:core" `
      --add-binary=".venv\Lib\site-packages\pyzbar\libiconv.dll:." `
      --add-binary=".venv\Lib\site-packages\pyzbar\libzbar-64.dll:." `
      --name QRebuild-GUI
```

10.4.2 Distribuce prostřednictvím GitHub Releases

Třetí úlohou ve workflow je vydání sestavených aplikací prostřednictvím platformy GitHub Releases. Úloha v prvním kroku stáhne nahrané artefakty vytvořené v předchozím kroku a spustí akci softprops/action-gh-release, která vytvoří nový release. Vydaný release obsahuje všechny varianty spustitelných souborů. Funkce zároveň vytvoří i tag v commitu, který spustil workflow. Po vytvoření release se ještě zaktualizuje tag latest tak, že starý latest tag je smazán a posléze je vytvořen nový na aktuální commit.

```
- name: Create Release
  uses: softprops/action-gh-release@v2
  with:
    tag_name: ${{ needs.determine-version.outputs.version }}
    name: QRebuild ${{ needs.determine-version.outputs.version }}
    files: |
      artifacts/binaries-*/*
    draft: false
    prerelease: false
```

```

- name: Update Latest Tag
  env:
    GITHUB_TOKEN: ${ secrets.GITHUB_TOKEN }
  run: |
    # Delete existing latest tag
    curl -s -X DELETE \
      -H "Authorization: token $GITHUB_TOKEN" \
      "https://api.github.com/repos/
        ${ github.repository }}/git/refs/tags/latest" || true

    # Create new latest tag
    curl -s -X POST \
      -H "Authorization: token $GITHUB_TOKEN" \
      -H "Content-Type: application/json" \
      -d '{"ref": "refs/tags/latest", "sha": "${ github.sha }}"' \
      "https://api.github.com/repos/${ github.repository }}/git/refs"

```

QRebuild 1.0.3 Latest

ci: linux change chown to chmod

▼ Assets 6

QRebuild-CLI-1.0.3-Linux	104 MB	1 hour ago
QRebuild-CLI-1.0.3-Windows.exe	62.1 MB	1 hour ago
QRebuild-GUI-1.0.3-Linux	269 MB	1 hour ago
QRebuild-GUI-1.0.3-Windows.exe	88.1 MB	1 hour ago
Source code (zip)		1 hour ago
Source code (tar.gz)		1 hour ago

Obrázek 33 – Distribuce sestavených QRebuild aplikací

(zdroj: vlastní)

ZÁVĚR

Tato diplomová práce se zabývala návrhem a implementací řešení pro zvýšení zabezpečení, důvěryhodnosti a autenticity QR kódu prostřednictvím digitálního podpisu a vodoznaku. Hlavním cílem byla analýza možností zabezpečení QR kódů, na základě kterých budou vytvořené nástroje, které umožní generování, ověřování a detekci QR kódů s přidávanými bezpečnostními prvky.

Přínosem této práce je primárně prakticky využitelná aplikace, která je jednoduše rozšiřitelná o další funkcionality, ale i metodika a doporučení pro běžné využití QR kódů v prostředích, kde je důležitá kontrola a ověření původu a obsahu dat. Práce ukazuje, že použitím moderních technologií lze rozšířit běžně dostupné nástroje o bezpečnostní prvky, které přispívají k ochraně proti podvrhům a neoprávněným zásahům do QR kódů.

Teoretická část práce se zaměřila na technologii QR kódu, jejich strukturu, typy a bezpečnostní rizika. Dále byly popsány metody ochrany proti zneužití a různé techniky vkládání vodoznaků a digitálního podpisu. Pozornost byla věnována také kryptografickým algoritmům RSA, ECDSA, hashovacímu algoritmu SHA-256 a metodám jejich využití v prostředí QR kódů.

Praktická část práce se zaměřila na návrh a realizaci webové aplikace postavené na technologiích HTML+CSS+JS, PyScript a GitHub Pages, a zároveň desktopové aplikace QRebuild, vyvinuté ve dvou variantách (GUI a CLI) pro operační systémy Windows a Linux. Zatímco webová aplikace slouží pro generování, podepisování a ověřování QR kódů, desktopové aplikace slouží pro rekonstrukci QR kódu z fotografie. Webová aplikace byla podpořena automatizovaným nasazením prostřednictvím GitHub Actions a GitHub Pages, zatímco desktopové aplikace využívají automatizovaný Build & Release proces prostřednictvím GitHub Actions a GitHub Releases.

V rámci testování byla porovnána tři řešení aplikace digitálního podpisu. Mezi tyto řešení se řadí přímé vložení do QR kódu, uložení pomocí LSB steganografie a distribuce podpisu jako samostatného souboru. Z analýzy vyplynulo, že právě oddělený digitální podpis v kombinaci s vodoznakem a odznakem poskytuje nejlepší rovnováhu mezi bezpečností, kompatibilitou a uživatelskou přívětivostí.

SEZNAM POUŽITÉ LITERATURY

- [1] *What is QR Code?* Online. In: DENSO WAVE. QRcode.com. Dostupné z: <https://www.qrcode.com/en/about/>. [cit. 2025-04-03].
- [2] GLASPIE, Armanii. *How QR Codes Work (+ a Quick Lesson in QR Code History)*. Online. In: QR Code Generator. 11 Nov 2024. Dostupné z: <https://www.qr-code-generator.com/blog/how-qr-codes-work/>. [cit. 2025-04-03].
- [3] *History of QR Code*. Online. In: DENSO WAVE. QRcode.com. Dostupné z: <https://www.qrcode.com/en/history/>. [cit. 2025-04-03].
- [4] P., Nove. *COVID-19 QR Code Usage Worldwide: Detailed Statistics Report*. Online. QR Tiger. © 2025, 07 February 2025. Dostupné z: <https://www.qrcode-tiger.com/qr-code-statistics-before-and-after-covid-19>. [cit. 2025-04-03].
- [5] P., Ashley. *When Were QR Codes Invented and How They Changed the World*. Online. QR Tiger. © 2025, 19 November 2024. Dostupné z: <https://www.qrcode-tiger.com/when-were-qr-codes-invented>. [cit. 2025-04-03].
- [6] ROWAN KELLEHER, Suzanne. *Why QR Codes Are Popping Up Everywhere During The Pandemic: And How To Read Them In A Snap*. Online. In: FORBES MEDIA. Forbes. 16 Jun 2020. Dostupné z: <https://www.forbes.com/sites/suzannerowankelleher/2020/06/16/why-qr-codes-are-popping-up-everywhere-during-the-pandemic/>. [cit. 2025-04-03].
- [7] *QR Code Structure: Everything you need to know*. Online. Scanova Blog. 19 February 2015, 5 August 2024. Dostupné z: <https://scanova.io/blog/qr-code-structure/>. [cit. 2025-04-03].
- [8] ANKHI. *The Ultimate Guide to QR Codes: Everything You Need to Know*. Online. UNIQUOTE PHYGITAL. UNIQUOTE. © 2025, 10 December 2024. Dostupné z: <https://www.uniquote.com/blog/qr-code-basics/comprehensive-guide-to-qr-code>. [cit. 2025-04-03].
- [9] *QR Code Standardization*. Online. DENSO WAVE. QRcode.com. Dostupné z: <https://www.qrcode.com/en/about/standards.html>. [cit. 2025-04-03].
- [10] *Types of QR Code*. Online. DENSO WAVE. QRcode.com. Dostupné z: <https://www.qrcode.com/en/codes/>. [cit. 2025-04-03].

- [11] *Error Correction Feature*. Online. DENSO WAVE. QRcode.com. Dostupné z: https://www.qrcode.com/en/about/error_correction.html. [cit. 2025-04-03].
- [12] *QR Code Error Correction: Everything You Need To Know in 2025!*. Online. Scanova Blog. 26 July 2018, 5 August 2024. Dostupné z: <https://scanova.io/blog/qr-code-error-correction/>. [cit. 2025-04-03].
- [13] *Information capacity and versions of the QR Code*. Online. DENSO WAVE. QRcode.com. Dostupné z: <https://www.qrcode.com/en/about/version.html>. [cit. 2025-04-03].
- [14] *Qrcodemonkey*. Online. Dostupné z: <https://www.qrcode-monkey.com/>. [cit. 2025-04-17].
- [15] *QR Code Generator*. Online. © 2025. Dostupné z: <https://www.qr-code-generator.com>. [cit. 2025-04-17].
- [16] *ISO quality standards for QR codes: ensuring accuracy and compliance*. Online. QRcodeKit. 15 November 2024. Dostupné z: <https://qrcodekit.com/news/iso-quality-standards-for-qr-codes/>. [cit. 2025-04-17].
- [17] *7 Genuine Advantages of QR Codes: Unveiling Benefits*. Online. Scanova Blog. 16 February 2016, 5 August 2024. Dostupné z: <https://scanova.io/blog/advantages-of-qr-code/>. [cit. 2025-04-17].
- [18] COPER, Becca. *The Pros and Cons of QR Codes*. Online. Image.works. 05 October 2022. Dostupné z: <https://www.imageworksdirect.com/blog/pros-and-cons-qr-codes>. [cit. 2025-04-17].
- [19] *QR Code Advantages and Limitations*. Online. ARTIFEX SOFTWARE. ByteScout. © 2006 - 2025. Dostupné z: <https://bytescout.com/blog/2019/04/qr-code-advantages-and-limitations.html>. [cit. 2025-04-18].
- [20] *50 Creative Uses of QR Codes*. Online. FLYTE NEW MEDIA. Flyte. Dostupné z: <https://www.takeflyte.com/blog/50-creative-uses-of-qr-codes>. [cit. 2025-04-18].
- [21] *The 15 Most Practical QR Code Uses*. Online. Supercode. 28 Apr 2021. Dostupné z: <https://www.supercode.com/blog/the-15-most-practical-uses-for-qr-codes>. [cit. 2025-04-18].

- [22] *QR Codes - what's the real risk?* Online. National Cyber Security Centre. 8 February 2024. Dostupné z: <https://www.ncsc.gov.uk/blog-post/qr-codes-whats-real-risk>. [cit. 2025-04-18].
- [23] *Quishing*. Online. SOSAFE. SoSafe. © 2025. Dostupné z: <https://sosafe-awareness.com/glossary/quishing/>. [cit. 2025-04-21].
- [24] CAMPANELLA, Sam. *What is quishing (QR code phishing)?* Online. TECHTARGET. WhatIs? © 1999 - 2025, September 2024. Dostupné z: <https://www.techtarget.com/whatis/definition/QR-code-phishing>. [cit. 2025-04-21].
- [25] *10 Real-Life Quishing Attack Examples to Strengthen Your Cybersecurity*. Online. KEEPNET LABS. Keepnet. 2024. Dostupné z: <https://keepnetlabs.com/blog/10-real-life-quishing-attack-examples-to-strengthen-your-cybersecurity>. [cit. 2025-04-21].
- [26] POSEY, Brien. *Understanding QR code security issues for enterprise devices*. Online. TECHTARGET. Search Mobile Computing. 05 Aug 2022. Dostupné z: <https://www.techtarget.com/searchmobilecomputing/tip/Understanding-QR-code-security-issues-for-enterprise-devices>. [cit. 2025-04-31].
- [27] *Social Engineering*. Online. SOSAFE. SoSafe. © 2025. Dostupné z: <https://sosafe-awareness.com/glossary/social-engineering/>. [cit. 2025-04-31].
- [28] CABRA, Paleg. *QR Code Red: Quishing Attacks and How to Prevent Them*. Online. Perception Point. 24 August 2023. Dostupné z: <https://perception-point.io/blog/qr-code-red-quishing-attacks-and-how-to-prevent-them/>. [cit. 2025-04-31].
- [29] *Introduction of QR code attacks and countermeasures*. Online. HKCERT. 20 Jan 2022. Dostupné z: <https://www.hkcert.org/blog/introduction-of-qr-code-attacks-and-countermeasures>. [cit. 2025-04-31].
- [30] *Qryptal explained in TWO MINUTES*. Online. Qryptal. ©2025. Dostupné z: <https://www.qryptal.com/landingpages/signed-qr-code/>. [cit. 2025-05-07].
- [31] *Hashing Algorithm Overview: Types, Methodologies & Usage*. Online. Okta. © 2025, 29. 8. 2024. Dostupné z: <https://www.okta.com/identity-101/hashing-algorithms/>. [cit. 2025-05-07].

- [32] BASATWAR, Govindraj. *Hashing Algorithms – An In-Depth Guide To Understanding Hash Functions*. Online. DoveRunner. 19 March 2025. Dostupné z: <https://doverunner.com/blogs/guide-to-understand-hashing-algorithms/>. [cit. 2025-05-07].
- [33] WESTON, Georgia. *What is SHA 256 Algorithm and How does it work?* Online. 101 Blockchains. 29 January 2025. Dostupné z: <https://101blockchains.com/sha-256-algorithm-explained/>. [cit. 2025-05-07].
- [34] *Hashing Algorithms*. Online. Saylor Academy. 24 July 2024. Dostupné z: <https://learn.saylor.org/mod/book/tool/print/index.php?id=36370>. [cit. 2025-05-07].
- [35] HASHEMI-POUR, Cameron; S. GILLIS, Alexander a LUTKEVICH, Ben. *Digital signature*. Online. TECHTARGET. Search Security. © 2000 - 2025, June 2024. Dostupné z: <https://www.techtarget.com/searchsecurity/definition/digital-signature>. [cit. 2025-05-07].
- [36] *Digital signatures: What they are & how they work*. Online. Sectigo. 15 Sep 2020. Dostupné z: <https://www.sectigo.com/resource-library/how-digital-signatures-work>. [cit. 2025-05-07].
- [37] *What are the digital signature algorithms?* Online. Tencent Cloud. 2025, 08. 05. 2025. Dostupné z: <https://www.tencentcloud.com/techpedia/108655>. [cit. 2025-05-08].
- [38] *Digital Signatures and Certificates*. Online. GeeksforGeeks. 07 Apr 2025. Dostupné z: <https://www.geeksforgeeks.org/digital-signatures-certificates/>. [cit. 2025-05-08].
- [39] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. *Digital Signature Standard (DSS)*. Online. 3 February 2023. Dostupné z: <https://doi.org/10.6028/NIST.FIPS.186-5>. [cit. 2025-05-08].
- [40] JOSEFSSON, S. a LIUSVAARA, I. *Edwards-Curve Digital Signature Algorithm (EdDSA)*. Online. Datatracker IETF. January 2017. Dostupné z: <https://datatracker.ietf.org/doc/html/rfc8032>. [cit. 2025-05-08].
- [41] HART, Laurence. *Benefits and challenges of electronic signatures*. Online. TECHTARGET. Search Content Management. 30 Jul 2024. Dostupné z: <https://www.techtarget.com/searchcontentmanagement/answer/252523027/What-are-the-pros-and-cons-of-electronic-signatures>. [cit. 2025-05-08].

- [42] SEMAK, Aksen. *How to Create QR Code for Certificate?* Online. Certifier. @ 2025, 21 January 2025. Dostupné z: <https://certifier.io/blog/how-to-create-a-certificate-with-a-qr-code>. [cit. 2025-05-08].
- [43] *FrameQR®*. Online. DENSO WAVE. DENSO WAVE. Dostupné z: <https://www.denso-wave.com/en/system/qr/product/frame.html>. [cit. 2025-05-13].
- [44] BROOK, Chris. *What Is Digital Watermarking?* Online. FORTRA. Data Loss Prevention. 14 June 2023. Dostupné z: <https://www.digitalguardian.com/blog/digital-watermarking>. [cit. 2025-05-13].
- [45] TANGRI, Yashika. *QR Code with Logo: Everything You Need To Know!*. Online. Scanova Blog. 26 October 2018, 20 August 2024. Dostupné z: <https://scanova.io/blog/qr-code-with-logo-a-comprehensive-guide/>. [cit. 2025-05-13].
- [46] CHOW, Yang-Wai; SUSILO, Willy; TONIEN, Joseph a ZONG, Wei. *A QR Code Watermarking Approach based on the DWT-DCT Technique*. Online. Dostupné z: <https://core.ac.uk/download/pdf/84840911.pdf>. [cit. 2025-05-13].
- [47] *Steganography: What Is It and How Does Steganography Work?* Online. Okta. © 2025, 30. 08. 2024. Dostupné z: <https://www.okta.com/identity-101/steganography/>. [cit. 2025-05-13].
- [48] SEMILOF, Margie a CLARK, Casey. *Steganography*. Online. TECHTARGET. Search Security. © 2000 - 2025. Dostupné z: <https://www.techtarget.com/searchsecurity/definition/steganography>. [cit. 2025-05-15].
- [49] *QR.io*. Online. QR.io. © 2025. Dostupné z: <https://qr.io/>. [cit. 2025-05-15].
- [50] *Digital Watermarks That Deliver More*. Online. Digimarc. © 2025. Dostupné z: <https://www.digimarc.com/product-digitization/data-carriers/digital-watermarks>. [cit. 2025-05-15].
- [51] *QR Code generator library*. Online. Project Nayuki. © 2025, 24. 11. 2021. Dostupné z: <https://www.nayuki.io/page/qr-code-generator-library>. [cit. 2025-05-15].
- [52] *Welcome to Zint*. Online. Zint. © 2008 - 2025, 11 March 2025. Dostupné z: <https://zint.org.uk/>. [cit. 2025-05-15].
- [53] *The PyScript User Guide*. Online. PyScript. 2025. Dostupné z: <https://docs.pyscript.net/2025.3.1/user-guide/>. [cit. 2025-03-15].

- [54] OMOROKUNWA, Alexander. *Facts You Should know About Pyscript: A web Development Framework*. Online. In: Foss Naija: Experiencing Opensource... 31 May 2022. Dostupné z: <https://fossnaija.com/facts-about-opensource-pyscript-web-framework/>. [cit. 2025-03-15].
- [55] GITHUB, INC. *What is GitHub Pages?* Online. GitHub Docs. © 2025. Dostupné z: <https://docs.github.com/en/pages/getting-started-with-github-pages/what-is-github-pages>. [cit. 2025-03-15].

SEZNAM OBRÁZKŮ

Obrázek 1 – Japonská stolní hra Go	16
Obrázek 2 – Struktura QR kódu	18
Obrázek 3 – Směr čtení dat z QR kódu	18
Obrázek 4 – Typy QR kódů.....	19
Obrázek 5 – Proces získání dat využitím Quishingu	24
Obrázek 6 – Znázornění hashovací funkce	29
Obrázek 7 – Princip digitálního podpisu a jeho ověření.....	31
Obrázek 8 – Infrastruktura technologie PyScript	41
Obrázek 9 – Změna velikosti QR kódu při vložení digitálního podpisu	45
Obrázek 10 – Detekovaný podpis z QR kódu použitím LSB steganografie	45
Obrázek 11 – Vystavený soubor s podpisem ke stažení	46
Obrázek 12 – Ukázky dvou přístupů aplikace vodoznaku	47
Obrázek 13 – Sekce ve webové aplikaci	54
Obrázek 14 – Wireframe aplikace QRebuild.....	55
Obrázek 15 – Odkaz na webovou aplikaci	56
Obrázek 16 – Sekce pro generování QR kódu.....	58
Obrázek 17 – Sekce pro dekódování podpisu z obrázku QR kódu	59
Obrázek 18 – Sekce ověření QR kódu.....	60
Obrázek 19 – Sekce detekování QR kódu	61
Obrázek 20 – Podepsaný QR kód bez vodoznaku.....	66
Obrázek 21 – Diagram aplikace digitálního podpisu.....	69
Obrázek 22 – QR kód s vodoznakem bez podpisu	70
Obrázek 23 – Diagram aplikace vodoznaku a odznaku.....	74
Obrázek 24 – Digitálně podepsaný QR kód s vodoznakem	76
Obrázek 25 – Diagram generování QR kódu.....	77
Obrázek 26 – Diagram dekódování QR kódu.....	78
Obrázek 27 – Diagram ověření QR kódu	80
Obrázek 28 – Odkaz na seznam vydaných verzí aplikace QRebuild	82
Obrázek 29 – Uživatelské rozhraní aplikace QRebuild.....	84
Obrázek 30 – Zobrazení rekonstruovaného QR kódu v GUI verzi aplikace	89
Obrázek 31 – Zobrazení rekonstruovaného QR kódu v CLI verzi aplikace.....	89
Obrázek 32 – Průběh automatického sestavení a distribuce.....	90
Obrázek 33 – Distribuce sestavených QRebuild aplikací.....	93

SEZNAM TABULEK

Tabulka 1 – Klíčové události ve vývoji QR kódů.....	17
Tabulka 2 – Kapacita QR kódu ve verzi 1 s pouze numerickými daty.....	20
Tabulka 3 – Srovnání hashovacích algoritmů.....	30
Tabulka 4 – Shrnutí dostupných nástrojů pro generování QR kódů.....	38
Tabulka 5 – Analýza výhod a nevýhod metod zabezpečení QR kódů.....	48
Tabulka 6 – Funkcionální požadavky	51
Tabulka 7 – Nefunkcionální požadavky	52
Tabulka 8 – Parametry CLI verze aplikace QRebuild	86

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

AIM	Association for Automatic Identification and Mobility
CD	Continous Delivery
CI	Continous Integration
CLI	Command Line Interface
CSS	Cascading Style Sheets
DCT	Discrete Cosine Transform
DOM	Document Object Model
DWT	Discrete Wavelet Transform
DSA	Digital Signature Algorithm
EAN	European Article Number
ECDSA	Elliptic Curve Digital Signature Algorithm
ECL	Error Correction Level
EdDSA	Edwards-curve Digital Signature Algorithm
FIPS	Federal Information Processing Standard
GDPR	General Data Protection Regulation
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTPS	Hypertext Transfer Protocol Secure
JIS	Japan Industrial Standards
JS	JavaScript
JSON	JavaScript Object Notation
LSB	Last Significant Bit
MPA	Multi Page Application
PNG	Portable Network Graphics
PSS	Probabilistic Signature Scheme"

QR	Quick Response
RGB	Red Green Blue
RGBA	Red Green Blue Alpha
RSA	Rivest–Shamir–Adleman
SHA	Secure Hash Algorithm
SMS	Short Message Service
SPA	Single Page Application
TOML	Tom’s Obvious Minimal Language
UI	User Interface
UPC	Universal Product Code
URL	Uniform Resource Locator

SEZNAM PŘÍLOH

PŘÍLOHA P I: OBSAH SOUBORU ZIP.ZIP

PŘÍLOHA P I: OBSAH SOUBORU ZIP.ZIP

Struktura obsahu přiloženého souboru zip.zip:

- Zdrojový kód webové aplikace **QRGuardian**

```
PS C:\dev\QRGuardian> Show-Tree
.
├── .github
│   └── workflows
│       └── github_pages.yml
├── css
│   ├── base.css
│   ├── fileUpload.css
│   ├── navigation.css
│   └── responsive.css
├── img
│   ├── flags
│   │   ├── cz.svg
│   │   ├── gb.svg
│   │   └── sk.svg
│   ├── github-mark-white.png
│   ├── QRGuardian-logo.png
│   ├── QRGuardian-logo-wide.png
│   ├── signed.png
│   └── utb-icon.png
├── js
│   ├── translations
│   │   ├── cs.json
│   │   ├── en.json
│   │   └── sk.json
│   ├── downloadLinks.js
│   ├── fileUpload.js
│   ├── main.js
│   ├── navigation.js
│   ├── translations.js
│   └── ui.js
├── scripts
│   ├── base64_signed_badge.py
│   ├── crypto.py
│   ├── event_handlers.py
│   ├── main.py
│   ├── qr_content.py
│   ├── qr_generator.py
│   ├── stego.py
│   └── ui_handlers.py
├── index.html
├── pyscript.toml
└── README.md
```

- Zdrojový kód desktopové aplikace **QRebuild**

```
PS C:\dev\QRebuild> Show-Tree
.
├── .github
│   └── workflows
│       └── build-and-release.yaml
├── cli
│   └── argparse.py
├── core
│   ├── content_analyzer.py
│   ├── qr_extractor.py
│   ├── qr_generator.py
│   └── utils.py
├── img
│   ├── logo.icns
│   ├── logo.ico
│   ├── logo.png
│   ├── logo-square.png
│   └── logo-wide.png
├── .gitignore
├── cli_main.py
├── gui_app.py
├── gui_main.py
├── README.md
└── requirements.txt
```