UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

DIPLOMOVÁ PRÁCE

2013 Bc. Vojtěch Oram

Univerzita pardubice

Fakulta elektrotechniky a informatiky

Captcha test

Bc. Vojtěch Oram

Diplomová práce

2013

Prohlašuji:
Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.
Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 15. května 2013

. . . . . .

Bc. Vojtěch Oram

PODĚKOVÁNÍ

Děkuji svému vedoucímu Ing. Lukáši Čeganovi, Ph.D. za vedení práce, cenné postřehy a rady. Dále bych rád poděkoval Lence Martinovské za psychickou podporu a všem přátelům, jež mi pomohli otestovat RoCAPTCHA test.

ANOTACE

Tato diplomová práce pojednává o problematice CAPTCHA testů, jež nejsou založeny na rozpoznávání zdeformovaného textu. Cílem je provést rešerši těchto testů a na jejím základě navrhnout a implementovat nový CAPTCHA test. Výstupem práce jsou dva CAPTCHA testy, z nichž jeden je veřejně dostupný přes API pro použití v internetových aplikacích třetích stran.

KLÍČOVÁ SLOVA

captcha, rotace, otáčení, vzpřímená poloha, segmentace, django, python

TITLE

Captcha test

ANNOTATION

This diploma thesis discuss CAPTCHA test which are not based on deformed text recognition. Purpose is to propose and implement new CAPTCHA test based on a research of tests mentioned above. As a result there are two CAPTCHA tests, one of them publicly accessible through API for use in third party internet applications.

KEYWORDS

captcha, rotation, upright position, segmentation, django, python

CITACE

Vojtěch Oram: Captcha test, diplomová práce, Pardubice, Univerzita Pardubice, 2013

Obsah

1 CAPTCHA test **11**
- 1.1 Problematika . . . 11
  - 1.1.1 Bezpečnost . . . 11
  - 1.1.2 Použitelnost . . . 12
- 1.2 Rozdělení CAPTCHA testů . . . 13
  - 1.2.1 Rozpoznávání textu . . . 13
  - 1.2.2 Rozpoznávání zvuku . . . 13
  - 1.2.3 Rozpoznávání obrazu . . . 15
  - 1.2.4 Sémantické a logické . . . 18
  - 1.2.5 Nevyžadující interakci . . . 18
  - 1.2.6 Individuální autentizace . . . 19
  - 1.2.7 Centralizovaná autentizace . . . 19
- 1.3 Analýza současných CAPTCHA testů . . . 20
  - 1.3.1 Rozpoznávání obrazu . . . 20
  - 1.3.2 Sémantické a logické . . . 26
  - 1.3.3 Nevyžadující interakci . . . 27
- 1.4 Vyhodnocení . . . 28

2 Analýza **32**
- 2.1 Obecné principy . . . 32
  - 2.1.1 Bezpečnost . . . 32
  - 2.1.2 API . . . 34
- 2.2 RoCAPTCHA . . . 34
  - 2.2.1 Vhodnost otáčení obrázků . . . 35
  - 2.2.2 Bezpečnost . . . 37
  - 2.2.3 Získávání obrázků . . . 39
  - 2.2.4 Hodnocení obrázků . . . 40
- 2.3 SegCAPTCHA . . . 43
  - 2.3.1 Princip . . . 44
  - 2.3.2 Bezpečnost . . . 44

3 Implementace **45**
- 3.1 Použité technologie . . . 45
- 3.2 Webová aplikace . . . 48
- 3.3 API . . . 49
- 3.4 RoCAPTCHA . . . 50
  - 3.4.1 Import zdrojových obrázků . . . 51
  - 3.4.2 Ohodnocení obrázků . . . 52
  - 3.4.3 Generování . . . 53
  - 3.4.4 Verifikace . . . 55
  - 3.4.5 Rozhraní . . . 55
  - 3.4.6 Pluginy . . . 56
- 3.5 SegCAPTCHA . . . 57
  - 3.5.1 Generování . . . 57
  - 3.5.2 Verifikace . . . 58
  - 3.5.3 Rozhraní . . . 59
- 3.6 Uživatelské testování RoCAPTCHA testu . . . 59

3.6.1 Systém testování 59
3.6.2 Vyhodnocení 60

Seznam obrázků

1.1 Zvukový CAPTCHA test HIPUU. 14
1.2 Ukázka jigsaw puzzle. 15
1.3 Ukázka různých CAPTCHA testů, založených na orientaci obrázku. 17
1.4 Key CAPTCHA 20
1.5 Asirra CAPTCHA test. 21
1.6 Ukázka Confident CAPTCHA testu. 22
1.7 PlayThru CAPTCHA. 23
1.8 Typy Solve media CAPTCHA testů. 23
1.9 Slide to fit captcha test. 24
1.10 VouchSafe CAPTCHA. 25
1.11 Ironclad CAPTCHA. 25
1.12 PICATCHA. 26
1.13 Egglue semantic CAPTCHA. 26

2.1 Životní cyklus CAPTCHA testu. 34
2.2 Ukázka dotykového ovládání RoCAPTCHA testu. 35
2.3 Ukázka fotografií, focených pod různým úhlem vůči rovině horizontu. 36
2.4 Reprezentace přímky pomocí křivky s parametry (r, θ). 42
2.5 Grafické znázornění metody ohodnocování přímek podle jejich úhlu ke svislé ose. 43

3.1 Postup detekce úseček v obrazu. 53
3.2 Ukázka detekce lidských obličejů. 54
3.3 Ukázka finálního vygenerovaného obrázku pro jeden RoCAPTCHA test. 54
3.4 Ukázka uživatelského rozhraní RoCAPTCHA testu. 56
3.5 Ukázka vygenerovaných obrázků pro SegCAPTCHA test. 58
3.6 Ukázka experimentálního rozhraní SegCAPTCHA testu. 59
3.7 Graf četností délek řešení RoCAPTCHA testu. 61
3.8 Graf četností odchylek úhlů od vzpřímené polohy. 61
3.9 Graf úspěšnosti otáčení obrázků do vzpřímené polohy. 62

A.1 Ukázka vzhledu vytvořené webové aplikace. 70

B.1 ER diagram SegCAPTCHA testu. Nejsou zde zobrazeny tabulky, jež se generují automaticky pro každý Django projekt. 71
B.2 ER diagram RoCAPTCHA testu. Nejsou zde zobrazeny tabulky, jež se generují automaticky pro každý Django projekt. 72

C.1 Diagram aktivit, modelující proces zobrazení CAPTCHA testu u klienta. 73
C.2 Diagram aktivit, modelující proces verifikace řešení CAPTCHA testu. 74

E.1 Ukázka verifikačních obrázků. 76

Seznam tabulek

1.1 Shrnutí poznatků o vizuálních CAPTCHA testech. 29
1.2 Shrnutí poznatků o sémantických a logických CAPTCHA testech. 30
1.3 Shrnutí poznatků CAPTCHA testech nevyžadující interakci. 31

3.1 Podrobný popis API. 50
3.2 Porovnání RoCAPTCHA testu s podobnými CAPTCHA testy. 63

D.1 Statusové kódy, jež mohou být vráceny CAPTCHA serverem v odpovědi. 75
D.2 Stavy, jež může CAPTCHA test nabývat. 75

Úvod

S přibývajícím množstvím internetových služeb roste i množství pokusů o jejich zneužití, především za účelem zisku. Zde přistupuje ke slovu CAPTCHA test, umožňující rozlišení legitimních uživatelů od útočníků, zneužívajících tyto služby s pomocí automatizovaných metod. V současné době existuje mnoho druhů těchto testů, založených na různých principech. Žádný z nich se ale nemůže pyšnit dokonalou ochranou před útočníky a zároveň 100% úspěšností při řešení legitimními uživateli.

Účelem této diplomové práce je zhodnotit současné CAPTCHA testy a na základě získaných poznatků navrhnout další řešení, jež by mohla posunout rozpoznávání člověka od počítače zase o krok dál. Je pravděpodobné, že nalezená řešení budou velmi brzy překonána, vzhledem k pokrokům v oblasti rozpoznávání obrazu a umělé inteligence. Překonávání nových CAPTCHA testů samo o sobě stojí za pokroky v této oblasti vědy, neboť inspiruje další lidi k nalezení netradičních řešení.

Toto zadání jsem si vybral vzhledem ke svému zaměření na internetové technologie, protože mi poskytuje skvělou příležitost, jak se zdokonalit právě v této oblasti.

Práce je rozdělena do tří kapitol. Kapitola 1 rozebírá problematiku CAPTCHA testů z hlediska bezpečnosti, použitelnosti a dalších hledisek. Součást této kapitoly tvoří rešerše současných CAPTCHA testů a vzájemné porovnání. Následuje 2 navržení nových řešení a jejich analýza. Poslední kapitola popisuje 3 postup implementace navrhovaných řešení a zhodnocení.

Kapitola 1

CAPTCHA test

1.1 Problematika

Název CAPTCHA test navrhl Luis Von Ahn *et al.*[27] a znamená ***C**ompletely **A**utomatic **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part* – jde tedy o test pro rozlišení člověka od počítače. Tato zkratka může být matoucí, protože CAPTCHA testy jsou ve skutečnosti reverzním Turingovým testem, neboť se dotazují člověka, ne počítače. Také naprostá většina CAPTCHA testů není plně automatická, ale je po člověku vyžadováno řešení zadaného testu. Hlavním cílem CAPTCHA testů použitých v internetových aplikacích je rozpoznávání lidských návštěvníků webových stránek od těch „ostatních“, například automatizovaných počítačových programů.

V roce 1997 Andrei Broder a jeho kolegové, vyvinuli tento test jako odpověď na problém s automatickými registracemi emailových účtů na Yahoo!. V v roce 2001 si tento způsob boje proti automatickému spamování patentovali[6]. Původní CAPTCHA test spočíval v rozpoznání a přepsání různě deformovaných písmen a číslic vygenerovaných do obrázku. Tento přístup se používá ve velké míře i dnes, nicméně bylo vyvinuto i mnoho odlišných CAPTCHA testů. Například rozpoznávání objektů na obrázku, mluveného textu, hraní her a další, kterým se budeme věnovat dále. Žádný z těchto CAPTCHA testů však není naprosto dokonalý. S rostoucí obtížností vyřešení testu pro počítač většinou roste obtížnost i pro člověka, což vede k frustraci řešitelů a ke snížení přístupů ke zdroji, který má CAPTCHA test chránit. Výkon počítačů neustále roste, takže současné CAPTCHA testy mohou být během pár let prolomeny jen s pomocí hrubé síly výpočetní techniky. Je velmi pravděpodobné, že za určitou dobu nebude možné v internetovém prostředí člověka od počítače vůbec odlišit.

CAPTCHA test by měl být [18, 5]:

- **Automatizovaný** – generování a hodnocení by mělo probíhat automaticky bez zásahu člověka.
- **Otevřený** – podkladová databáze a použitý algoritmus by měly být veřejně známy. I přes to by ale měl tento test zůstat bezpečný.
- **Použitelný** – test by měl být řešitelný všemi lidmi bez rozdílu jazyka, vzdělání, fyzického stavu nebo schopnosti vnímání. Navíc by měl jít vyřešit v rozumném čase.
- **Bezpečný** – test by měl být dostatečně složitý, aby ho nebylo možné vyřešit automaticky s použitím počítače.

1.1.1 Bezpečnost

Dle zdrojů [18, 25] existují 4 základní typy útoků pro prolomení CAPTCHA testů:

- **Náhodné hádání**

- **Přímé přiřazování**
- **Strojové učení**
- **Lidské zdroje**

Prvním z nich je náhodné hádání. Tato metoda velmi závisí na typu CAPTCHA testu. Například u opisování textu zde máme tolik možných variant, kolik jen je možno sestavit slov v daném jazyce. Pokud vybíráme k z n možností, pravděpodobnost náhodného uhádnutí je dána vzorcem k/n. Obecně by měla být pravděpodobnost náhodného uhádnutí správného řešení CAPTCHA testu pod 0,01% [8].

Útok přímým přiřazováním předpokládá CAPTCHA test s omezenou bází dat. Samotný útok spočívá ve vytvoření databáze správných řešení, která je poté použita při řešení zadaných CAPTCHA testů. S omezenou bází dat nemá smysl otázka, zda je možné tento útok úspěšně provést, ale zda by se to útočníkům z ekonomického hlediska vyplatilo. Například u CAPTCHA testu Asirra při použití 3.000.000 obrázků z databáze a při 12 obrázcích u jednoho testu by útočník kompromitoval 95% této databáze již za 750.000 pokusů [16].

Strojové učení představuje nejslibnější metodu útoku pro obrázkové CAPTCHA testy. Princip spočívá v extrakci důležitých topologických prvků ze vzorového obrázku a následné hodnocení shody těchto prvků se získanými prvky u zkoumaného obrázku. Tímto způsobem můžeme například rozpoznat na fotografii lidský obličej.

Díky levné pracovní síle třetího světa je v současné době možné zaplatit si řešení CAPTCHA testu lidmi [25], proti čemuž je prakticky nemožné se bránit, neboť to podrývá samotný princip CAPTCHA testu. Nejde zde pouze o rozpoznání člověka od počítače, ale také o rozpoznání legitimního uživatele od uživatele najmutého pro prolomení CAPTCHA testu. Například společnost *Death by Captcha*[1] nabízela v době psaní této diplomové práce vyřešení 1000 CAPTCHA testů za $1,39.

1.1.2 Použitelnost

Obtížnost vyřešení jednotlivých CAPTCHA testů může člověk od člověka kolísat, například pro slepce je vizuální CAPTCHA test prakticky neřešitelný. Proto je u nás dáno přímo zákonem[19], aby všechny služby poskytované státem (to zahrnuje i webové služby) byly dostupné také hendikepovaným a zdravotně postiženým uživatelům. Vzhledem k decentralizaci internetu se takovéto pravidlo nedá vynutit. Záleží hlavně na jednotlivých provozovatelích webových služeb, aby mysleli také na hendikepované uživatele. U současných CAPTCHA testů převažuje vizuální varianta v kombinaci se zvukovou pro nevidomé uživatele.

Další z prvků použitelnosti představuje míra úspěšnosti uživatelů a míra úspěšnosti počítačů při řešení CAPTCHA testu. Udává se, že prolomení CAPTCHA testu automatickým útokem by mělo být úspěšné v méně než 0,01% případů, zatímco člověk by měl být schopen tento test úspěšně vyřešit v 90% případů [8].

[1] http://www.deathbycaptcha.com/

1.2 Rozdělení CAPTCHA testů

CAPTCHA testy můžeme rozdělit podle různých hledisek. Obecně se používá rozdělení podle typu úlohy na rozpoznávání textu, zvuku a obrazu [5, 21, 10]. Tyto metody jsou podrobněji popsány v následujících sekcích. V této práci jsou zahrnuty ještě další metody implementace CAPTCHA testu, jako například CAPTCHA testy na sémantickém a logickém principu (1.2.4), testy nevyžadující uživatelovu interakci (1.2.5), individuální autentizaci (1.2.6) a centralizovanou autentizaci (1.2.7).

1.2.1 Rozpoznávání textu

Rozpoznávání deformovaného textu je vůbec první metoda, která byla u CAPTCHA testů použita, viz 1.1. Princip spočívá v generování různě deformovaných písmen nebo textu (a případně čar a obrazců) do obrázku, který se následně zobrazí testovanému subjektu. Ten pak přečte z obrázku zadaný text a vepíše ho do textového pole. Text z tohoto pole je následně odeslán společně s dalšími daty od uživatele. Na aplikačním serveru proběhne porovnání uživatelem zadaného textu s textem, který byl do obrázku vygenerován. Shoda znamená úspěšné splnění CAPTCHA textu.

Největší výhoda tohoto postupu spočívá v jednoduchosti generování a prakticky neomezeném množství variant generovaného textu a také způsobu, jakým lze tento text generovat do obrázku. Další výhodou je nezávislost na dorozumívacím jazyce (pouze na dané základní abecedě).

Společně s vývojem nových způsobů jak generovat textové CAPTCHA testy také roste množství metod, které se snaží tyto testy vyřešit automaticky. Pro automatické řešení textových CAPTCHA testů se používá OCR (Optical Characted Recognition) software, umožňující rozpoznat jednotlivá písmena v obrázku. V reakci na použití OCR přišli tvůrci CAPTCHA testů s novým postupem, a to shlukováním písmen (CCT, Crowding Characters Together). Částečný překryv písmen znemožnil přímé rozpoznání přes OCR – bylo nejdříve třeba provést jejich segmentaci. Nicméně ani segmentace se neukázala jako příliš velký problém. Experimentálně bylo dosaženo více než 46% úspěšnosti při rozpoznání takto generovaného textu [8].

Rozpoznání deformovaného textu může být v některých případech velmi obtížné až nemožné, například pro lidi s očními vadami, dyslektiky a mentálně postižené. Při složitějších textových deformacích je i pro člověka bez těchto vad poměrně náročné test úspěšně splnit. Přesto je tento CAPTCHA test aktuálně (březen 2013) nejpoužívanější.

1.2.2 Rozpoznávání zvuku

Jak bylo řečeno v sekci 1.1.2, je při vytváření CAPTCHA testů třeba myslet i na uživatele s různými poruchami vnímání. Především jde o uživatele se zrakovým postižením, kterých bylo v roce 2009 v USA 10 milionů, z toho 1,3 milionu úplně slepých [24]. Zde nastupují ke slovu CAPTCHA testy na principu rozpoznání zvuku.

Nejpoužívanější varianta tohoto testu spočívá v generování zvukového záznamu čteného textu s pomocí skládání jednotlivých písmen z předem nahraného záznamu, nebo přímo strojově syntetizovaných. Poté se může do záznamu přidat šum a zkreslení pro zhoršení možného strojového rozpoznání. Tento

zvukový záznam se poté přehraje uživateli. Ten musí do textového pole zadat text, který uslyší v záznamu. Po odeslání se na straně serveru porovná uživatelem zadaný text s původním textem, jež byl generovaný do zvukového záznamu. Při shodě je uživatel označen jako člověk.

Už samotné použití rozpoznávání zvuku na rozdíl od vizuálních metod sebou přináší řadu problémů. Především jde o časovou náročnost, neboť informace ve zvukové podobě může být podaná pouze lineárně – aby uživatel měl přehled o celé informaci v záznamu, musí si jej nejdříve celý poslechnout. Tento fakt kontrastuje s obrázky, ze kterých dokáže uživatel vstřebat informace mnohem rychleji, protože je vidí v jeden okamžik jako celek.

Dále můžeme narazit na problém s rozhraním. Uživatelé s postižením zraku často používají zařízení pro čtení textu na obrazovce, nazývané *Screen reader*. Pro tyto uživatele je poměrně obtížná navigace na webové stránce a tím i ovládání zvukového CAPTCHA testu. J. Bigham a A. Cavender ve své práci [1] vyvinuli nový způsob, jakým by mohli uživatelé se zrakovým postižením ovládat zvukový CAPTCHA test. Umožňuje uživatelům zároveň psát řešení CAPTCHA testu a ovládat přehrávání zvuku pomocí klávesových zkratek. Navíc je jim umožněno nejen prosté přehrání, ale i pozastavení záznamu a vrácení se o určitý časový úsek. Experimentálně dokázali, že při použití tohoto řešení lze dosáhnout až o 59% lepší úspěšnosti při řešení standardního zvukového CAPTCHA testu.

Samotné rozpoznávání textu z deformovaného zvukového záznamu má obecně velmi špatnou uživatelskou úspěšnost. Výzkum ukázal, že jen 43% slepých uživatelů uspělo při řešení těchto testů na první pokus [1].

Bylo zde několik pokusů o použitelnější řešení na bázi rozpoznávání zvuku, ale mnohem méně, než u rozpoznávání obrazu. Haichang Gao *et al.* ve své studii [10] aplikoval opačný postup – namísto analyzování zvukového záznamu uživatelem nechal uživatele přečíst zadaný text a tím vytvořit záznam svého hlasu. Ten posléze analyzoval a na základě této analýzy zjistil, zda jde o člověka, nebo o počítač. Tato metoda vykázala 97% úspěšnost u uživatelů a 4% u experimentálního pokusu o prolomení počítačem. Průměrná doba řešení se pohybuje kolem 7,8 s. Vzhledem k velkým pokrokům na poli syntézy řeči je však použitelnost této metody nejistá.

Obrázek 1.1: Zvukový CAPTCHA test HIPUU, převzato z [24].

Další možnou metodu vyvinul Graig Sauer *et al.*. Tato metoda, nazývaná HIPUU[2] kombinuje obrázkový CAPTCHA test (konkrétně tagovací) se zvukovým (viz obrázek 1.1). Uživateli je prezentován set obrázků s určitým motivem, každý s vlastním textovým polem, kam musí uživatel napsat název předmětu na obrázku. U setu obrázků je také možnost přehrát si zvukový záznam, který určitým způsobem tyto obrázky charakterizuje. Tento způsob bohužel nebyl prozkoumán více do hloubky.

[2]Universally Usable Approach to Defeating Automated Bots

1.2.3 Rozpoznávání obrazu

Obrázkové CAPTCHA testy byly vyvinuty jako alternativa k rozpoznávání textu, protože zpracování obrazu je pro počítač těžší a pro uživatele většinou zábavnější[12, 11, 16, 18]. Tato metoda však požaduje pro prezentaci testu více prostoru, než metoda rozpoznávání textu.

Existuje vícero typů CAPTCHA testů, založených na rozpoznávání obrazu, v následujícím textu si představíme alespoň část z nich.

Tagovací

Pro prezentovaný set obrázků je třeba zadat společnou vlastnost buď vepsáním do textového pole, nebo vybráním z množiny možných vlastností [5]. U tohoto přístupu však musíme myslet na to, že neexistuje pouze jedna správná odpověď, ale naprostá většina objektů na obrázku se dá popsat více způsoby. I pokud bereme v potaz pouze jednoslovné odpovědi, stále musíme myslet na možná synonyma. V neposlední řadě mohou mít některé výrazy více významů.

Anomální

Anomální CAPTCHA test předkládá set obrázků, mezi nimiž má uživatel najít anomálii – obrázek, který mezi ostatní nepatří [5]. Tato technika však předpokládá *a priori* znalost obsahu těchto obrázků, aby bylo možno sestavit jednotlivé testy. Velkou výhodu představuje nezávislost na jazyce.

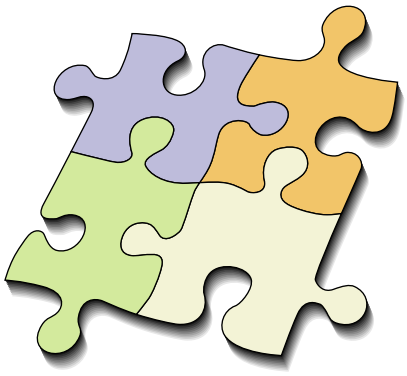
Rozlišovací

Stejně jako u předchozích dvou typů se zde nejdříve prezentuje uživateli set obrázků. Řešením není najít anomálie, nebo najít společnou vlastnost, ale rozdělit obrázky do skupin podle určitého zadaného znaku[5]. Typickými zástupci rozlišovacích testů jsou zde zmíněné testy Asirra, Confident CAPTCHA^{TM}a PICATCHA.

Společnou vlastností všech výše zmíněných obrázkových testů je potřeba znát nejdříve popis obsahu obrázků, které se používají pro generování jednotlivých testů. Ten se dá získat ručním zpracováním, nebo strojovým tagováním, které ale nemusí být vždy přesné.

Jigsaw puzzle

Jigsaw puzzle je starý herní mechanismus, spočívající ve skládání jednotlivých dílků obrázku k sobě tak, aby výsledkem byl kompletní původní obrázek 1.2. Za velkou výhodu se dá považovat fakt, že pro řešení CAPTCHA testu tohoto typu není třeba žádný textový popisek, není tedy závislý na jazykce. Navíc se tyto testy dají velmi dobře automaticky generovat. Pro použití v CAPTCHA testu stačí, když bude uživatel požádán o doplnění dvou chybějících dílků do obrázku. Není třeba nutit uživatele skládat celý obrázek. Haichang Gao et al. [11] doložil, že jigsaw



Obrázek 1.2: Ukázka jigsaw puzzle.

puzzle CAPTCHA test je lidmi dobře a poměrně rychle řešitelný
– může být vyřešen v čase pod 8s 98% účastníků s úspěšností 88,7%. V této práci bylo také experimentálně zjištěno, že 88% testovaných subjektů preferovalo tuto metodu přes opisováním deformovaného textu a 74% testovaných subjektů označilo toto řešení jako zábavné.

Z aktuálně funkčních zástupců jigsaw puzzle CAPTCHA testů je zde podrobněji popsána Key CAPTCHA 1.4.

Otáčení obrázků

V následujících odstavcích budou popsány 4 CAPTCHA testy, které ve svém principu používají rotaci obrázků do vzpřímené polohy. Žádné z těchto řešení není aktuálně použitelné pro nasazení, u Sketchy však existuje alespoň demo[3].

R. Gossweiler, M. Kamvar a S. Baluja [12] byli jedni z prvních, kteří aplikovali otáčení obrázků do vzpřímené polohy jako CAPTCHA test. Jejich **What's Up CAPTCHA** umožňovala otáčení obrázků pomocí posuvníku, jehož pozice byla svázána s úhlem otočení zadaného obrázku (ukázka rozhraní viz 1.3a). Ve své studii se také zabývali aspekty možného automatického rozpoznání polohy obrázku. Zde upozornili na nutnost správného filtrování vstupní sady obrázků, neboť u některých by mohla být automaticky rozpoznána poloha. Jde například o obrázky s nízko a vysokoúrovňovými prvky, které je možno lehce identifikovat, jako například obloha, tráva, obličeje a nebo text (viz 2.2.1). Dále je třeba vyřadit obrázky, ve kterých se uživatelům špatně orientuje, což lze udělat postupným vyřazováním detekcí určitého počtu neúspěchů při řešení konkrétního obrázku. Také experimentálně prokázali, že tento způsob řešení CAPTCHA testu je lidmi preferovaný před rozpoznáváním deformovaného textu a zároveň je více zábavný. Při nutnosti otočení tří obrázků do vzpřímené polohy s tolerancí 8° na každou stranu (16° celkem) můžeme vypočítat pravděpodobnost náhodného uhádnutí správného řešení jako $(16/360)^3 \doteq 0.00009$, což je hodnota mírně nad minimální požadovanou hodnotou.

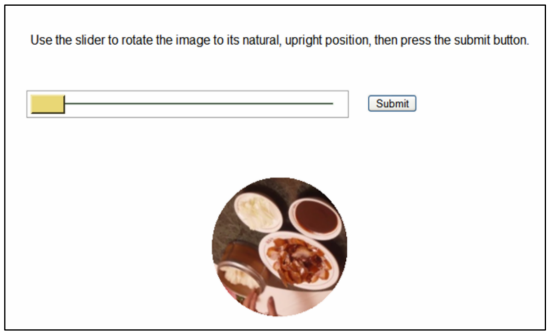
Multiple SEIMCHA [18] sice přímo neumožňuje uživateli otáčet obrázky, nicméně ve svém principu otáčení obsahuje. Jak lze vidět na obrázku 1.3b, první krok před generováním CAPTCHA testu představuje určení oblasti obrázku, která je nahoře (Key image). Z toho dostaneme klíčový obrázek. Původní i klíčový obrázek je následně otočen a promítnut na 3D objekt (válec, kužel, koule a jiné). Původní zdeformovaný obrázek je následně prezentován klientovi s úkolem, aby kliknutím označil horní část původního obrázku. Stejně jako u Asirry 1.3.1 zde byla použita technika téměř správné odpovědi pro zlepšení výsledků uživatelů. Výzkum ukázal, že úspěšnost řešení Multiple SEIMCHA testu se pohybuje kolem 92% a průměrný čas při řešení setu 8 obrázků je 26s. Je otázkou, zda by uživatelé preferovali tuto metodu před rozpoznáváním deformovaného textu. Výzkum na toto téma zatím nebyl proveden.

Sketcha [23] na rozdíl od ostatních předkládá inovativní způsob získávání zdrojových obrázků pro testy, a to generováním kreseb ze 3D modelů. Jako zdrojovou databázi modelů používá Google 3D Warehouse[4] (dnes nazývaná Trimble galerie 3D objektů). I když je tato databáze 3D modelů veřejně dostupná, generováním pouze kreseb (obrysů) těles tak může dosáhnout velmi dobré bezpečnosti proti přímému přiřazování (objekty se dají v určitých mezích natočit) i proti útokům skrze strojové učení, neboť

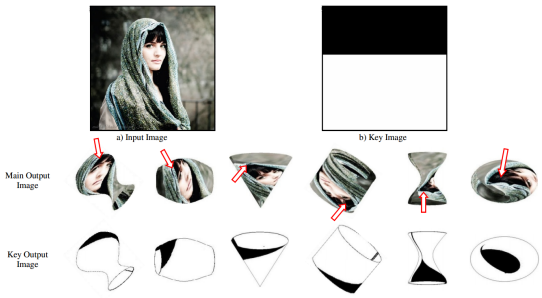
[3] http://www.sketcha.net/
[4] http://sketchup.google.com/3dwarehouse/

zde není mnoho detailů, podle kterých by se dalo orientovat. To ovšem může ztěžovat orientaci i lidským uživatelům. Konkrétní test sestává z deseti takto generovaných obrázků, které má uživatel možnost klikem myši otočit o 90° (viz obrázek 1.3c). Celkově může mít každý obrázek 4 stavy (0°, 90°, 180°, 270°). Při deseti obrázcích má útočník šanci pro náhodné uhádnutí správného řešení $(1/4)^{10} \doteq 0,000001$, což je přesná obecně uznávaná hodnota. Dále bylo experimentálně prokázáno, že při 10 předložených obrázcích v jednom testu tento test úspěšně splnilo 88% uživatelů. Generování obrázků pro CAPTCHA test ze 3D modelů má velký potenciál, neboť lze při generování použít nejen převedení do kresby, ale také další renderovací stylizace.

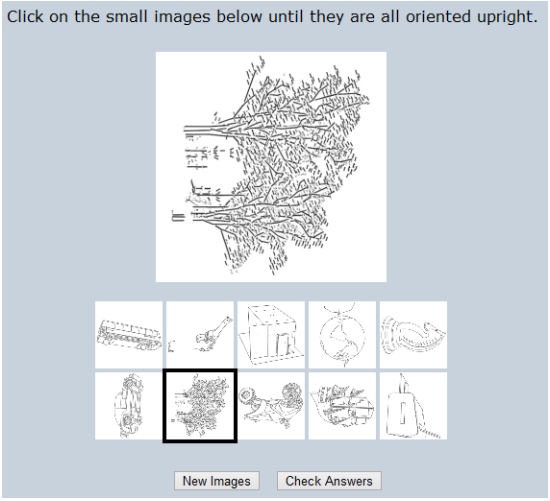
Posledním z CAPTCHA testů založených na otáčení obrázků je **Orientation based CAPTCHA** [3]. Stejně jako Sketcha nepoužívá přímo původní obrázky, ale nejdříve na ně aplikuje 2D filtry, prahování a přidání šumu. Tím se zmenší počet prvků, podle kterých by se mohl program na bázi strojového učení orientovat. Tím se zvýší i chybovost u člověka. Stejně jako u Sketchy jsou zde obrázky rotovány jen do 4 základních úhlů. Konkrétní implementace CAPTCHA testu se však liší, neboť zde je uživateli prezentováno 12 obrázků, přičemž úkolem je označit obrázky s nesprávnou rotací (viz obrázek 1.3d). Tím se zhoršuje pravděpodobnost pro náhodné uhádnutí, protože namísto 4 možností u Sketchy máme jen dvě možnosti. Celkově dostaneme při použití 12 obrázků $(1/2)^{12} \doteq 0,0002$. Úspěšnost uživatelů při řešení testu byla stanovena na 92% a průměrná délka provádění testu na 14s.



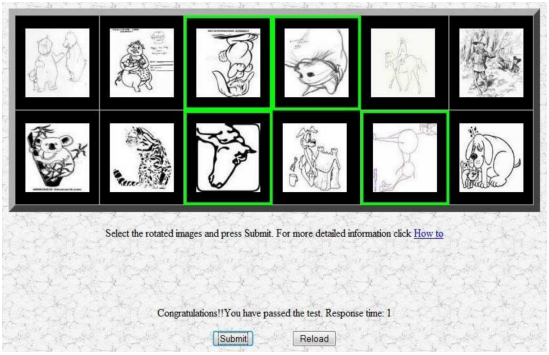
(a) What's Up CAPTCHA, převzato z [12].



(b) Multiple SEIMCHA, převzato z [18]



(c) Sketcha, převzato z [23]



(d) Orientation based image CAPTCHA, převzato z [3]

Obrázek 1.3: Ukázka různých CAPTCHA testů, založených na orientaci obrázku.

1.2.4 Sémantické a logické

U tohoto typu CAPTCHA testu předpokládáme zadávání v čistě textové podobě. Princip spočívá v pokládání otázek, k jejichž zodpovězení je třeba jistých triviálních znalostí. Nejlépe takových, které jsou pro naprostou většinu lidí jasné, ale pro počítač poměrně těžké k zodpovězení.

Jedním z příkladů může být výpočetní otázka „Kolik je 1+5?“. Tento typ otázky však v současné době není žádnou překážkou, například internetový vyhledávač Google umí na tuto otázku odpovědět výsledkem operace[4].

Triviálních otázek lze ovšem vymyslet mnohem větší množství: „Jakou barvu má pomeranč?“, „Kolik má stran trojúhelník?“ a podobně. Navíc nám to dává možnost pro řešení vyžadovat buď textovou odpověď a nebo vybrat jednu z možností. V druhém případě ale narazíme na problém s možností náhodného uhádnutí výsledku – při 3 možnostech má útočník šanci na úspěšné uhádnutí 33%. U textové odpovědi zase musíme brát v potaz více správných odpovědí.

Jako velkou výhodu zde můžeme označit automatickou podporu zrakově postižených uživatel, neboť otázka, položená v prostém textu, může být přečtena softwarem pro čtení textu. Naopak nejnepříznivější fakt proti použití této varianty představuje malá báze dat, respektive velmi špatné automatické získávání těchto triviálních otázek. Vzhledem k textovému zadání je tento typ CAPTCHA testu závislý na jazyce [7].

Jako příklad sémantických nebo logických CAPTCHA testů můžeme uvést v této práci popsané CAPTCHA testy Egglue a Text CAPTCHA 1.3.2.

1.2.5 Nevyžadující interakci

Jak již název napovídá, hlavní přednost těchto testů představuje neobtěžování uživatelů řešením CAPTCHA testů. Jak může tento způsob fungovat? Využívá jednoho známého faktu, a to relativní hlouposti automatických útoků, potažmo útočníků.

Pro ilustraci si popíšeme, jak může probíhat automatický útok na webové služby s cílem šířit nevyžádaná reklamní sdělení (spam). Útočník (spamer) si napíše skript, který bude postupně procházet jednotlivé webové stránky a hledat v nich formuláře, které by mohl využít. U nalezeného formuláře zkusí vyplnit všechny jeho pole a poté odeslat. Tento postup se poté opakuje. „Hloupost“ útočníkova skriptu většinou spočívá ve způsobu načtení webových stránek. Tyto stránky je pro automatický skript nejjednodušší načíst v jejich zdrojové podobě, tedy v HTML kódu. Nebudou spuštěny žádné uživatelské skripty (například JavaScript), ani aplikovány styly pro vykreslení stránky (CSS – Cascade Style Sheet). Tohoto a dalších způsobů lze vhodným způsobem využít pro obranu vůči možným útočníkům. Většina těchto metod však v praxi není účinná při masivnějším používání, jejich síla tkví v jednoduchosti a individuální implementaci. Následuje podrobnější soupis těchto metod.

- **Honeypot** – v překladu „hrnec medu“[5]. Tato metoda spoléhá na to, že útočníkův skript vyplní všechna pole formuláře. Toho může autor webové aplikace využít a přidat k formuláři pole, které musí zůstat nevyplněné. Před normálními uživateli jej může skrýt pomocí CSS a nebo JavaScriptu.

[4] https://www.google.com/search?q=kolik+je+1%2B5
[5] http://en.wikipedia.org/wiki/Honeypot_(computing)

Útočníkův skript, stahující formulář v čistém HTML, toto pole vidí a pokusí se jej vyplnit. Po odeslání stačí na serveru aplikace zkontrolovat, zda kontrolní pole zůstalo nevyplněno. Další možností je neskrývat toto pole před legitimními uživateli, ale označit ho vhodným způsobem, například „Toto pole nechejte nevyplněné“. Útočníkův skript však může náhodně nechávat některá pole prázdná a tím tuto ochranu prolomit [7, 2].

- **Session klíč** – chrání před útokem, který obchází formulář a jde přímo na požadavek pro zpracování formuláře. Útočník v tomto případě odesílá vlastní (většinou POST) HTTP požadavek s parametry, takže nepotřebuje ani načítat původní formulář. Na tento způsob útoku existuje jednoduchá odpověď – při vygenerování formuláře se zároveň vygeneruje a vloží do formuláře session klíč. Při zpracování odeslaného formuláře poté kontrolujeme, zda přijatý session klíč odpovídá vygenerovanému session klíči [7].
- **Kontrola obsahu** – na rozdíl od předchozích nepracuje na straně klienta, ale na straně serveru. Princip spočívá v rozpoznávání útočníka ne podle jeho chování, ale podle dat, co se pokouší odeslat. Většinou se používají filtry obsahu na bázi Bayesových filtrů v kombinaci s heuristickými filtry pro rozpoznání potencionálního spamu. Například pokud budou obsahovat data odesílané útočníkem mnoho odkazů a nebo slova jako „Viagra“, dá se s určitou pravděpodobností předpokládat, že jde o spam [7, 2].

  V praxi jsou data přijatá od uživatele na aplikačním serveru nejdříve zaslána pro kontrolu na CAPTCHA server provozující kontrolu obsahu a ten obratem vrátí procentuální hodnotu indukující, zda je zadaný obsah spam. Při označení zprávy jako spam je ještě většinou upozorněn správce webové aplikace, aby případně mohl toto rozhodnutí zamítnout.

  Na tomto principu pracují zde zmíněné CAPTCHA testy Akismet, BOTCHA a Keypic, viz 1.3.3.

1.2.6 Individuální autentizace

Individuální autentizace se uplatňuje hlavně tam, kde potřebujeme velmi vysokou úroveň ochrany, jako například u bankovních plateb. Nestačí jen zjištění, zda jste člověk, ale je také třeba vědět, který člověk konkrétně. Tento způsob ověřování se ale dá použít i u aplikací, kde není nutné rozlišení konkrétního člověka.

Nejjednodušší způsob uplatnění představuje například registrace uživatele ještě před tím, než je mu umožněno přidávat obsah. U samotné registrace je třeba kontrolovat, zda je registrovaný uživatel člověk, takže by se mohlo zdát, že jsme si nijak nepomohli. Pokud se ale použije u registrace individuální autentizace, později již není třeba autentizaci používat vůbec, neboť již máme (měli bychom mít) jistotu, že daný uživatel je člověk.

Mezi často používané metody individuální autentizace patří zadání čísla platební karty, opsání textu ze zaslané SMS a nebo ověření pomocí zaslání dopisu na adresu uživatele s ověřovacím kódem [7].

1.2.7 Centralizovaná autentizace

Centralizovaná autentizace uplatňuje princip individuální autentizace, nicméně není omezena na jediný web – je ji možno použít pro autentizaci u více webových služeb. Samotné přihlašovací a další údaje

o uživateli jsou pak uloženy na centrálním autentizačním serveru. Tento způsob ukládání autentizačních informací představuje největší výhodu a zároveň i nevýhodu tohoto systému, neboť pokud by byla uložená data kompromitována, mohl by útočník získat přístup ke všem službám, kde byly tyto údaje použity. Velmi důležitým faktorem je důvěra v autentizační autority. V neposlední řadě může tento systém podkopat prostý fakt, že pro samotnou registraci u poskytovatele centralizované autentizace je třeba provést obvyklý CAPTCHA test pro odlišení člověka od počítače.

Velmi pěkným příkladem centralizované autentizace může být OpenID[6]. OpenID řeší otázku zabránění kompromitace osobních dat tak, že není omezen na jednoho poskytovatele, ale umožňuje, aby tuto službu mohlo poskytovat více společností. Zákazník si tedy může vybrat, kterému poskytovateli nejvíce věří, a může mezi nimi libovolně přecházet. Další výhoda OpenID spočívá v ochraně autentizačních údajů při samotném přihlašování na cílovou webovou službu – tato služba nemá přístup k zadanému heslu, to jde přímo na autentizační server OpenID a zpět už putuje pouze indikace, zda jsou údaje platné.

Jak již bylo zmíněno výše, slabé místo představuje samotné vytvoření OpenID účtu, kde je třeba použít klasické ověřovací mechanismy [7, 2].

1.3 Analýza současných CAPTCHA testů

1.3.1 Rozpoznávání obrazu

Key CAPTCHA[7]

Key CAPTCHA jako mnoho dalších obrázkových CAPTCHA testů využívá hry. Konkrétně jde o hru puzzle, tedy složení celého obrázku z jednotlivých dílů. V tomto testu je po uživateli požadováno, aby přesunul dva až tři dílky do neúplného obrázku na správné místo, viz obrázek 1.4. Není zde však žádná podpora pro nevidomé uživatele.

Obrázek 1.4: Key CAPTCHA

Podle informací u pluginu Key CAPTCHA testu pro WordPress [8] je šance uhádnutí správné odpovědi 1 ku 30.000.000, tedy 0,000003%. Přesto bylo provedeno úspěšné automatické vyřešení hackerem s přezdívkou Intellect [15]. Samotný postup automatického vyřešení tohoto testu je poměrně složitý. Obrázek, který je třeba složit, je ke klientovi zaslán zakódovaný, takže autor automatického skriptu nejdříve napsal algoritmus, který provede dekódování obrázku. Poté bylo třeba „přilepit“ jednotlivé části obrázku na správné místo tak, aby předmět na obrázku byl kompletní. Proto autor nejdříve provedl detekci jednotlivých částí puzzle detekcí hran v obrázku a následnou analýzu. Poté stačilo pro každou část obrázku měnit její polohu vůči ostatním a získávat ohodnocení podle plynulosti přechodu barev na okrajích části obrázku.

[6] http://openid.net/
[7] https://www.keycaptcha.com/
[8] http://wordpress.org/extend/plugins/keycaptcha/

Tento postup může díky detekci hran fungovat pouze na starší verzi Key CAPTCHA testu, který používal dílky s rovnými hranami. Nynější varianta používá nepravidelně tvarované dílky, čímž vylučuje použití detekce rovných hran. K prolomení tohoto testu by stačilo pouze vylepšit způsob detekce jednotlivých dílků puzzle.

Asirra[9]

Málokterý CAPTCHA test poskytuje nějakou přidanou hodnotu kromě reklamy, tomuto projektu od Microsoftu se to ale podařilo. Princip řešení spočívá ve výběru obrázků, na nichž je kočka nebo pes (podle aktuálního zadání), z celkem 12 obrázků. Použití obrázků psů a koček je velmi vhodné, protože pro automatický skript je poměrně dost těžké rozlišit, zda se na obrázku nachází pes, nebo kočka, protože tyto dva druhy zvířat sdílejí některé společné rysy.

Obrázek 1.5: Asirra CAPTCHA test. Po najetí kurzoru myši nad malý náhled obrázku se zobrazí jeho větší verze.

Asirra[10] je úzce spojena s portálem PetFinder[11], který sdružuje útulky zvířat a pomáhá jim najít nové majitele pro jejich svěřence. Tento portál má v databázi více než tři miliony obrázků psů a koček, roztříděných s pomocí dobrovolníků v jednotlivých zvířecích útulcích. PetFinder poskytl tyto obrázky pro potřeby Asirry a ta na oplátku zveřejňuje u každého obrázku odkaz na adopci zobrazeného zvířete. Asira tedy nejen umožňuje rozpoznat člověka od počítače, ale také pomáhá najít domov psům a kočkám v útulcích[16]. Vzhledem k průběžné aktualizaci databáze obrázků zvířat z PetFinderu by mohla být znalostní báze Asirry potenciálně nekonečná.

Pravděpodobnost náhodného uhádnutí správného řešení Asirry je v základní variantě $1/4096$, s použitím dalších podpůrných technik[16] lze dosáhnout až úspěšnosti 1 ku $5,2$ milionu. Na testované skupině 147 lidí byla zjištěna úspěšnost řešení $83,4\%$ po prvním pokusu řešení Asirry. Po druhém řešení byla úspěšnost $97,2\%$ a po třetím již $99,5\%$. Z hlediska časové náročnosti Asirra patří k těm náročnějším, její řešení zabere uživateli zhruba 15 sekund[16].

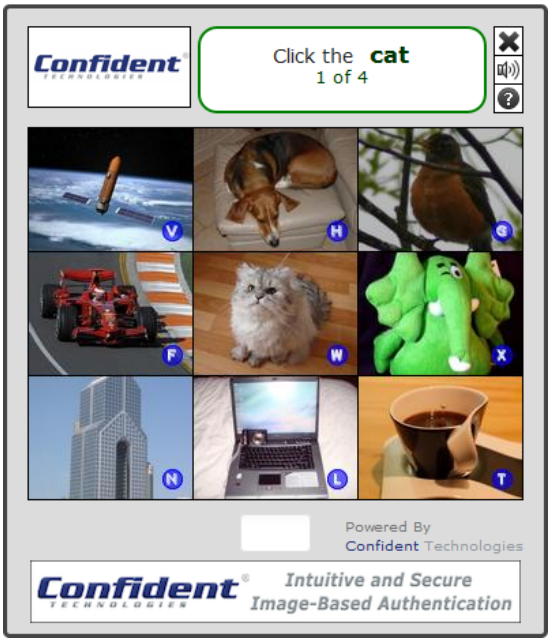
Pokusy o prolomení Assiry nebyly zaznamenány, nicméně Jarret Minkler [20] se pokusil o rozbor možného automatického útoku. Upozornil na to, že pro rozpoznání, zda je na zadaném obrázku pes či kočka lze použít Google API pro vyhledání podobných obrázků. Po nalezení množiny podobných obrázků by bylo samozřejmě ještě třeba z kontextu obrázku a webové stránky kde se nachází určit, zda jde o psa a nebo kočku, na což by šlo použít webovou službu (poskytovanou také Googlem) „Keyword tool“. Žádný reálný pokus o provedení automatického útoku na Asirru však nebyl uskutečněn. Svou roli hraje více faktorů – Asirra pomáhá zvířatům z útulku, není komerční a v neposlední řadě není příliš

[9] http://www.asirra.com/
[10] „Animal Species Image Recognition for Restricting Access“
[11] http://www.petfinder.com/

používaná.

Confident CAPTCHA™[12]

Confident CAPTCHA™ představuje zástupce rozlišovacích obrázkových CAPTCHA testů. Na rozdíl od testů stejného typu (například Asirry 1.3.1 nebo PICATCHA testu 1.3.1) nejde o označení obrázků s určitým motivem, ale o postupné označování zadaných obrázků. Celkově test sestává ze 4 kol, přičemž v každém kole je uživatel vyzván, aby označil (myší a nebo napsáním znaku přiřazeného k obrázku) obrázek, na kterém se nachází zadaný objekt. Poskytuje také poměrně dobrou audio variantu pro zrakově postižené uživatele. Nevýhodou tohoto řešení může být omezená báze dat a závislost na jazyce.



Obrázek 1.6: Ukázka Confident CAPTCHA™ testu. Uživateli jsou zadány příkazy, na které má klikat (může použít i písmena u obrázku namísto kliknutí myší). Celkově musí uživatel takto vykonat 4 příkazy.

Na rozdíl od všech ostatních testů poskytuje možnost použití místo písemného hesla pro přihlašování. Tato varianta je však zpoplatněna, což vyvažuje možnou ekonomickou ztrátu pro nepoužívání reklamy jako u některých jiných CAPTCHA testů tohoto druhu.

Andrew J. Wismer *et al.* v [30] experimentálně dokázal, že Confident CAPTCHA™ dosahuje velmi dobrých výsledků v porovnání s rozpoznáváním zdeformovaného textu, ale i v porovnání s jiným druhem obrázkového CAPTCHA testu (ESP-PIX [16]). Konkrétně bylo zjištěno, že průměrný čas pro vyřešení Confident CAPTCHA™ testu je 7,59s (směrodatná odchylka 1,43s) a míra chyby uživatelů se rovná nule.

Pravděpodobnost náhodného uhádnutí výsledku je možno vypočítat jako $(1/9)*(1/8)*(1/7)*(1/6)$, což odpovídá cca 0,033%.

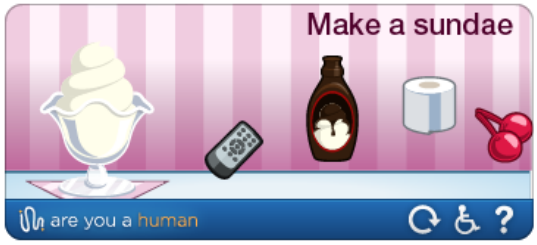
PlayThru[13]

Jak již název napovídá, PlayThru CAPTCHA od společnosti *Are you a human* se prezentuje uživateli jako hra. Jednotlivá řešení jsou velmi variabilní, ovšem sdílejí společný princip – přesunutí určitých předmětů na správné místo 1.7. Předměty nemají statickou polohu, ale náhodně se po vymezené ploše pohybují, což znesnadňuje automatické vyřešení, neboť je třeba v reálném čase sledovat pohyb těchto předmětů. Pohyb však může znesnadňovat řešení i starším lidem a lidem s motorickými problémy. Díky hernímu motivu a jednoduchosti je řešení PlayThru CAPTCHA testu poměrně zábavné a rychlé. Ačkoli je zadání testu přítomno v písemné formě, vzhledem kjednoduchosti a vizuálnímu provedení lze přijít na řešení i

[12] http://www.confidenttechnologies.com/products/confident-captcha
[13] http://areyouahuman.com/

bez znalosti daného jazyka. Pro platící zákazníky je zde možnost propagovat své produkty dosazením do herních variant.

Vypočítat množství všech možných řešení PlayThru CAPTCHA testu je poměrně obtížné, neboť při řešení se operuje ve 2D prostoru a jednotlivé varianty testu se od sebe odlišují.



Obrázek 1.7: PlayThru CAPTCHA. Řešením této varianty je přetáhnutí třešně a topinku na zmrzlinový pohár.

PlayThru CAPTCHA představuje velmi moderní řešení využívající HTML5 a CSS3. Pro zpětnou kompatibilitu se staršími internetovými prohlížeči nabízí zobrazení pomocí Adobe Flash playeru. Nechybí zde možnost řešení zvukového CAPTCHA testu pro zrakově postižené uživatele.

Byl proveden i úspěšný pokus o prolomení PlayThru CAPTCHA testu pomocí automatického skriptu [14]. Pro provedení byla použita open-source Python knihovna SimpleCV[14]. Autorovi se podařilo prolomit většinu testovaných PlayThru CAPTCHA testů do několika sekund. Podrobnější informace o implementaci nebyly zveřejněny.

Solve Media[15]

Solve Media CAPTCHA předkládá řešení problému rozpoznání člověka od počítače založené přednostně na bázi reklamy. Zahrnuje poměrně zdařilou variantu CAPTCHA testu pro zrakově postižené uživatele. Celkem nabízí 3 varianty vizuálního CAPTCHA testu.

(a) Výdělečný (b) Perspetkivní (c) Bezpečný

Obrázek 1.8: Typy Solve media CAPTCHA testů.

První z nich je „výdělečný“ test 1.8a, založený na nejznámějším principu – přepsání zobrazeného textu, u něhož je zobrazený obrázek, proklamující určitou obchodní značku. Samotný text určitým způsobem popisuje tuto obchodní značku. Předpokládaným účelem může být, aby si zákazník zafixoval zadaný text se zobrazenou značkou. Z hlediska zabezpečení vůči automatizovanému útoku však není tento test na první pohled prakticky nijak zabezpečen, neboť text, který je nutno přepsat, není nijak deformován. Po několika špatných pokusech test přejde do „bezpečné“ varianty, nicméně vzhledem k až 100% možnosti rozpoznání zobrazeného nedeformovaného textu by k této variantě nemuselo nikdy dojít.

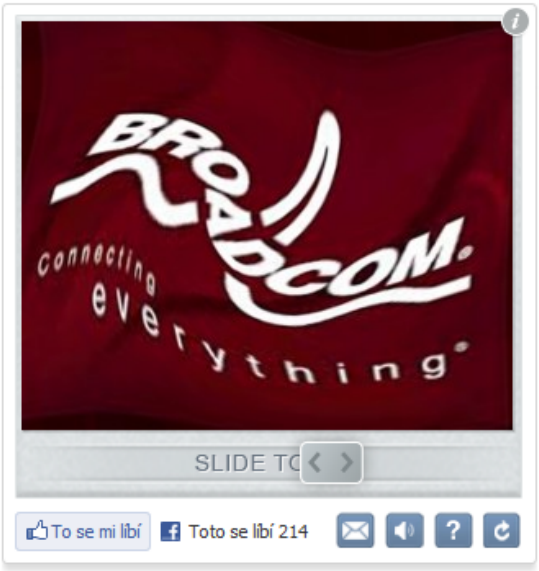
[14] http://www.simplecv.org/
[15] http://www.solvemedia.com/

Druhá „perspektivní“ varianta 1.8b také zobrazuje určitou obchodní značku, ale na rozdíl od první varianty od uživatele požaduje, aby svými slovy tuto obchodní značku popsal. Na jeden tento test tedy existuje více správných odpovědí. Stejně jako v prvním případě zde však značně pokulhává zabezpečení, protože na jakoukoli značku je správná odpověď prosté „good“ (dobrá), což jsem ověřil experimentálně 10 pokusy. Tato varianta se dá považovat za výsměch jakémukoli zabezpečení.

Poslední varianta – „bezpečná“ 1.8c konečně představuje překážku pro automatizovaný útok, neboť je založena na přepisování zdeformovaného textu. Ač je tento text velmi dobře deformovaný, dobře pracuje s barvami a přidává i další emelenty do obrázku (čáry, písmena, tvary), samotný text je vždy tvořen smysluplnými slovy, což velmi usnadňuje možný útok. Při nerozpoznání části slova pomocí OCR software může být použito slovníkových termínů pro nalezení nejpravděpodobnějšího vhodného slova.

Slide to fit captcha[16]

Slide to fit captcha od společnosti Minteye[17] představuje další z vizuálních CAPTCHA testů postavených na reklamě. Princip tohoto řešení spočívá v posunu posuvníku, který ovládá deformaci zobrazeného obrázku 1.9. Řešením je pak nalezení polohy posuvníku, ve které není obrázek nijak zdeformovaný. Těchto možných poloh je 30 a počítá se zde i s určitou tolerancí ± 2 polohy. Toto rozložení dává šanci na prosté uhádnutí řešení 5/30, tedy zhruba 16%, což je poměrně dost.



Obrázek 1.9: Slide to fit captcha test.

Větší problém spočívá ve velmi jednoduchém provedení audio varianty testu. Při zvolení audio varianty testu uživatele hlas vyzve, aby posouval posuvník s pomocí šipek doprava nebo doleva. Poté uživatel musí buď stiskem kláves ALT + X a nebo kliknutím na ikonu audio testu vyvolat hlasovou odpověď, která buď odpoví, že má uživatel pohnout posuvníkem doprava, doleva, a nebo potvrdí správnou pozici posuvníku. Hlas odpovědi není nijak zkreslený, proto lze tento test jednoduše vyřešit například s použitím neoficiálního speech2text API[18] [9].

Automatické rozpoznání správného řešení je s pomocí analýzy vizuální části Slide to fit CAPTCHA testu ještě jednodušší. Při posouvání posuvníkem dochází k výměně obrázků, na kterých je různě zdeformovaný původní obrázek. Těchto obrázků je 30, stejně jako možných poloh posuvníku. Při deformování a následné kompresi jednotlivých obrázků musí nutně dojít ke změně velikosti výsledného obrázku, a tedy by jednotlivé obrázky měly různou velikost. Tento rozdíl tvůrci Slide to fit CAPTCHA testu dorovnávají nulami, takže stačí postupně procházet obrázky, počítat nuly na konci souboru a porovnat tento počet s počtem nul na předchozím obrázku. Obrázek, pro který je rozdíl největší, je zřejmě originální obrázek. Tento způsob byl úspěšný v 10 z 10 případů [22].

[16] http://www.minteye.com/products.aspx
[17] http://www.minteye.com/
[18] https://github.com/taf2/speech2text

Pluginy pro nasazení Slide to fit CAPTCHA testu jsou dostupné pro naprostou většinu programovacích jazyků a redakčních systémů.

Slide to fit test od Minteye představil zajímavý a pro člověka velmi dobře řešitelný CAPTCHA test s reklamním využitím a podporou pro nevidomé, nicméně pro automatické řešení CAPTCHA testů nebyl žádnou velkou výzvou.

VouchSafe[19]

VouchSafe přináší odlišný způsob řešení CAPTCHA testů. Uživateli se zobrazí obrázek, v němž se nachází určitý počet předmětů různého typu, jak lze vidět na obrázku 1.10. Úkolem uživatele je nakreslit pomocí dostupného vstupního zařízení (myš, dotyková obrazovka) cestu od určitého předem zadaného objektu k jinému objektu, který má s původním objektem sémantickou vazbu. V případě obrázku 1.10 jde o nakreslení cesty od náhrdelníku s kamenem ke kravatě, protože stejně jako kravata se náhrdelník nosí na krku. Zároveň je třeba dodržet pravidlo, aby nakreslená cesta neprotínala žádný další objekt. U některých textů se také lze setkat s požadavkem kliknutí na objekt, který mezi ostatní objekty nepatří. Vzhledem k použité metodě řešení je pravděpodobnost náhodného uhádnutí správného řešení prakticky mizivá.

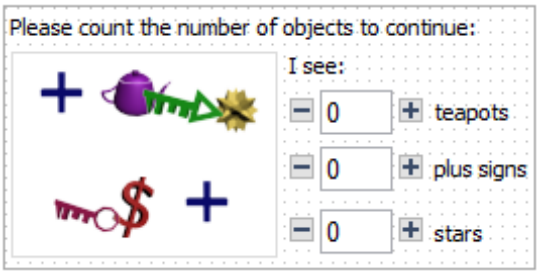
Obrázek 1.10: VouchSafe CAPTCHA. Řešením je v tomto případě nakreslení spojnice mezi přívěškem a kravatou.

Také VouchSafe využívá vizuální podobu pro reklamní účely a poskytuje možnost nahrání vlastních obrázků produktů. Výhodou je přítomnost audio varianty testu pro zrakově postižené uživatele.

U některých předmětů však může nastat problém se správným přiřazením sémanticky sdruženého objektu, neboť sémantických vazeb může být mezi objekty více. Jako u většiny obrázkových CAPTCHA testů zde zůstává problém omezené báze znalostí.

Ironclad CAPTCHA[20]

Tento CAPTCHA test používá pro generování obrázkového CAPTCHA testu 3D objekty, jak lze vidět na obrázku 1.11. Každý je náhodně umístěn, natočen, vybarven a velikostně upraven. Uživatel musí pro úspěšné vyřešení testu spočítat, kolik je na obrázku předmětů určitého zadaného typu (celkem 3 typy).



Obrázek 1.11: Ironclad CAPTCHA.

Pravděpodobnost náhodného uhádnutí správné odpovědi není nikde uvedena, nicméně experimentálně

[19] http://www.vouchsafe.com/
[20] http://www.securitystronghold.com/products/ironclad-captcha/

bylo zjištěno, že předměty z tří hledaných kategorií se v obrázku objevují v počtu 1 až 2. Pravděpodobnost náhodného uhádnutí lze jednoduše vypočítat: $(1/2)^2 = 12,5\%$, na CAPTCHA test velmi mnoho. Není zde ani audio varianta pro zrakově postižené a i přes 3D generování velmi omezené množství předmětů, které se do obrázků generují.

Pro implementaci je možno použít pouze pluginy pro PHP, ASP.NET, Dreamweaver a WordPress.

PICATCHA[21]

PICATCHA je velmi podobná CAPTCHA testu Asirra 1.3.1 – princip spočívá v označení obrázků se zadaným obsahem. Tento test nám nabízí mnohem více možností, než jen rozlišení kočky od psa 1.12. Navíc nabízí také možnost kustomizace vzhledu i bezpečnosti (přídání šumu do obrázků), případně i nahrání obrázků pro reklamní účely. Nenajdete zde ale podporu pro zrakově postižené.

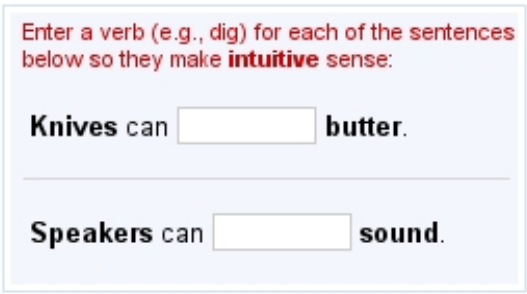
Obrázek 1.12: PICATCHA.

Experimentálně jsem zjistil, že pravděpodobnost náhodného uhádnutí PICATCHA testu při 8 zobrazených obrázcích se pohybuje od 0,3% (3 obrázky z 8 představují řešení) do 0,005% (6 obrázků z 8 představuje řešení). Při použití více obrázků se tato šance ještě více snižuje.

1.3.2 Sémantické a logické

Egglue semantic CAPTCHA[22]

Na rozdíl od předchozích CAPTCHA testů je založen na sémantických závislostech. Princip spočívá v doplnění správného slovesa do věty 1.13. Například místo teček ve větě „Nože mohou ... máslo.“ můžeme dosadit slovesa „mazat“, „roztírat“ nebo „krájet“. Je zde tedy více možných správných řešení. Pravděpodobnost prostého uhádnutí závisí na počtu slov angličtiny, je tedy téměř mizivá.



Obrázek 1.13: Egglue semantic CAPTCHA.

Toto řešení má bezesporu velkou výhodu s tom, že může být řešeno i lidmi se zrakovými vadami, protože si lze text zadání nechat přečíst softwarem pro čtení textu. Egglue CAPTCHA test však pracuje pouze s anglickým jazykem. Navíc přesto, že má více než 10.000 variant, báze znalostí je stále omezená.

Pro implementaci je dostupná pouze knihovna pro PHP a plugin do Drupalu[23].

[21] http://picatcha.com/
[22] http://www.egglue.com/
[23] http://drupal.org/project/egglue_captcha

Text CAPTCHA[24]

Text CAPTCHA řeší rozpoznání člověka od počítače položením jednoduché otázky v textové podobě, přičemž možné odpovědi jsou již v samotné otázce obsažené – například „Které číslo z čísel 13, 8 a 32 je nejmenší?“ nebo „Která barva je druhá v pořadí z prvků bílá, hlava, fialová?“. Vhodně v sobě kombinuje potřebu nejen syntaktické, ale i sémantické analýzy. Pravděpodobnost prostého uhádnutí závisí opět na počtu slov angličtiny. Generovat lze velké množství variant, typy otázek však generovat nelze. S narůstajícím počtem sémantických databází, jako například DBpedia[25] není problém automaticky přiřazovat slovům sémantickou hodnotu.

Výhoda tohoto řešení tkví v jeho jednoduchosti, neboť prostý text je přístupný také zrakově postiženým skrze software pro převádění textu na zvuk.

1.3.3 Nevyžadující interakci

BOTCHA[26]

BOTCHA („BOT Computerized Heuristic Analysis“) představuje zástupce CAPTCHA testů, nevyžadující žádnou interakci od uživatele. Spoléhá na to, že automatické programy/skripty pro spamování nejsou zrovna dvakrát chytré. Kombinuje v sobě několik metod:

- Kontrola vícenásobného odeslání – pokud je formulář vícekrát odeslán, není odesílatel považován za člověka.
- Časová brána – kontrola, zda nebyl formulář odeslán příliš rychle, což ukazuje na automatický skript.
- Honeypot – do formuláře se přidá skryté pole s určitou hodnotou, která se poté modifikuje s pomocí JavaScriptu. Počítá se s tím, že automatické skripty přistupují pouze k HTML části formuláře.
- Honeypot2 – stejné jako předchozí případ, ale změna hodnoty pole se provádí přes CSS.
- Zmatení URL – změna URL cíle formuláře s pomocí JavaScriptu.

Hlavní výhodu představuje neobtěžování uživatel, nicméně tento CAPTCHA test nemůže obstát proti automatickému skriptu napsanému na míru.

Plugin je dostupný pouze pro Drupal.

Keypic[27]

Ač Keypic od uživatele nevyžaduje interakci, není zcela neviděn, neboť u formuláře vždy zobrazí obrázek. Tento obrázek nemá po technické stránce žádný význam, nicméně je příležitostí pro reklamu. Veškeré rozpoznání člověka se provádí na Keypic serveru, který vyhodnotí získané údaje o klientovi a zašle serveru aplikace procentuální zhodnocení, zda je klient člověk.

[24] http://textcaptcha.com/
[25] http://dbpedia.org/
[26] http://drupal.org/project/botcha
[27] http://keypic.com/

Implementace je dostupná pro jazyky PHP, ASP, ASP.NET a pro redakční systémy WordPress a Drupal.

Akismet[28]

Akismet (doslova znamená „Pravděpodobně nejlepší způsob ochrany webu proti spamu“) je velmi populární webová služba pro boj se spamem, nevyžadující uživatelovu interakci. Na rozdíl od ostatních CAPTCHA testů nerozpoznává uživatele od počítače, ale hodnotí samotný obsah zpráv, které uživatelé odesílají. Kontrola probíhá na serveru aplikace, který zašle veškerý uživatelem odeslaný obsah serveru Akismetu a ten poté odpoví, zda je daný obsah spam.

V současné době je tato služba dostupná pouze v placené variantě s možností implementace do všech nejrozšířenějších programovacích jazyků a redakčních systémů.

1.4 Vyhodnocení

Tato sekce obsahuje shrnutí zjištěných poznatků o CAPTCHA testech, popsaných v sekci 1.3. Jsou zde zachyceny jak vlastnosti, které již byly v textu zmíněny, tak i jejich ostatní vlastnosti. V této sekci také najdete srovnávací tabulky pro vizuální (tabulka 1.1), sémantické a logické CAPTCHA testy (tabulka 1.2) a testy nevyžadující interakci (tabulka 1.3).

Podpora zrakově postižených je samozřejmá v případě CAPTCHA testů nevyžadujících interakci, nicméně i všechny testované sémantické a logické CAPTCHA testy tento požadavek splňují. U vizuálních testů lze tento požadavek vzhledem k použití obrazového kanálu splnit prakticky jen přidáním další varianty CAPTCHA testu, která nebude založena na vizuálním řešení. Vesměs všechny CAPTCHA testy s podporou audio varianty si zvolily opisování textu z vygenerovaného zkresleného zvuku, až na Slide to fit CAPTCHA test 1.3.1, který umožňuje podat zvukovou odezvu při procesu řešení (to je také jeho velkou slabinou).

Co se týká technologií, jde zde především o technologie použité v klientské části CAPTCHA testu, neboť nelze zjistit konkrétní implementační technologii serverové části (pokud to nespecifikuje přímo provozovatel). Akismetu a Text CAPTCHA test s klientskou aplikací komunikují pouze na straně serveru, proto u nich není implementační technologie uvedena. Ostatní CAPTCHA testy používají pro provoz na straně klienta minimálně CSS a HTML, kromě Egglue také JavaScript (JS). Key CAPTCHA, PlayThru a Solve media testy používají pokročilejší technologie HTML5 (canvas) a CSS3 společně s technologií Flash, která zajišťuje kompatibilitu i ve starších verzích internetových prohlížečů, které tyto nové technologie nepodporují. Všechny tyto testy by tedy měly fungovat v současných i starších ještě používaných internetových prohlížečích.

Až na Akismet všechny CAPTCHA testy nabízejí použití bez poplatku. Více než polovina zde popsaných CAPTCHA testů má také komerční variantu, většinou na bázi reklamy. Vizuální testy se pro tento způsob komerce výborně hodí, dvě třetiny zde uvedených vizuálních testů tohoto využívají. Keypic, ač by se díky svému principu fungování nemusel uživateli prezentovat vůbec, možnost vizuální reklamy poskytuje.

[28] http://akismet.com/

Tabulka 1.1: Shrnutí poznatků o vizuálních CAPTCHA testech.

	Asirra	Confident CAPTCHA™	Ironclad CAPTCHA	Key CAPTCHA	PICATCHA	PlayThru	Slide to fit captcha	Solve media	VouchSafe
Podpora zrakově postižených	ne	ano	ne	ne	ne	ano	ano	ano	ano
Technologie	JS, CSS, HTML	JS, CSS, HTML	JS, CSS, HTML	JS, CSS3, HTML5, Flash	JS, CSS, HTML	JS, CSS3, HTML5, Flash	JS, CSS, HTML	JS, CSS3, HTML5, Flash	JS, CSS, HTML
Komerční	ne	ano	ne	ano	ano	ano	ano	ano	ne
Varianta bez poplatku	ano	ano	ano	ano	ano	ano	ano	ano	ano
Umožňuje reklamu	ne	ne	ne	ano	ano	ano	ano	ano	ano
Podpora mobilních zařízení	ano	ano	ano	ano	ano	ano	ano	ano	ano
Podpora více jazyků	ne	ano	ano	ano	ano	ano	ne	ano	ne
Závislost na jazyku	ano	ano	ano	ne	ano	částečně	částečně	ano	ano
Omezená báze znalostí	ano	ano	ano	ano	ano	ano	ano	ano/ano/ne	ano
API	ano	ano	ano	ano	ano	ano	ano	ano	ano
SSL	ne	ano	ne	ano	ne	ano	ne	ano	ne
Dokumentace	ano	ano	ano	ano	ano	ano	ano	ano	ano
Pluginy	ano	ano	ano	ano	ano	ano	ano	ano	ano
Přidaná hodnota	ano, adopce zvířat	ne	ne	ne	ano, reklama	ano, reklama	ano, reklama	ano, reklama	ano, reklama
Pravděpodobnost uhádnutí	2×10^{-10}	$3,3 \times 10^{-4}$	$0,037$	3.3×10^{-8}	3×10^{-3} až $1,2 \times 10^{-4}$	*	$0,16$	**	*

* Pravděpodobnost uhádnutí PlayThru a VouchSafe CAPTCHA testu nelze jednoduše spočítat.
** Pravděpodobnost uhádnutí těchto CAPTCHA testů je závislá na počtu možných slov a jejich kombinací v daném jazyce.

Tabulka 1.2: Shrnutí poznatků o sémantických a logických CAPTCHA testech.

	Egglue Semantic CAPTCHA	Text CAPTCHA
Podpora zrakově postižených	ano	ano
Technologie	CSS, HTML	-
Komerční	ne	ne
Varianta bez poplatku	ano	ano
Umožňuje reklamu	ne	ne
Podpora mobilních zařízení	ano	ano
Podpora více jazyků	ne	ne
Závislost na jazyku	ano	ano
Omezená báze znalostí	ano	ano
API	ano	ano
SSL	ne	ne
Dokumentace	ne	ano
Pluginy	ano	ano
Přidaná hodnota	ne	ne
Pravděpodobnost uhádnutí	*	*

* Pravděpodobnost uhádnutí těchto CAPTCHA testů je závislá na počtu možných slov a jejich kombinací v daném jazyce.

U CAPTCHA testů, které nejsou závislé na jazyce, je podpora více jazyků samozřejmá, nicméně u ostatních tomu tak není. Sémantické a logické CAPTCHA testy jsou z principu jejich fungování na jazyku závislé a je potřeba překládat jednotlivé varianty do cílového jazyka uživatel. Vizuální texty se mohou obejít bez použití jazykových prostředků, pokud bude grafická podoba testu dostatečně zřejmá už při prvním pohledu. Tento případ dobře ilustruje Key CAPTCHA 1.3.1. Protipříkladem budiž Asirra 1.3.1, u které potřebujeme pro úspěšné vyřešení znát, zda hledáme kočku nebo psa.

Pro zaručení dostatečné bezpečnosti je třeba skrýt logiku CAPTCHA testu na serverové straně. Tím ovšem vzniká požadavek na přenos dat mezi klientem a serverem a také na standardizaci této komunikace, což řeší použití API (Application Programming Interface). Naprostá většina zde hodnocených CAPTCHA testů používá pro svůj provoz API, vyjma BOTCHA CAPTCHA testu 1.3.3, protože ten funguje převážně na straně klienta. Až na PICATCHA CAPTCHA test 1.3.1 a Keypic 1.3.3 mají všechny komerční testy podporu zabezpečeného přenosu dat pomocí HTTPS protokolu (Hypertext Transfer Protocol Secure), z nekomerčních žádný.

Základem každého CAPTCHA testu je dobrá dokumentace a množství pluginů pro programovací jazyky a redakční systémy, protože provozovatelé webových služeb musí být schopni implementovat rozhraní pro komunikaci se serverovou stranou CAPTCHA testů. Komerční testy samozřejmě poskytují pluginů nejvíce, u nekomerčních nalezneme kromě Egglue 1.3.2 CAPTCHA testu minimálně dokumentaci a plugin pro alespoň jeden programovací jazyk.

Přidanou hodnotou může být již zmíněná možnost zobrazování reklamy třetích stran. Jedině Asirra dokázala využít problém rozpoznávání člověka od počítače smysluplněji a pomáhá najít zvířatům z útulků

nový domov.

Tabulka 1.3: Shrnutí poznatků CAPTCHA testech nevyžadující interakci.

	Akismet	BOTCHA	Keypic
Podpora zrakově postižených	ano	ano	ano
Technologie	-	JS, CSS, HTML	JS, CSS, HTML
Komerční	ano	ne	ano
Varianta bez poplatku	ne	ano	ano
Umožňuje reklamu	ne	ne	ano
Podpora mobilních zařízení	ano	ano	ano
Podpora více jazyků	ano	ano	ano
Závislost na jazyku	ne	ne	ne
API	ano	ne	ano
SSL	ano	ne	ne
Dokumentace	ano	ano	ano
Pluginy	ano	ano	ano
Přidaná hodnota	ne	ne	ano, reklama

Kapitola 2

Analýza

V současné době se nejvíce používají CAPTCHA testy založené na rozpoznávání zdeformovaného textu. Už v roce 2012 však Američané konzumovali více internetového obsahu přes mobilní zařízení (chytré telefony, tablety), než přes televize a počítače[1]. Čím dál tím více vzniká potřeba přizpůsobovat zobrazení a hlavně ovládání internetových aplikací pro mobilní zařízení. Pro CAPTCHA testy jako nedílné součásti internetových aplikací zde vzniká prostor pro zavedení přirozenějšího a zábavnějšího ovládání, než jak tomu bylo u počítačů. Především jde o využití možností dotykových obrazovek, které jsou pro lidi daleko přirozenější, než ovládání počítače pomocí myši. Základní vlastnosti CAPTCHA testu musí zůstat nadále platné – musí být lehce řešitelný pro člověka (s možnou poruchou vnímání) a zároveň dostatečně bezpečný, aby ho nebylo možno automaticky prolomit s pomocí počítače.

Na základě rešerše CAPTCHA testů, provedené v sekci 1, jsem se rozhodl detailněji analyzovat dvě kandidátní řešení s pracovními názvy RoCAPTCHA (sekce 2.2) a SegCAPTCHA (sekce 2.3).

2.1 Obecné principy

2.1.1 Bezpečnost

Existují techniky, které umožňují omezit možnosti útočníka, ať už zvolí jakoukoli výše zmíněnou metodu útoku. V následující sekci bude jedna z nich popsána podrobněji.

Token Bucket

Token bucket (v překladu „vědro s tokeny“) je technika obvykle používaná v počítačových sítích pro omezování množství přenesených dat za jednotku času. U CAPTCHA testu byla tato metoda použita poprvé v testu Asirra [16], kde tímto způsobem umožňuje odolávat vůči útokům hrubou silou. Původní předpoklad počítá s tím, že počítačový bot může být rozpoznán podle malého počtu IP adres a velkého počtu nesprávných řešení. Při každé žádosti o vygenerování nového testu je pro IP adresu, ze které jde požadavek, vytvořeno vědro (pokud již není vytvořeno) a do něj vygenerováno *TB-Max* tokenů. Pokaždé, když klient odešle jakoukoli odpověď, jeden token je z vědra odebrán až do minima 0. Pokaždé, když klient úspěšně vyřeší test, je do jeho vědra přidáno *TB-Refill* tokenů až do maxima *TB-Max*. Po určitém čase (například 24 hodin) může být počet tokenů ve vědru znovu inicializován na *TB-Max*. Pokud uživatel odešle odpověď a jeho vědro je prázdné, bude jeho odpověď považována za nesprávnou i kdyby odpověděl správně [16].

[1] http://www.fiercemobilecontent.com/story/americans-now-consume-more-media-mobile-devices-tv-pcs/2012-08-14

Automatický útok hrubou silou by pravděpodobně vyprázdnil vědro s tokeny velmi brzy, neboť množství jeho nesprávných odpovědí převažuje nad množstvím správných. Při *TB-Refill* = 3 by musel správně vyřešit dva testy, než by mohl dostat další token za správnou odpověď.

Matematicky je možno popsat výpočet výsledné pravděpodobnosti náhodného uhádnutí řešení jako [16]

$$p_{res} = p \times (1 - (1 - p)^{TB-Refill})$$

Princip této techniky je založen na předpokladu, že každý uživatel má unikátní IP adresu. V praxi tento předpoklad však nefunguje, protože mnoho uživatelů často sdílí jednu IP adresu. Dá se předpokládat, že automatický útok hrubou silou by probíhal mnohem rychleji, než je uživatel schopen řešit testy, což by vedlo k rychlému vyprázdnění vědra s tokeny a tím zabránění uživateli v úspěšném používání požadované služby [16].

Řešení tohoto problému předpokládá vytvoření dalšího vědra, ovšem ne pro IP adresu, ale pro vytvoření sezení (*session*). Jsou přidána následující pravidla [16]:

- Při vytvoření nového sezení se pro něj vytvoří samostatné vědro a inicializuje se na hodnotu aktuálního počtu tokenů ve vědru pro IP adresu. Také se odečte jeden token z vědra pro IP adresu.
- Po odeslání jakékoli odpovědi se odečte jeden token jak z vědra pro IP adresu, tak z vědra pro sezení.
- Při správné odpovědi je přidáno TB-Refill tokenů jak do vědra pro IP adresu, tak z vědra pro sezení.
- Uživateli je potvrzeno správné řešení jen v případě, že vědro pro jeho sezení není prázdné.

To umožňuje uživatelům používat požadovanou službu pokud vyřeší jeden extra test navíc správně, i když sdílí IP adresu s útočníkem, který provádí automatický útok. Použití pouze vědra pro sezení by nebylo vhodné, neboť útočník by mohl vytvořit před každým pokusem nové sezení. Legitimní uživatelé však vytvoří pouze jediné sezení a budou jej používat delší dobu, čímž je můžeme odlišit od možných útočníků. Pokud by uživatel začal sezení na stejné IP jako útočí útočník, bude mu vytvořeno vědro pro jeho sezení, inicializováno na 0 tokenů. Při odeslání správné odpovědi mu však bude navýšen počet tokenů v jeho vědru pro sezení o *TB-Refill*. Tyto tokeny nemůžou být zrušeny útočníkem, neboť ten má vlastní vědro pro sezení. Při druhé správné odpovědi je tedy uživateli umožněno používat danou službu, kterou CAPTCHA test ochraňuje[16].

Veřejný a privátní klíč

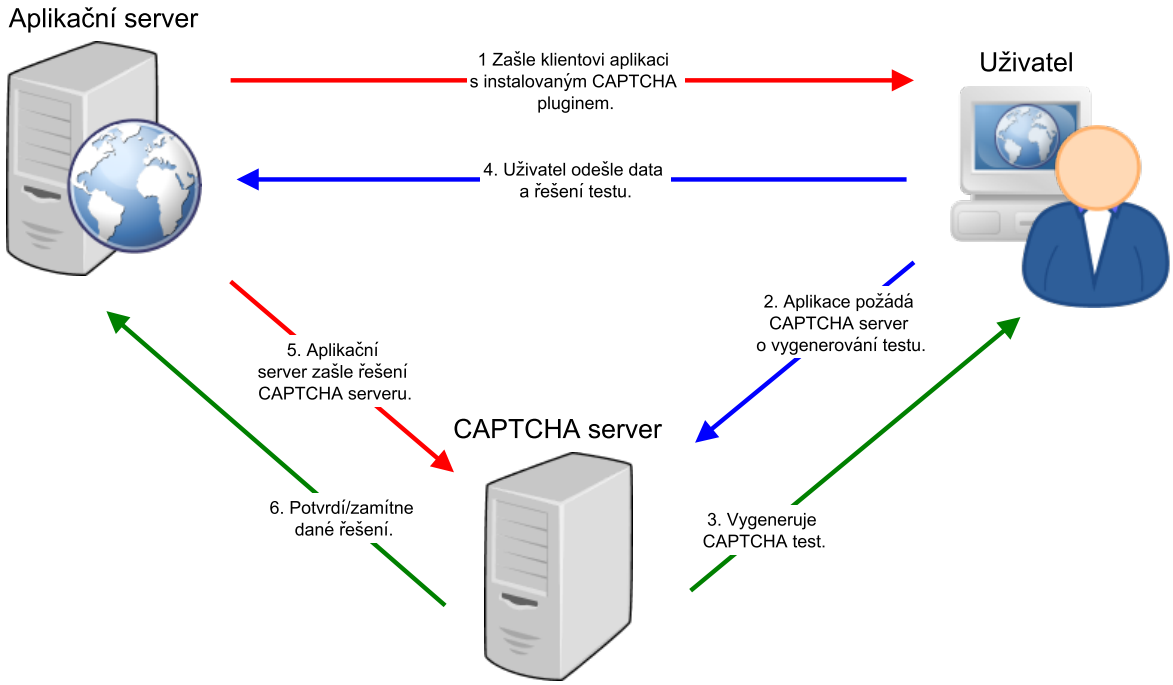
Při poskytování služeb CAPTCHA testu vzniká potřeba identifikovat webové aplikace a klienty, využívající tuto službu. Identifikace a sledování množství přístupů skrze jednotlivé webové aplikace nám umožní detekovat a zablokovat možný útok hrubou silou, nehledě na možnost zpoplatnění požadavků nad určitou nastavenou hranici.

Jedna z možností, jak toto provést, spočívá v zavedení unikátního veřejného a privátního klíče pro každou webovou aplikaci, používající náš CAPTCHA test. Veřejný klíč je použit pro identifikaci klientů, neboť požadavky na vygenerování CAPTCHA testu přicházejí přímo z jejich koncových zařízení (viz sekce 2.1.2). Jak již název napovídá, obsah tohoto klíče je veřejně známý. Privátní klíč slouží ke

komunikaci mezi serverem webové aplikace a CAPTCHA serverem. Server webové aplikace prokazuje svou totožnost CAPTCHA serveru při ověřování, zda klient správně vyřešil CAPTCHA test. Tento klíč musí zůstat v tajnosti, protože v opačném případě by mohl být využit k útoku hrubou silou, který by vyústil v zablokování tohoto privátního klíče a tím nedostupnost ověření CAPTCHA testu v původní webové aplikaci. Identifikace serveru webové aplikace například jen podle IP adresy by v tomto případě nestačila, protože daná webová aplikace se může v průběhu času přesouvat na jiný server, kde by měla jinou IP adresu.

2.1.2 API

API[2] představuje obecné rozhraní pro komunikaci. Samotná komunikace probíhá na daném protokolu, v našem případě na protokolu HTTP. Komunikace probíhá mezi 3 stranami - aplikačním serverem, uživatelovým koncovým zařízením a CAPTCHA serverem. Životní cyklus CAPTCHA testu můžete vidět na obrázku 2.1. Konkrétní implementace API musí z tohoto životního cyklu vycházet.



Obrázek 2.1: Životní cyklus CAPTCHA testu.

2.2 RoCAPTCHA

Název RoCAPTCHA představuje zkratku z ***Ro**tational **CAPTCHA***. V podstatě jde o kombinaci *What's Up* CAPTCHA testu a testu *Asirra*. Z prvního byl převzat princip řešení, což je otáčení obrázků do vzpřímené polohy (odtud pochází název). Hlavní přednost tohoto řešení spočívá v možnosti velmi pohodlného a rychlého řešení s pomocí prstů na dotykových zařízeních. Z výzkumu [12] vyplývá, že uživatelé preferují tento způsob řešení CAPTCHA testu před rozpoznáváním deformovaného textu. Další velkou výhodu

[2] Application Programming Interface

představuje nezávislost na jazyce. Také je poměrně náročné automatické nalezení vzpřímené polohy počítačem při správném vytřídění snadno rozpoznatelných obrázků (viz 2.2.1).

Z Asirry je převzato využití fotografií zvířat jako zdrojových obrázků, což přináší několik výhod. Především jsou tyto fotografie často pořizovány ve vnitřních prostorách, čímž odpadají pro počítač lehce rozpoznatelné prvky jako obloha, tráva nebo písek. Úhel zachycení fotografie vzhledem k rovině horizontu obvykle bývá kolem -45°, protože lidé jsou větší než focená zvířata a fotografují přibližně z výšky svého obličeje. Tím se znesnadní automatické rozpoznání vzpřímené polohy, protože na fotografii není zachycen horizont. V neposlední řadě zde najdeme velmi málo lidských obličejů, předmětem focení jsou především zvířata. Posledním a možná jedním z nejdůležitějších faktorů je fakt, že lidé obecně mají kladný vztah ke zvířatům, což může přispívat k větší uživatelské spokojenosti s řešením RoCAPTCHA testu.

Ukázku tohoto řešení lze vidět na obrázku 2.2.

Obrázek 2.2: Ukázka dotykového ovládání RoCAPTCHA testu. Cílem je otočit obrázek do vzpřímené polohy.

2.2.1 Vhodnost otáčení obrázků

V této sekci se blíže podíváme na metodu otáčení obrázků do vzpřímené polohy z hlediska použití pro CAPTCHA test.

Analýza principu

Správnou polohu obrázku lze obecně zjistit pomocí nízkoúrovňové a vysokoúrovňové analýzy atributů obrázku. Při nízkoúrovňové analýze se orientujeme například pomocí barvy, textury, nebo hran. U vysokoúrovňové analýzy je třeba brát v potaz sémantický význam předmětů na obrázku – například obloha, tráva, auto, člověk a podobně [17].

Velmi důležité z hlediska automatického rozpoznání polohy je vertikální úhel roviny, pod kterým je fotografie pořízena vzhledem k rovině horizontu. Pokud totiž pořizujeme fotografii pod úhlem -90° (směrem k zemi) a nebo 90° (směrem k obloze), není možné správnou polohu rozlišit, neboť vzpřímená poloha se určuje vzhledem k rovině horizontu (příklad fotografie pod úhlem -90° na obrázku 2.3a). Pro rozpoznání polohy je obecně nejlepší úhel kolem 0° (viz 2.3c) vůči rovině horizontu a naprostá většina fotografií z venkovního prostředí je takto focena. Při zvětšujícím se úhlu focení vůči rovině horizontu se rozpoznání správné polohy znesnadňuje, neboť se již nelze spoléhat na pozici oblohy, trávy a nebo vodních ploch (fotografie 2.3b).

[3] http://www.flickr.com/photos/martine266/

(a) -90° (autor: Ann Rafalko[3])

(b) -45° (zdroj: PetFinder.com)

(c) 0° (autor: Ann Rafalko[3])

Obrázek 2.3: Ukázka fotografií, focených pod různým úhlem vůči rovině horizontu.

Rozpoznávání vzpřímené polohy lidmi

V otáčení obrázků do vzpřímené polohy mají lidé úspěšnost větší než 95%, pokud bereme v potaz obrázky o velikosti 256 × 384 pixelů a větší [17]. Toho dosahují získáváním určitých vodítek z obrázku, které lze rozčlenit do několika kategorií [28, 12]:

1. **Objekty s dobře rozpoznatelnou orientací**: lidé, obličeje, stromy, zvířata a text.
2. **Objekty s obvykle neměnnou polohou**: obloha, tráva, vodní plochy, písek.
3. **Nízkoúrovňové prvky**: světlo, textura, symetrie, okraje. Světlejší prvky jsou většinou nahoře, tmavší dole.

První dvě kategorie představují vysokoúrovňové prvky, neboť pro rozpoznání je třeba znát jejich sémantický význam.

Automatické rozpoznávání vzpřímené polohy

Na poli automatického rozpoznávání vzpřímené polohy fotografií již byly učiněny velké pokroky, hlavně díky potřebě usnadnit lidem práci s otáčením fotografií, pořízených s vertikálním sklonem fotoaparátu. Dnešní fotoaparáty mohou mít zabudován detektor polohy, takže se poloha snímku uloží do jeho EXIF dat. U starších fotografií toto není možné a zde nastupuje automatické rozpoznávání polohy. V tomto případě jde především o rozpoznání 4 základních poloh (0°, 90°, 180°, 270°). Tato funkce bývá zabudována do mnoha programů pro úpravu fotogalerií. A. Vailaya, H. Zhang a A, Jain ve své práci [26] experimentálně dokázali, že pro automatické otáčení obrázků do vzpřímené polohy lze použít Bayesův klasifikátor s úspěšností až 87,8% pro běžné venkovní fotografie.

Automatické rozpoznání polohy fotografie počítačem spoléhá především na nízkoúrovňové prvky obrazu, jako světlo, textura nebo okraje. Zároveň však může používat i vysokoúrovňové prvky, jako lidské obličeje. Tyto a další prvky budou podrobněji rozebrány dále.

Obloha

Umístění oblohy na fotografiích je v drtivé většině nahoře, čehož lze využít pro zjištění orientace fotografie. Tento předpoklad však sám o sobě nestačí, je třeba brát v potaz další kritéria – oblasti oblohy musí být propojené, musí procházet okrajem snímku a nesmí být obklopeny oblastmi jiného typu [28].

Světlo

Vzhledem k tomu, že světlo z většiny přírodních světelných zdrojů přichází shora, mohla by se nabízet hypotéza, že světlejší oblasti fotografií se nacházejí nahoře a tmavší dole [28]. U fotografií ve venkovním prostředí u mnoha případů opravdu platí, nicméně u fotografií z vnitřních prostor to nemusí platit, neboť zde musíme brát v potaz umělé zdroje světla.

Textura

Zvláště u venkovních snímků si můžeme všimnout rozdílu v celkové složitosti obrazu v dolní části oproti horní části, což je způsobeno především gravitací (objekty se drží při zemi). I tento fakt lze využít při nízkoúrovňové detekci orientace fotografie [28].

Symetrie

Mnoho přírodních i člověkem vytvořených objektů je symetrických. Některé mohou mít více os symetrie, nicméně velké množství z nich má vertikální osu symetrie (obličeje, lidé, domy, stromy). Analýzou symetrie v obraze můžeme získat tuto osu a tím dvě možné vzpřímené polohy. Proto je nutno tuto metodu kombinovat s jinými, pokud chceme dostat pouze jednu vzpřímenou polohu [28].

Obličej

Lidský obličej má velmi dobře detekovatelnou polohu, neboť ze vzájemné pozice očí můžeme najít osu obličeje a z pozice nosu a úst odvodit finální orientaci. Obličej je také dobře detekovatelný a existuje mnoho metod jeho detekce [28].

Hrany

Ve městech, vnitřních prostorách ale i venku se můžeme setkat s objekty, jejichž hrany jsou buď rovnoběžné s horizontem (obzor, stůl, chodník), a nebo rovnoběžné se svislou osou (roh domu, tyč dopravní značky, nohy stolu). Jednoduchou detekcí hran ve fotografii lze tyto prvky získat a odvodit z nich alespoň přibližnou orientaci fotografie (4 možné orientace).

Při nižší variabilitě fotografií by bylo možné upřesnit orientaci až na 2 možnosti (pokud by se na fotografii vyskytovalo více horizontálních hran než svislých). Při kombinaci s výše zmíněnými metodami může tato metoda mít slibné výsledky.

2.2.2 Bezpečnost

V následujících sekcích bude rozebrána bezpečnostní problematika RoCAPTCHA testu. Především bude zhodnoceno použití různých typů útoků, zmiňovaných v sekci 1.1.1 a také další možnosti ochrany.

Náhodné hádání

Jak bylo zmíněno v sekci 1.1, automatický útok s použitím náhodného hádání by obecně neměl být úspěšný ve více než 0,01% případů. U RoCAPTCHA testu předpokládáme u jednoho obrázku 360°

volnosti. Dále je třeba stanovit, jak velká bude tolerance odchylky od správné polohy pro úspěšné řešení.

Pokud bude například tolerance 10° na každou stranu (obrázek ve vzpřímené pozici má otočení 0°), dostaneme celkovou toleranci 20°. Jednoduchým výpočtem poté dostaneme, že úspěšné řešení můžeme uhádnout v $(20/360) * 100 \doteq 5,5\%$ případů. To ovšem nestačí, proto je třeba buď zmenšit toleranci, a nebo zvětšit počet obrázků, které bude uživatel v rámci jednoho testu řešit. I kdybychom zmenšili toleranci na 0°, nebude pravděpodobnost náhodného uhádnutí dost malá. Zde se vyplatí využít techniky Token Bucket (podrobněji rozebraná v sekci 2.1.1). S ní můžeme při stejné toleranci dostat pravděpodobnost náhodného uhádnutí jako

$$P = p \times \left(1 - (1-p)^{TB-Refill}\right) = 0,0088$$

kde $p = 20/360$ a *TB-Refill* = 3. S touto technikou bychom dosáhli dostatečně bezpečné hladiny pravděpodobnosti náhodného uhádnutí (při stejné toleranci 10° na každou stranu) již se 2 obrázky, $P = 0,000028$. Přesné stanovení tolerance odchylky úhlu od správného řešení by mělo být provedeno experimentálně, aby se našla hodnota s vysokou úspěšností u uživatelů a nízkou pravděpodobností náhodného uhádnutí.

Přímé přiřazování

Zabezpečení CAPTCHA testu by nemělo záviset na nedostupnosti originálních dat a algoritmů pro generování, tyto prvky by naopak podle pravidla otevřenosti 1.1 měly být veřejné. Tím ale v případě RoCAPTCHA testu nastává problém, neboť útočník by se mohl pokusit zrekonstruovat databázi obrázků ze stejných zdrojů a poté zjistit u RoCAPTCHA testu správné řešení pomocí nalezení správného původního obrázku. Pokud by zdrojová databáze obsahovala 1 000 obrázků (každý obrázek o kruhovém rozměru s poloměrem 100px), musel by útočník provést $1000 * 360 * 100^2 * \pi \doteq 11$ miliard operací, což je při dnešním výpočetním výkonu velmi dobře proveditelné. Pokud bychom ale použili databázi s milióny obrázků, počet operací bude v miliardách a to už bude časově i finančně náročnější. Tento útok je tedy v praxi možný, nicméně nemusí být pro útočníka ekonomicky přijatelný.

U každého RoCAPTCHA je zobrazen odkaz na originální obrázek a informace o autorovi kvůli dodržení vlastnických práv autora. Mohlo by se zdát, že tato funkce umožňuje útočníkovi jednoduše zjistit správnou polohu řešeného obrázku. Bez dalších bezpečnostních opatření by to byla pravda, nabízí se nicméně několik řešení. CAPTCHA test Asirra [16] tento problém řeší tak, že při žádosti o přesměrování na zdrojový obrázek invaliduje celý právě řešený test. Dalším řešením by bylo nejdříve počkat, až uživatel zadaný test vyřeší a až poté mu dovolit zobrazení zdrojového obrázku. Předchozí způsoby je navíc možno kombinovat s omezením přesměrování na zdrojový obrázek na určitý počet za den.

Jednu z možností boje proti tomuto typu útoku představuje neustálá aktualizace zdrojových dat a vyřazování starších, několikrát použitých obrázků. Tento princip byl úspěšně aplikován například v CAPTCHA testu Asirra [16].

Strojové učení

Strojové učení představuje nejvíce sofistikovaný typ útoku. Princip spočívá v rozpoznání vzpřímené polohy obrázku analyzováním jeho nízko a vysokoúrovňových prvků, jak bylo popsáno v sekci 2.2.1. Proto je důležité ještě před zařazením obrázku do databáze provést jeho klasifikaci a vyloučit ty obrázky, u kterých by mohla být snadno automaticky rozpoznána vzpřímená poloha. Především jde o text, obličeje, lidi, oblohu, trávu a písek [12, 28]. Kromě zmíněných prvků také usnadňují automatické rozpoznání hrany v obrázku, které korelují se vzpřímenou polohou (například horizont), ty je třeba také vytřídit.

Při zvolení správného zdroje obrázků můžeme některé ze zmíněných filtrů vynechat. Například u obrázků zvířat ze serveru PetFinder.com chybí některé nízkoúrovňové prvky, podle kterých by bylo možno provést automatické rozpoznání, protože jsou foceny především ve vnitřních prostorách. Tím odpadá vliv oblohy, světla a textury, ve většině případů i obličeje (foceny jsou především samotná zvířata). I na zvířecí obličeje, ač jsou více variabilnější, než lidské, je však možno natrénovat neuronovou síť a s její pomocí rozpoznat vzpřímenou polohu obrázku. Experimentální ověření by bylo již nad rámec této práce. V případě dalšího rozvoje tohoto řešení by bylo třeba vyřadit i obrázky s obličeji zvířat v polohách korelujících s orientací obrázku.

Lidské zdroje

Proti použití lidských zdrojů při řešení CAPTCHA testů existuje jen malá možnost obrany. Asi jedinou možnost představuje omezení počtu možných řešení CAPTCHA testu z určité IP adresy za určitý čas, nicméně s dostatečně velkým botnetem a rozsahy IP by i přesto nebylo obtížné tuto překážku překonat. Nejdůležitějším hlediskem je ekonomická návratnost tohoto útoku.

2.2.3 Získávání obrázků

Problematiku získávání zdrojových obrázků pro CAPTCHA test velmi dobře vyřešila Asirra [16]. Využívá vzájemně prospěšného spojení s portálem PetFinder.com, který sdružuje Americké zvířecí útulky. Za poskytování velkého množství obrázků psů a koček Asirra na oplátku u každého zobrazeného obrázku zveřejňuje odkaz „Adoptuj mě“, směřující na stránku věnovanou adopci zobrazovaného zvířete. Jak již bylo zmíněno v sekci 2.2.2, obrázky z PetFinderu jsou pro použití v RoCAPTCHA testu obzvláště vhodné, neboť neobsahují některé počítačem snadno rozpoznatelné prvky. Navíc umožňuje získávat údaje o zvířatech přes své API[4].

Existuje samozřejmě mnohem více webových služeb, které by bylo možno využít pro získávání zdrojových obrázků. V podstatě vyhovuje každá služba, která má veřejně dostupné rozhraní a přibývají zde rychle data. Výhodou je veřejně dostupné API, což významným způsobem usnadňuje proces importování nových obrázků. V ideálním případě by byla nejlepší vzájemná dohoda určitým způsobem prospěšná oběma stranám, například za poskytnutí reklamy. I v tomto případě je však třeba dodržovat nastavené limity množství dotazů a přenesených dat. V neposlední řadě je třeba ctít vlastnická práva autorů a webových služeb.

[4] http://www.petfinder.com/developers/api-docs

Jako příklad webových služeb, ze kterých může RoCAPTCHA čerpat zdrojové obrázky, můžeme zmínit *Flickr.com*[5] – uchovává skoro 240 milionů fotografií, zvětšující se každým dnem, a navíc poskytuje velmi rozsáhlé veřejné API. Zmínit zasluhuje i *Wikimedia Commons*[6], obsahující aktuálně přes 16 milionů obrázků, převážně s licencí umožňující jejich volné použití. Poskytuje také několik různých implementací veřejného API. Existuje mnoho dalších, jako *Stock Exchange*[7], *Morgue File*[8], *Stockvault*[9] a další, nicméně naprostá většina neumožňuje získávat obrázky skrze API. Samozřejmě, lze použít i základního vyhledání skrze *Google images*[10], nicméně bez dostupného API by bylo velmi obtížné udržovat povědomí o již stažených obrázcích, aby nevznikaly duplicity a o licenci, s níž byl obrázek poskytnut.

2.2.4 Hodnocení obrázků

Po importování zdrojových obrázků je především třeba ohodnotit a vytřídit ty, u kterých by mohla být snadno automaticky nalezena vzpřímená poloha. U RoCAPTCHA testu se počítá hlavně s využitím obrázků ze serveru PetFinder.com a případně dalších, z nichž by bylo možno získávat především obrázky zvířat. Vzhledem k již zmíněným přednostem těchto obrázků již nebude třeba mnoho hodnotících algoritmů. Především jde o ohodnocení výskytu lidských obličejů a výrazných hran, rovnoběžných s horizontální a nebo s vertikální osou. Na základě výsledného hodnocení obrázku pak bude rozhodnuto, zda bude tento obrázek použit pro generování CAPTCHA testu.

Základní postup hodnocení zdrojových obrázků:

1. Odstranění poškozených a nečitelných souborů. Ačkoli by webové služby, ze kterých obrázky získáváme, měly udržovat svá data v konzistentním stavu, mnohdy tomu tak není.
2. Odstranění obrázků s menšími rozměry, než je minimální stanovená hranice.
3. Detekce obličejů – vytřízení obrázků, kde byl úspěšně detekován alespoň jeden lidský obličej.
4. Detekce hran – odstraníme všechny obrázky, u nichž by hrany mohly napovědět vzpřímenou polohu obrázku.

V následujících sekcích si podrobněji popíšeme poslední dva body.

Detekce obličejů

Existuje mnoho metod pro detekci a lokalizaci lidského obličeje v obrazu. Tyto metody lze klasifikovat do 4 kategorií, jež se mohou vzájemně překrývat [31].

- **Metody založené na znalostech** (*Knowledge-based*) předpokládají předchozí znalost prvků lidského obličeje. Popisují jednoduchými pravidly znaky lidského obličeje, jako například lokaci očí, úst a nosu a jejich relativní vzdálenosti. Z kandidátního obrázku jsou extrahovány všechny

[5] http://www.flickr.com/
[6] http://commons.wikimedia.org
[7] http://www.sxc.hu/
[8] http://www.morguefile.com/
[9] http://www.stockvault.net/
[10] http://images.google.com/

možné znaky obličeje a výsledek získáme porovnáním s referenčními znaky. Nevýhoda této metody spočívá v obtížné aplikaci na obličeje v různých pózách, proto se používá hlavně pro detekci obličejů v čelní pozici. Obvykle potřebují pro rozpoznání obrázky o větším rozlišení, než používají přiřazovací metody. Uplatňují se především pro lokalizaci obličeje.

- **Hledání neměnných znaků obličeje** (*Feature-invariant*) usiluje o hledání neměnných obličejových prvků (oči, nos, ústa, obočí, linie vlasů), které lze nalézt bez ohledu na aktuální pózu, osvětlení nebo pohled. Obvykle jsou tyto prvky extrahovány s pomocí detekce hran. Na základě těchto extrahovaných prvků je vytvořen statistický model, který ověří, zda prvky opravdu tvoří obličej. Výhodu tedy představuje větší nezávislost na variacích poloh obličeje, nicméně schopnost rozpoznání může být narušena šumem a osvětlením. Tyto metody jsou uplatňovány především pro lokalizaci obličeje.
- **Přiřazování k předloze** (*Template matching*) je jednoduchá metoda, spočívající v hledání normalizované korelace mezi obrázkem předlohy (objekt v trénovací množině) a obrázkem, který klasifikujeme. Použitá trénovací množina představuje soubor několika vzorů obličeje s uloženými údaji buď o celém obličeji, a nebo jen o některých jeho částech. Pro nalezení shody je třeba projít postupně všechny pixely v klasifikovaném obrázku, porovnat je s pixely obrázků v trénovací množině a zjistit jejich vzájemnou korelaci. Náročnost tohoto výpočtu roste úměrně s velikostí trénovací množiny. Tyto metody jsou velmi jednoduché a snadno pochopitelné, nicméně patří k těm nejpomalejším. Používají se k lokalizaci i k detekci obličejů.
- **Metody založené na vzhledu** (*Appearance-based*) řeší problém detekce obličeje analýzou jednotlivých částí obrazu. U těchto částí poté s pomocí klasifikátorů naučených na trénovací množině obrázků rozhodnou, zda se zde nachází obličej. Na rozdíl od přiřazování k předloze obsahuje trénovací množina rozmanitější obličeje a klasifikátor pro rozpoznání přítomnosti/nepřítomnosti obličeje s tím musí počítat. Tyto metody se používají především pro detekci obličejů a patří do nich neuronové sítě, Eigenfaces, distribuované metody, AdaBoost učení a jiné.

Pro ohodnocení obrázků s lidskými obličeji pro účely RoCAPTCHA testu postačuje pouze detekce lidských obličejů. Není třeba zjišťovat přibližnou polohu obličeje ani další údaje. Proto by bylo vhodné v implementaci používat pro detekci obličejů metodu založenou na vzhledu, neboť tento typ metod je pro prostou detekci lidských obličejů nejvhodnější.

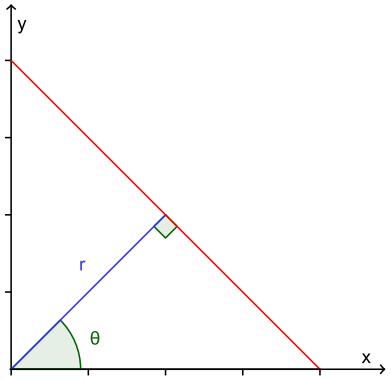
Přesnost detekce obličeje nemusí být vždy vysoká, je proto dobré zavést hodnotící metriky, jež budou vyjadřovat úspěšnost detekce obličeje. Zde je dobré brát v potaz nejen nalezení samotného obličeje, ale také jeho strukturálních prvků, například očí, úst a nosu.

Detekce hran

Detekce hran se obecně řadí pod metody extrakce prvků z obrazu. V rámci třídění zdrojových obrázků pro RoCAPTCHA test je třeba odstranit ty, u kterých by mohly hrany v obrázku napovědět alespoň přibližně jejich vzpřímenou polohu (horizontální a vertikální hrany). Potřebujeme získat především úhel, který svírají hrany s horizontální a nebo vertikální osou.

Tomuto zadáni nejlépe vyhovuje metoda detekce přímek v obraze s použitím **Houghovy transformace** [13, 29]. Tato transformace vychází ze směrnicového tvaru přímky $y = kx + q$ (k je směrnice přímky a q je druhá souřadnice průsečíku přímky s osou x). Hlavní idea Houghovy transformace definování přímky ne pomocí dvou bodů $(x_1, y_1)(x_2, y_2)$, ale s pomocí strmosti k a parametrem protnutí q. Problém nastává u přímek rovnoběžných s osou y, neboť v tomto případě roste hodnota q až do nekonečna. Proto se používá jiný přístup, a to použití polárních souřadnic r a θ (viz obrázek 2.4). Rovnici přímky poté můžeme přepsat na $r = xcos\theta + ysin\theta$. Ke každé přímce v obrázku můžeme přiřadit krivku danou parametry (r, θ), které jsou jedinečné, pokud $r \in \langle 0, \pi)$ a $r \in R$ nebo pokud $\theta \in \langle 0, 2\pi)$ a $r \geq 0$. Množina křivek (r, θ) se také někdy označuje jako Houghův prostor.

Co je důležité – pokud se křivky protínají, lokace průniku v Houghově prostoru určuje parametry přímky nacházející se na základních souřadnicích v původním zkoumaném obrázku. Problém nalezení přímek v obrázku je nyní přetvořen v problém nalezení průniku křivek. Toho můžeme dosáhnout s pomocí prostého prahování. Prahování je také třeba použít před samotnou aplikací Houghovy transformace, protože na vstupu musí být pouze jednotlivé body.



Obrázek 2.4: Reprezentace přímky pomocí křivky s parametry (r, θ). Parametr r reprezentuje vzdálenost mezi přímkou (červeně) a počátkem, zatímco θ představuje úhel vektoru od počátku do nejbližšího bodu přímky.

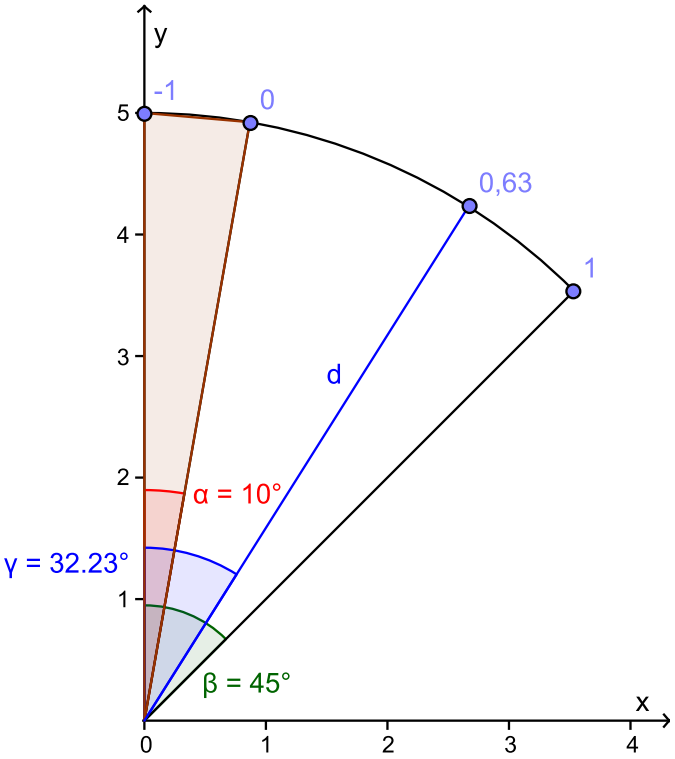
Výhodu Houghovy transformace představuje schopnost najít přímku, i když je tvořena nesouvisejícími body. Nevýhodou je citlivost na šum v obraze a a problematické použití na hledání jiných útvarů, než přímek a kruhů. Pro potřeby RoCAPTCHA testu však Houghova transformace postačuje.

Po získání jednotlivých přímek v obraze musíme vypočítat jeho celkové ohodnocení vzhledem k úhlům, které přímky svírají s horizontální nebo svislou osou. Výsledné ohodnocení bude odpovídat 0 při nenalezení žádných přímek, -1 při nalezení pouze přímek rovnoběžných s horizontální, nebo vertikální osou a 1 při nalezení přímek, které budou svírat úhel 45° a nebo 135° se svislou osou.

Pro celkový proces nalezení a ohodnocení hran v obraze navrhuji následující postup.

1. Převedení obrázku do stupňů šedi.
2. Vyprahování a detekce hran v obraze. Po tomto procesu v obraze zůstanou pouze body hran s maximální hodnotou šedi (255), ostatní body s nulovou hodnotou šedi (0). Je třeba správně zvolit hodnotu prahu při prahování.
3. Kruhové ořezání obrázku. Uživateli se bude zobrazovat pouze kruhový výřez obrázku (aby nebyla poznat správná poloha). Proto můžeme zanedbat oblasti obrázku mimo kruhovou oblast výřezu.
4. Detekce přímek v obrázku pomocí Houghovy transformace.
5. Převedení jednotlivých úhlů přímek do první poloviny I. kvadrantu kartézských souřadnic (0° až 45°). To nám ulehčí následné hodnocení.
6. Ohodnocení úhlů dostaneme na základě výpočtu $(\gamma - \alpha)/(45 - \alpha)$ pro úhel větší, než α a $-(1-\gamma)/\alpha$ pro úhel menší, než α, viz obrázek 2.5.

7. Zprůměrování všech ohodnocení přímek do jedné výsledné hodnoty *r*.
8. Úprava výsledného hodnocení pomocí penalizace v případě, že bylo nalezeno méně přímek, než p_{max}. V takovém případě získáme výsledné hodnocení jako $r = r/((p_{max} - (p_{pocet} - 1)) * m)$ kde p_{pocet} je počet nalezených přímek a m je multiplikátor určující strmost penalizace. Toto opatření koriguje nepřesnost algoritmu pro hledání přímek.



Obrázek 2.5: Grafické znázornění metody ohodnocování přímek podle jejich úhlu ke svislé ose. Úhel β představuje maximální ohodnocení (1), úhel α ambivalentní ohodnocení (0) a úhel 0° negativní ohodnocení (-1). Pro příklad ohodnocení zde uvádím přímku d, která svírá se svislou osou y úhel $\gamma = 32,23°$, tudíž se její ohodnocení rovná $0,63$.

2.3 SegCAPTCHA

Pracovní název SegCAPTCHA představuje zkratku ze spojení ***Seg**mentation **CAPTCHA***, hlavním principem je v tomto případě segmentace.

Automatické vyřešení CAPTCHA testu obvykle sestává ze dvou částí – segmentace a rozpoznávání. Co se týká rozpoznávání, výzkum ukázal [4], že počítače jsou schopny rozpoznávání jednotlivých písmen z obrázku lépe, než lidé. U rozpoznávání obrázků je situace samozřejmě složitější, nicméně vývoj pokračuje mílovými kroky. Podle výše zmíněného výzkumu však problém segmentace představuje pro lidi podobnou překážku jako pro počítače. CAPTCHA testy by se tedy měly spíše zaměřit na segmentaci.

2.3.1 Princip

SegCAPTCHA využívá především segmentace, nicméně zahrnuje v sobě také problém rozpoznávání pro lepší odolnost vůči automatickému útoku. Předpokladem pro vygenerování SegCAPTCHA testu je množina obrázků, jež obsahují jeden určitý objekt (například jablko, pomeranč, pánev apod.). U každého obrázku musíme tento objekt dopředu znát. Samotný SegCAPTCHA test sestává z několika obrázků těchto objektů, jež jsou vygenerovány s určitým překryvem tak, aby nikdy tento objekt nebyl zobrazen samostatně (bez úplně bez překryvu), ale zároveň musí být alespoň jeho část vidět. Úkolem uživatele je označit všechny objekty určitého druhu (jablka, pomeranče, pánve apod.).

Tento navrhovaný CAPTCHA test poskytuje několik výhod. Především je ho možno oproti RoCAPTCHA testu generovat, čímž dostaneme potenciálně nekonečné množství variant. Také je možno jej velmi dobře řešit na mobilních zařízeních s pomocí dotyku. Dále nevyžaduje příliš místa, neboť je zobrazen jako jeden obrázek a může být populárnější mezi provozovateli internetových aplikací (zbude více místa pro obsah).

Toto řešení má ovšem i své nevýhody. Především jde o závislost na jazyku, neboť uživateli musíme předat informaci o tom, jaké druhy předmětů má na obrázku označit. Další nevýhodou může být snížená úspěšnost u lidí při zvolení nevhodných předmětů, jež by mohly při překrytí splývat díky stejnému zbarvení nebo tvaru. Tuto domněnku je třeba experimentálně vyvrátit.

Princip SegCAPTCHA testu však nebyl příliš prozkoumán, neboť nebyl shledán tak uživatelsky zajímavý a použitelný jako RoCAPTCHA.

2.3.2 Bezpečnost

Co se týče bezpečnosti, na rozdíl od RoCAPTCHA testu vykazuje velkou odolnost vůči útoku náhodným hádáním, neboť pravděpodobnost náhodného uhádnutí roste kvadraticky s množstvím objektů, jež mají za úkol uživatelé označit. Konkrétně při 4 objektech a ploše obrázku 200x200px bychom dostali pravděpodobnost $p = 1/(200 * 200)^4 = 3,9 * 10^{-19}$, což je o mnoho méně než obecně platná tolerance 0,01%. Rekonstrukce databáze obrázků a následné použití metody přímého přiřazování by byla také mnohem obtížnější, neboť nikdy nejsou na jednom testu vidět kompletní obrázky (vždy jsou některé části překryté) a navíc mohou být libovolně natočeny, umístěny a může být upravena jejich velikost. Navíc můžeme ještě aplikovat další transformace a filtry pro ještě lepší zabezpečení. Je otázka, jak dobře by si tento test vedl vůči útoku strojovým učením. V tomto případě by však bylo třeba provést segmentaci, jež může být dále znesnadněna například použitím obrázků s podobnými barvami.

Kapitola 3

Implementace

V rámci této práce byly implementovány dva CAPTCHA testy s pracovními názvy RoCATCHA a SegCAPTCHA. Oba jsou postaveny na stejném technologickém základu jak co se týče použitých technologií, tak způsobu komunikace (API).

V rámci této práce bylo bráno v potaz pouze řešení na bázi vizuálního přístupu. Řešení s pomocí jiných přístupů (například zvuku) by bylo nad rámec této práce.

V následujících sekcích bude rozepsány konkrétní implementační aspekty. Nejprve budou uvedeny obecné prvky, jež obě zde implementovaných řešení sdílejí (technologie, webové rozhraní a API). Poté bude podrobněji rozebrána implementace RoCAPTCHA a SegCAPTCHA testu a na závěr vyhodnocení uživatelského testování RoCAPTCHA testu.

3.1 Použité technologie

V následujících sekcích budou popsány technologie, použité pro implementaci CAPTCHA testů. Všechny z těchto technologií jsou poskytovány zdarma i pro komerční účely. U některých technologií je v názvu uvedena verze, jež byla použita pro implementaci.

Python 2.7

Python[1] je dynamicky typovaný programovací jazyk. Spravuje jej *Python Software Foundation*[2] a poskytuje pod vlastní licencí. V této práci byl použit z několika důvodů.

- Multiplatformnost,
- jednoduchá syntax,
- objektově orientovaný,
- velká modularita a hierarchické balíčky,
- mnoho standardních knihoven a modulů třetích stran,
- podpora modulů i v jiných jazycích, jako C, C++ a další,
- velmi užitečné zabudované datové struktury (list, dict, tuple).

Jednoduchá syntax Pythonu společně s mnoha dostupnými moduly třetích stran umožňuje velmi rychlý vývoj. Python je navíc poměrně rychlý, nicméně stále ne tak rychlý jako například programy napsané v C. Tuto nevýhodu vyvažuje právě rychlejším vývojem. Pokud by se ukázalo, že v reálných

[1] http://python.org/
[2] http://python.org/psf/

podmínkách jeho rychlost nestačí, vždy je zde možnost přepsat již odzkoušený algoritmus do rychlejšího jazyka, ovšem s menším úsilím, než kdyby byl tento algoritmus psán přímo pro finální jazyk.

V neposlední řadě mám s tímto jazykem poměrně velké zkušenosti.

Použité Python knihovny třetích stran:

- PIL[3].
- pytz 2013b[4].

Django 1.5

Django[5] představuje vysokoúrovňový webový framework postavený na Pythonu. Je spravován společností *Django Software Foundation*[6], která jej poskytuje zdarma jako open-source. Následující seznam zachycuje některé z výhod Djanga.

- Rychlý vývoj s DRY[7] principem,
- automaticky generovaná administrace,
- vlastní ORM[8],
- velké množství komponent třetích stran,
- MVC[9] architektura,
- zabudovaní internacionalizace,
- cachování a další.

V rámci této práce bude Django použito hlavně pro vybudování rozhraní mezi klientem a vlastní logikou CAPTCHA serveru. Kromě toho bude vytvořena komponenta pro snadné použití CAPTCHA testu v Django aplikacích třetích stran.

Django jsem zvolil také proto, že jsem v tomto frameworku již v minulosti vytvořil několik projektů.

Použité aplikace třetích stran pro Django, jež bylo třeba doinstalovat zvlášť:

- django-bootstrap-toolkit 2.5.6[10]
- django-localeurl 1.5[11]
- django-enumfield 1.0c1[12]

[3] http://www.pythonware.com/products/pil/
[4] https://pypi.python.org/pypi/pytz/2013b
[5] https://www.djangoproject.com/
[6] https://www.djangoproject.com/foundation/
[7] *Don't Repeat Yourself* – neopakuj se
[8] *Object-Relational Mapping* – objektově-relační mapování
[9] *Model-View-Controller*
[10] https://github.com/dyve/django-bootstrap-toolkit
[11] https://pypi.python.org/pypi/django-localeurl
[12] https://pypi.python.org/pypi/django-enumfield/1.0c1

MySQL 5.5.27

Jako databázový systém byla zvolena relační databáze MySQL[13] od společnosti *Oracle*[14]. Hlavní výhodu představuje open source distribuce a multiplatformnost. Navíc je to jeden z nejrozšířenějších databázových systémů na světě.

Apache HTTP server 2.4.3

Pro provoz serverové části CAPTCHA testu byl použit open source HTTP server Apache[15], spravovaný společností *The Apache Software Foundation*[16]. Jde o vůbec nejpoužívanější webový server na světě, kompatibilní s již zméněnými technologiemi, což z něj dělá logickou volbu.

OpenCV 2.4.5

OpenCV[17] (zkratka z *Open source Computer Vision*) představuje multiplatformní open source knihovnu pro počítačové vidění. Obsahuje přes 2500 optimalizovaných algoritmů jak klasických, tak s použitím strojového učení. Umožňuje například detekovat objekty, obličeje a hrany v obraze, což bude v této práci použito. Ačkoli je knihovna OpenCV napsaná převážně v C a C++, obsahuje rozhraní pro použití v Javě a Pythonu.

HTML5

HTML5 je značkovací jazyk pro definici struktury a vzhledu obsahu webových stránek. Vychází z předchozích verzí HTML 4.01, XHTML 1.1 a přidává další značky a funkčnost[18]. Specifikaci HTML5 spravuje konsorcium W3C[19] (zkratka z *World Wide Web Consortium*). V době vydání této práce byla většina vlastností HTML5 podporována všemi nejpoužívanějšími internetovými prohlížeči. Tato technologie bude v tomto projektu použitá společně s CSS3 pro vytvoření webové stránky projektu a také pro vytvoření rozhraní CAPTCHA testu.

CSS3

CSS3 představuje nejnovější verzi CSS (zkratka z *Cascading Style Sheets* – kaskádové styly), jež umožňují popsat způsob zobrazení webových stránek. Specifikaci CSS3 spravuje konsorcium W3C, stejně jako u HTML5. Pro potřeby RoCAPTCHA testu bylo z této technologie použito především transformací, umožňujících otáčení obrázku.

[13] http://www.mysql.com/
[14] http://www.oracle.com/
[15] http://httpd.apache.org/
[16] http://www.apache.org/
[17] http://opencv.org/
[18] http://www.w3.org/TR/2010/WD-html5-20100624/
[19] http://www.w3.org/

JavaScript

Další z technologií je JavaScript[20] – objektově orientovaný skriptovací jazyk. Zde je použit především pro ovládání CAPTCHA testu v internetovém prohlížeči klienta.

JSON

JSON (zkratka z *JavaScript Object Notation*) – formát pro výměnu dat, který je jednoduše čitelný, stručný a editovatelný pro lidi i stroje. V této práce je použit pro formátování přenášených dat mezi jednotlivými účastníky komunikace.

Python Flickr API kit[21]

Tato knihovna poskytuje rozhraní v pythonu pro používání Flickr API. Výhodu představuje dobrá dokumentace a velké pokrytí API Flickru.

3.2 Webová aplikace

V rámci této práce byla vytvořena webová aplikace jednak pro poskytování API a také pro prezentaci dokumentace a ukázek implementace. Tato aplikace byla implementováno ve frameworku Django. Pro provozování byly zakoupeny domény rocaptcha.com (primární) a rocaptcha.net (počítá se s možnou budoucí popularitou této služby). Na těchto doménách byl systém v době psaní této práce testován. Kromě funkční ukázky RoCAPTCHA testu je na webovém rozhraní také umístěn registrační formulář pro získání veřejného a privátního klíče, dokumentace API a dokumentace implementace do aplikací třetích stran. Navíc je přes výše zmíněné domény dostupné API pro generování a ověřování RoCAPTCHA testu třetími stranami. Ukázku vzhledu této aplikace můžete najít v příloze A.1.

Celý systém běží na HTTP serveru Apache s pomocí modulu *mod_wsgi* v daemon módu.

Databázové schéma bylo vygenerováno automaticky s pomocí Djanga na základě navrhnutých modelů. Výsledné ER diagramy výsledných tabulek RoCAPTCHA i SegCAPTCHA testu můžete nalézt v příloze B.2 a B.1.

Struktura webové aplikace

Struktura webové aplikace se podřizuje především obecným rozvržením Django projektů. Adresář projektu obsahuje soubory pro správu a spuštění projektu (`manage.py`, `wsgi.py`) a adresáře jednotlivých aplikací. Tyto aplikace obsahují především Python soubory pro jednotlivé aspekty MVC přístupu – pro model `models.py` a pro kontrolér `views.py`. Pohled je tvořen HTML šablonami, jež jsou obvykle umístěny ve složce templates (buď na úrovni projektu a nebo u aplikací). Dále aplikace často obsahuje soubor `urls.py`, jež popisuje mapování URL adres na pohledy, zpracovávající požadavky.

Konkrétní struktura aplikace sestává z následujících souborů a složek (popsána pouze základní úroveň).

[20] http://en.wikipedia.org/wiki/JavaScript
[21] http://stuvel.eu/projects/flickrapi

- **captcha** – Hlavní aplikace, jež se stará o zobrazení webového rozhraní. Obsahuje především základní nastavení a mapování URL.
- **captcha_server** – Bázová aplikace obsahující moduly, z nichž dědí aplikace pro konkrétní implementaci CAPTCHA testu.
- **libs** – Adresář obsahující znovupoužitelné moduly a moduly třetích stran.
- **media** – Adresář obsahující zdrojové a vygenerované obrázky jednotlivých CAPTCHA testů.
- **rocaptcha** – Plugin pro RoCAPTCHA test.
- **rocaptcha_server** – Aplikace pro správu RoCAPTCHA testu.
- **segcaptcha** – Plugin pro SegCAPTCHA test.
- **segcaptcha_server** – Aplikace pro správu SegCAPTCHA testu.
- **static** – Adresář pro poskytování statických souborů, jež jsou potřeba pro provoz všech aktivních aplikací (CSS, obrázky apod.). Obsah je spravován automaticky pomocí Djanga.
- **staticfiles** – Adresář obsahující základní statické soubory pro provoz této webové aplikace (CSS, obrázky apod.).
- **manage.py** – Python skript pro spouštění základních příkazů pro správu aplikace.
- **run_makemessages.py** – Python skript ulehčující vytvoření a kompilaci překladů webového rozhraní do jiných jazyků.

3.3 API

Životní cyklus CAPTCHA testu je možno rozdělit do dvou kroků.

- Komunikace mezi uživatelem (jeho koncovým zařízením) a CAPTCHA serverem, jež sestává ze 3 kroků.

    1. Získání JavaScriptového kódu, který bude ovládat další proces zobrazení a řešení CAPTCHA testu.
    2. Získání HTML kódu rozhraní CAPTCHA testu.
    3. Generování nového CAPTCHA testu a získání jeho identifikátoru (hash).

- Komunikace mezi aplikačním serverem a CAPTCHA serverem. Zde se odesílá pouze jeden požadavek.

    1. Ověření správnosti uživatelovy odpovědi.

Pro lepší kontrolu nad tím, kdo využívá RoCAPTCHA test a také pro lepší odolání možnému útoku je zavedeno používání veřejného a privátního klíče pro identifikaci požadavků. Provozovatel internetové aplikace, jež chce využít služeb RoCAPTCHA testu, se proto nejdříve musí zaregistrovat skrze webové rozhraní, kde mu bude vygenerován veřejný a privátní klíč. Tyto klíče bude poté používat při komunikaci s RoCAPTCHA API.

Konkrétní požadavky, jejich parametry a odpovědi na ně zachycuje tabulka 3.1. V případě prvních tří požadavků je parametrem *key* myšlen veřejný klíč, u posledního jde ale o privátní klíč. Parametr *lang* umožňuje nastavit jazyk rozhraní (zatím pouze *cs* pro češtinu a nebo *en* pro angličtinu). U požadavku `/api/verify/` jsou povinné parametry *hash* (identifikační hash CAPTCHA testu), *response* (hodnota odpovědi), *remote_ip* (IP adresa uživatele nebo aplikačního serveru), *session_id* (klíč sezení) a poslední nepovinný parametr *lang*, se stejným významem jako u předchozích požadavků.

Tabulka 3.1: Podrobný popis API. Parametry značené prázdným kroužkem jsou nepovinné. Zástupný řetězec `<status_code>` vyjadřuje jeden ze stavů z tabulky D.1.

URL	Typ požadavku	Parametry	Odpověď
/api/js/	GET	• key ○ lang	JavaScript
/api/form/	GET	• key ○ lang	JSON: `{'status': <status_code>, 'content': <html_form>}`
/api/generate/	GET	• key ○ lang	JSON: `{'status': <status_code>, 'content': <hash>}`
/api/verify/	POST	• key • hash • response • remote_ip • session_id ○ lang	JSON: `{'status': <status_code>}`

Klíč sezení je zde potřeba pro implementaci metody Token Bucket [16], neboť umožňuje identifikovat uživatele unikátnějším způsobem než IP adresa. Při prvním požadavku od uživatele na CAPTCHA server (`/api/js/`) se automaticky vytvoří klíč sezení, jež se poté vloží do JavaScript kódu, zaslaného klientovi jako odpověď. Klíč sezení se poté zapíše do formuláře, jež bude po vyřešení CAPTCHA testu odeslán na aplikační server. Požadavku na verifikaci z aplikačního serveru na CAPTCHA server se poté nastaví stejný klíč sezení, jaký byl vytvořen při první komunikaci uživatele s CAPTCHA serverem. CAPTCHA server je na základě přijatého klíče sezení schopen identifikovat jednotlivé uživatele a podle toho jim odečítat/přičítat tokeny v jejich vědrech. Při konkrétní implementaci metody Token Bucket bylo postupováno přesně podle 2.1.1, přičemž *TB-Max* = 100 a *TB-Refill* = 3.

Samotný CAPTCHA test se může nacházet v několika stavech, jež jsou uvedeny v tabulce D.2.

3.4 RoCAPTCHA

V následujících sekcích bude rozvedena implementace jednotlivých částí RoCAPTCHA testu. Jednotlivé sekce tvoří sled operací, jež hrají vitální roli v životním cyklu tohoto testu – import a ohodnocení zdrojových obrázků, generování a následná verifikace. Ke konci bude ještě zmíněna implementace uživatelského rozhraní a pluginů pro jazyky a aplikace třetích stran.

3.4.1 Import zdrojových obrázků

Při implementaci importování zdrojových obrázků jsem bral v potaz dvě možné varianty, jak tyto obrázky získat. Obě tyto varianty jsou implementovány jako rozšíření základních akcí, jež jsou dostupné v Djangu přes skript *manage.py*. Používají se skrze příkazovou řádku, což umožňuje například zavedení pravidelného spouštění přes CRON[22] a nebo dávkové zpracování.

Import z lokálního úložiště

Pro import z lokálního úložiště byla vytvořena akce `collect_images_manual` <`path_to_folder`>; parametr <`path_to_folder`> určuje cestu k adresáři s obrázky pro import). Vstupní data v adresáři pro import však musí mít specifickou strukturu. Kořenová složka určuje zdroj daných obrázků (například *Flickr*). V této složce mohou být další podsložky, ve kterých mohou být opět další podsložky a nebo obrázky pro import. Tato hierarchie složek určuje tagy obrázků. Například pokud je obrázek ve složce `/Flickr/Cat/White/`, můžeme z toho určit, že zdroj obrázků je galerie *Flickr* a obrázky obsahují tagy *Cat* a *White*. Po zjištění těchto údajů je každý obrázek spolu s těmito údaji zařazen do databáze.

Neprobíhá zde žádná kontrola duplicit ani ohodnocení obrázků, s touto metodou se počítá hlavně pro počáteční naplnění databáze daty.

Import přes API třetích stran

Na rozdíl od importu z lokálního úložiště je tato metoda importu vytvořená pro pravidelnou aktualizace databázových dat. Při implementaci bylo bráno v úvahu několik možných zdrojů obrázků, především však Flickr, PetFinder a WikiMedia Commons, jelikož všechny umožňují přístup přes API.

Prozatím bylo implementováno stahování obrázků přes API Flickru, neboť toto API se pro účely RoCAPTCHA testu ukázalo jako nejjednodušší. Na rozdíl od API PetFinderu a WikiMedia Commons umožňuje s pomocí jednoho volání `flickr.photos.search`[23] získat (po stránkách) všechny fotografie, jež mají určité zadané parametry (například licenci, klíčová slova apod.). Jde především o obrázky s volnou licencí a nebo alespoň s licencí, požadující odkaz na autora. Konkrétní invokace tohoto importu je implementována přes akci `collect_images` <`how_many`>; parametr <how_many> určuje, kolik obrázků se má při jednom volání importovat. Pro implementaci importu obrázků z Flickr API byl použit *Python Flickr API kit*. Při procesu importu se kontrolují duplicitní obrázky, zjišťují se informace o autorovi a licenci obrázku a také se zjistí výsledné ohodnocení (viz 3.4.2).

PetFinder poskytuje poměrně přehledné API s dobrou dokumentací, nicméně pro účely této práce není příliš vhodné, neboť neumožňuje vyhledat všechna aktuálně dostupné zvířata, ale jen podle jednotlivých států USA. API WikiMedia Commons poskytuje sice více různých API, nicméně také poměrně zmatenou a složitou dokumentaci, takže možnost importu s pomocí této služby nebyla prozatím implementována.

Především bylo experimentálně prokázáno, že průběžné rozšiřování báze dat importem obrázků přes API třetích stran je funkční řešení a v případě větší popularity RoCAPTCHA testu není problém

[22] Softwarový plánovač, který umožňuje na linuxových systémech spouštět příkazy a nebo procesy v určitou dobu a nebo v určitých intervalech.

[23] http://www.flickr.com/services/api/flickr.photos.search.html

implementovat i import z dalších webových služeb.

3.4.2 Ohodnocení obrázků

Importované zdrojové obrázky před zařazením do sady použité pro generování RoCAPTCHA testu procházejí několika třídícími a hodnotícími algoritmy. Na začátku procesu se všechny importované obrázky nacházejí v databázi a mají vyplněné pouze údaje o původu, ne však o hodnocení. V prvním stupni jsou označeny obrázky, které mají špatný datový formát, nejdou přečíst a nebo jsou nějakým jiným způsobem poškozené. Tyto obrázky již se dál nehodnotí a při pravidelném úklidu databáze budou smazány. V další fázi se hodnotí přítomnost hran a obličejů v obraze, což bude podrobněji popsáno v následujících sekcích. Všechny zde zmíněné hodnotící algoritmy jsou součástí modulu `classifiers`, jež se nachází v aplikaci `rocaptcha_server`. Po aplikaci všech hodnotících algoritmů máme u každého záznamu o obrázku v databázi informace o jednotlivých hodnoceních, na jejichž základě se poté nastaví flag *is_suitable* (logická hodnota pravda/nepravda), určující, zda je obrázek vhodný pro použití v RoCAPTCHA testu. Uchovávat informaci o výsledcích jednotlivých hodnotících algoritmů se může vyplatit, pokud bychom v budoucnu chtěli tyto algoritmy pozměnit. V takovém případě nebude třeba znovu propočítat ohodnocení všech algoritmů, ale jen změněných.

Ohodnocení výskytu hran

Při zvolení konkrétní implementace ohodnocení hran jsem bral v potaz dvě možnosti. V první řadě jsem experimentoval s možností využití automatizovaného skriptu v Matlabu[24] od autora Tao Penga, umožňující s pomocí Houghovy transformace detekovat v obraze úsečky[25]. Tento postup fungoval s poměrně velkou úspěšností, nicméně při spojení ohodnocování s importem obrázků se ukázalo, že je velmi problematické spouštět matlabovské skripty z Pythonu. Matlab navíc představuje komerční software.

Ve finální implementaci jsem použil open source knihovnu OpenCV, jež v sobě obsahuje metody pro aplikaci Houghovy transformace a také rozhraní pro Python.

Algoritmus hodnocení výskytu hran v obraze provádí následující kroky.

1. Převedení do stupňů šedi, což nám usnadní prahování.
2. Detekce hran obrázku s pomocí Canny operátoru knihovny OpenCV.
3. Oříznutí obrázku do čtvercového tvaru.
4. Všechny části obrazu mimo vnitřní kruh nastavit na černou barvu, takže nám zbudou jen data z oblasti, kterou při řešení RoCAPTCHA testu vidí uživatel (viz obrázek 3.1b).
5. Detekce úseček v obraze s použitím probabilistické Houghovy transformace pomocí metody `HoughLinesP` z knihovny OpenCV (viz obrázek 3.1c).
6. Vypočítání úhlu každé úsečky ke svislé ose a převedení tohoto úhlu do horní části I. kvadrantu.
7. Získání ohodnocení každého úhlu v intervalu $\langle -1, 1 \rangle \in \mathbb{R}$, viz obrázek 2.5.

[24] http://www.mathworks.com/products/matlab/

[25] http://www.mathworks.com/matlabcentral/fileexchange/9226-detect-lines-in-grayscale-image-using-hough-transform/content/Hough_Grd.m

8. Vytvoření váženého průměru všech úhlů vzhledem k délce úsečky, vázající se na úhel.
9. Penalizace váženého průměru, pokud je počet nalezených úseček menší než p_{min}.

(a) Originální obrázek (autor: Penn State[26]).

(b) Obrázek po detekci hran s pomocí canny operátoru a oříznutí.

(c) Původní obrázek s červeně vyznačenými detekovanými úsečkami.

Obrázek 3.1: Postup detekce úseček v obrazu.

Ohodnocení výskytu obličejů

Pro ohodnocení výskytu obličejů bylo použito metody `cv.HaarDetectObjects` z knihovny OpenCV (rozhraní pro Python) společně s několika sety Haar kaskád[27], natrénovaných na rozpoznávání obličeje zepředu, z profilu, obou očí zepředu a jednotlivých očí.

Konkrétní postup nalézaní obličejů v obraze probíhá v několika krocích. Nejdříve se v obraze hledají potenciální obličeje zepředu a z profilu. V případě úspěšného nalezení se ještě pro každou nalezenou oblast zjistí, zda neobsahuje také oči, což by zvýšilo pravděpodobnost úspěšného nálezu (viz obrázek 3.2a).

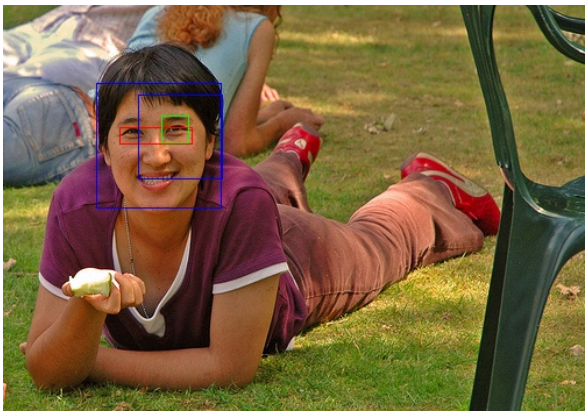
Výsledkem ohodnocení obrázku je desetinné číslo v intervalu $\langle 0, 1\rangle \in \mathbb{R}$, kde 0 představuje obrázek, jež se 100% určitostí obsahuje obličej a 1 obrázek, jež obličej neobsahuje.

3.4.3 Generování

Při přijetí požadavku na generování RoCAPTCHA testu se nejdříve aplikuje kontrola všech požadovaných hodnot, zachycených v sekci API 2.1.2. Konkrétní postup při generování RoCAPTCHA testu je následující.

1. Vygenerování náhodného úhlu otočení v intervalu $\langle T, 360 - T\rangle \in \mathbb{N}$, kde $T \in \mathbb{N}$ je tolerance odchylky úhlu.
2. Vybrání jednoho náhodného obrázku ze všech obrázků, jež mají flag *is_suitable* nastavený na logickou hodnotu *pravda*.
3. Oříznutí obrázku na požadovanou čtvercovou velikost (experimentálně se používá rozměr 220px).

[26] http://www.flickr.com/photos/pennstatelive/
[27] http://alereimondo.no-ip.org/OpenCV/34
[28] http://www.flickr.com/photos/lathos/



(a) Fotografie s úspěšně detekovaným obličejem. Modré obdélníky určují oblast obličeje zepředu a z profilu, červeně je vyznačena oblast obou očí a zeleně oblast s okem.

(b) Fotografie se špatně rozpoznanými obličeji.

Obrázek 3.2: Ukázka detekce lidských obličejů (Autor fotografií: Simon Cozens[28]).

4. Otočení obrázku o dříve vygenerovaný úhel.
5. Aplikace kruhové masky tak, že oblast mimo vnitřní kruh je nastavena na černou barvu.
6. Vygenerování náhodného alfanumerického řetězce pro identifikaci RoCAPTCHA testu (hash) o délce 15 znaků.
7. Uložení výsledného obrázku jako soubor *hash*.JPG.
8. Uložení nového RoCAPTCHA testu do databáze (ukládá se hash, vygenerovaný úhel, cesta k obrázku, IP adresa uživatele a informace o internetové aplikaci, z níž požadavek přišel).

Obrázek 3.3: Ukázka finálního vygenerovaného obrázku pro jeden RoCAPTCHA test.

Na obrázku 3.3 můžete vidět finální vygenerovaný obrázek, jež se zobrazí uživateli (části mimo vnitřní kruhovou výseč jsou při procesu zobrazování uživateli skryty).

3.4.4 Verifikace

Stejně jako u generování, i zde se před samotnou verifikací ověřuje zadání všech požadovaných hodnot, což je zachyceno v sekci API 2.1.2. Postup při verifikaci správnosti řešení RoCAPTCHA testu pracuje následovně.

1. Získání RoCAPTCHA testu z databáze podle přijatého identifikačního hash kódu.
2. Pokud se nalezený RoCAPTCHA test nachází v jiném stavu, než *SOLVING*, je vrácena odpověď *FAILED*, neboť jde zřejmě o vícenásobný požadavek.
3. Kontrola, zda je řešení testu přijato ještě před vypršením časového limitu (experimentálně nastaveno 20min). Pokud již limit uplynul, posílá se odpověď *TIMEOUT*.
4. Kontrola, zda se řešení test (úhel otočení) nachází v mezích, určených intervalem $\langle -T, 0\rangle \in \mathbb{Z}$ nebo $(0, T\rangle \in \mathbb{Z}$, kde T je tolerance odchylky úhlu. Pokud se přijaté řešení nachází v daném intervalu, aplikačnímu serveru se vrátí odpověď *PASSED*, v opačném případě *FAILED*. V případě odpovědi *PASSED* se ale ještě před jejím odesláním provede kontrola neprázdnosti vědra sezení metody Token Bucket (viz sekce API 2.1.2).
5. Nakonec se ještě provede uložení statistik o úspěšnosti řešení a dalších údajích do databáze.

3.4.5 Rozhraní

Uživatelské rozhraní RoCAPTCHA testu přestavuje jednu z jeho nejdůležitějších částí, neboť zde probíhá samotné řešení testu. Ukázku rozhraní lze vidět na obrázku 3.4. V levé části rozhraní je umístěn obrázek, jež má uživatel za úkol otočit do správné pozice. K tomu má k dispozizi dvě možné metody.

- **Otočení s pomocí kurzoru myši.** Funguje na principu drag&drop – uživatel může kdekoli nad obrázkem stisknout levé tlačítko myši a při jeho držení posunovat ukazatel myši po obrázku, což způsobí otáčení obrázku vzhledem k aktuální poloze ukazatele myši.
- **Otočení s pomocí dotyku.** Pokud uživatel přistupuje k testu skrze zařízení, podporující dotykové ovládání, může otáčení obrázku ovládat prstem podobně, jako v předchozím případě.

Technickou stránku rotování obrázků zajišťuje CSS3 transformace `transform: rotate`. V rámci vývoje byla otestována i metoda s použitím HTML5 elementu canvas, nicméně to se ukázalo jako zbytečně komplikované řešení. Canvas nabízí velkou variabilitu možností, nicméně v tomto případě bohatě stačí pouze zajistit otáčení obrázku. Nehledě na fakt, že canvas i CSS3 transformace rotace jsou podporovány od přibližně stejných verzí internetových prohlížečů.

Pro obsluhu událostí klienta je zde použit JavaScript ve své základní podobě, neboť při použití frameworku (například jQuery[29]) by se tento framework musel posílat uživateli společně s vlastním JavaScriptovým kódem, neboť nelze spoléhat na dostupnost tohoto frameworku v aplikaci, kde se RoCAPTCHA zobrazuje. Toto opatření navíc šetří i server, na kterém RoCATPCHA běží, neboť při velkém množství požadavků na zobrazení testu by velmi narůstal datový přenos.

[29] http://jquery.com/

Obrázek 3.4: Ukázka uživatelského rozhraní RoCAPTCHA testu.

Rozhraní také umožňuje načíst další test, pokud se aktuálně načtený uživateli jeví jako příliš těžký (tlačítko *Reload*). Další z ovládacích prvků představuje tlačítko *Author*, jež otevře novou záložku v internetovém prohlížeči a načte zde stránku se zdrojovým obrázkem aktuálně zobrazeného RoCAPTCHA testu společně s informacemi o autorovi a odkaz na zdroj, ze kterého byl obrázek získán. To vše ale za předpokladu, že uživatel již tento test buď vyřešil, nechal si načíst jiný s pomocí tlačítka *Reload* a nebo počkal do uplynutí časového limitu na vyřešení RoCAPTCHA testu. Bez splnění alespoň jedné z těchto podmínek je uživateli zobrazena pouze stránka s výše zmíněnými podmínkami. Poslední z ovládacích prvků je tlačítko *Help*, jež by mělo zobrazovat informaci o tom, jak postupovat při řešení RoCAPTCHA testu, nicméně prozatím pouze přesměruje uživatele na oficiální stránku RoCAPTCHA testu, kde jsou tyto informace dostupné.

Vzhled rozhraní je prozatím tvořen s pomocí inline CSS3 stylů, nicméně můžou zde vznikat konflitky s použitými styly v jednotlivých webových aplikacích, kde se tento test bude zobrazovat. Proto by v budoucnu měl být celý vzhled rozhraní tvořen především obrázky, neboť ty nejsou interpretací stylových informací ovlivněny.

3.4.6 Pluginy

Pluginy umožňují co nejjednodušší možnost implementace RoCAPTCHA testu do internetových aplikací třetích stran. Sestávají obvykle ze dvou částí – jedna část se stará o vykreslení rozhraní RoCAPTCHA testu v internetovém prohlížeči koncového uživatele (HTML, CSS a JavaScript) a druhá pracuje na straně serveru, umožňujíc verifikaci správnosti řešení RoCAPTCHA testu (odesláním HTTP požadavku na RoCAPTCHA server).

V rámci této práce byly vytvořeny dva pluginy pro RoCAPTCHA test, oba jsou dostupné online ze serveru GitHub[30].

- Plugin pro **Django** – `https://github.com/flaiming/django-rocaptcha` inspirovaný django-

[30] `https://github.com/`

recaptcha widgetem[31].

- Plugin pro **PHP** – https://github.com/flaiming/rocaptcha-php inspirovaný recaptcha pluginem[32].

Aktuálně je ve vývoji plugin pro myBB[33] fórum a v budoucnu budou dostupné pluginy i pro další jazyky a redakční systémy.

3.5 SegCAPTCHA

Implementace SegCAPTCHA testu vychází z podobných principů, jako RoCAPTCHA. Tím je myšlena především nutnost nejdříve vytvořit databázi zdrojových obrázků, ze které se poté může generovat SegCAPTCHA test. Jak už bylo řečeno v sekci 2.3, zdrojové obrázky musí splňovat určité vlastnosti. Především musí obsahovat pouze jeden objekt, neboť znalost počtu objektů nám napomůže k určení správnosti řešení. Okolí daného objektu musí být vyplněno transparentní barvou. Tento požadavek vychází z čistě praktického hlediska, neboť při generování bude třeba, aby se jednotlivé objekty částečně překrývaly.

Pro účely testování bylo manuálně získáno několik obrázků z WikiMedia Commons[34] a pomocí hledání na Google Obrázky[35]. U obrázků, jež neobsahovaly průhlednost byla tato průhlednost doplněna manuálně. Toto se však nepovedlo úplně dokonale, neboť tato operace nepočítá s postupným přechodem mezi pozadím a objektem, takže kolem objektu může zůstat aura v barvě odstraňovaného pozadí. Při reálném použití by se měl tento proces automatizovat, zde bylo použito manuální získání a ladění obrázků z důvodu testování.

3.5.1 Generování

Při přijetí požadavku na generování SegCAPTCHA testu se nejdříve aplikuje kontrola všech požadovaných hodnot, zachycených v sekci API 2.1.2. Konkrétní postup při generování SegCAPTCHA testu je následující.

1. Inicializace třídy `figures.MultipleFigures`, jež se stará o proces generování a verifikace. V rámci inicializace se vytvoří prázdný podkladový obrázek o dané velikosti, identifikační hash kód a složka pro uložení verifikačních obrázků.
2. Z databáze získáme nejdříve určitý počet náhodných obrázkových kategorií (v našem případě 2) a ke každé kategorii několik náhodných obrázků (4). Jedna z kategorií se označí jako řešení.
3. Jednotlivé obrázky se budou postupně přidávat do podkladového obrázku, přičemž jsou na tento obrázek postupně aplikovány následující transformace:

    3.1. náhodná rotace,

[31] https://github.com/praekelt/django-recaptcha/
[32] http://code.google.com/p/recaptcha/
[33] http://www.mybb.com/
[34] http://commons.wikimedia.org/
[35] http://images.google.com/

3.2. náhodné zvětšení/zmenšení velikosti (v určitých mezích),

3.3. náhodné zvolení pozice na obrázku.

4. Po aplikaci transformací se zjistí počet pixelů, jež tvoří průnik s aktuálním obrázkem. Pokud průnik tvoří 10% až 70% nově přidávaného obrázku, je tento obrázek přidán do aktuálního (vložením do pozadí za aktuální obrázek). Zároveň je vytvořen a uložen obrázek pro verifikaci. V opačném případě je třeba znovu zopakovat celý transformační postup, dokud nebude překryv v daném intervalu.
5. Výsledný obrázek se uloží do adresáře, ze kterého bude později získán pro zobrazení u uživatele.
6. Posledním krokem je uložení údajů o SegCAPTCHA testu do databáze.

Tento postup generování v sobě skrývá určitá úskalí. Především cyklus, v němž získáme náhodnou pozici a testujeme překryv nemusí potenciálně nikdy skončit, neboť zde používáme náhodné hodnoty. Pro lepší efektivitu by se pozice neměla určovat na základě náhody, ale měla by být určena v jednom kroku.

Se současnou metodou generování trvá jedno generování cca 2, pokud je omezen počet cyklů hledání správné velikosti překryvu na 100.

Ukázku výsledných vygenerovaných obrázků můžete vidět na obrázku 3.5.

(a) (b) (c)

Obrázek 3.5: Ukázka vygenerovaných obrázků pro SegCAPTCHA test. Na každém z nich se nalézají 4 jablka a 4 rajčata.

3.5.2 Verifikace

Při verifikaci se opět nejdříve kontrolují požadované hodnoty podle API 2.1.2. Postup verifikace SegCAPTCHA testu probíhá následovně.

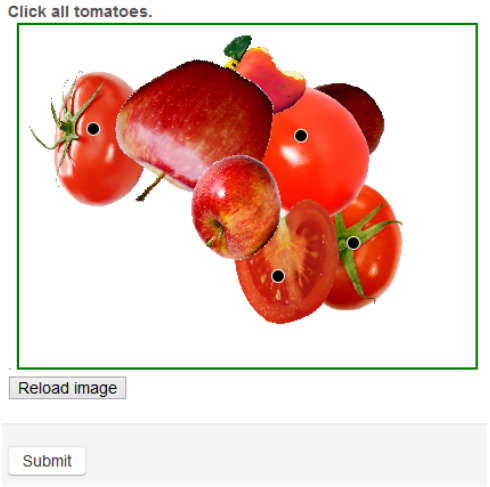
1. Získání SegCAPTCHA testu z databáze podle zadaného hash kódu.
2. Kontrola, zda již neuplynul vymezený čas pro řešení testu (odpověď *TIMEOUT*).
3. Zkontrolujeme platnost přijatých souřadnic bodů, označujících řešení.
4. Postupně procházíme verifikační obrázky a zjišťujeme, zda se některý ze zadaných bodů nachází uvnitř neprázdné oblasti verifikačního obrázku. Pro každý bod musí být nalezen přesně jeden obrázek, aby bylo možno řešení označit jako úspěšné. Pokud zůstane nějaký verifikační obrázek bez přiřazeného bodu, řešení je neplatné. V případě úspěchu se odešle odpověď *PASSED*, v opačném případě *FAILED*.

V příloze E.1 můžete vidět, jak jednotlivé verifikační obrázky vypadají.

3.5.3 Rozhraní

Pro vytvoření rozhraní SegCAPTCHA testu bylo použito stejně jako u RoCAPTCHA testu CSS, JavaScript a HTML. Kromě těchto technologií zde pro zobrazení a řešení testu byl použit HTML5 canvas, neboť ten zjednodušuje technologickou stránku procesu řešení. Konkrétní ukázku experimentálního rozhraní můžete vidět na obrázku 3.6. Uživatel řeší SegCAPTCHA test tak, že klikne s pomocí myši (levé tlačítko) a nebo prstu (u dotykové obrazovky) na místa, jež představují jednotlivé objekty, jež má uživatel za úkol najít. V případě obrázku 3.6 má uživatel za úkol označit všechny rajčata, takže řešením je jeden klik/dotek pro každé rajče na obrázku. Uživatel může označit kteroukoli viditelnou část objektu.

Rozhraní také obsahuje tlačitko „Reload image“ pro načtení jiného obrázku, pokud by si uživatel u současného nebyl jist.



Obrázek 3.6: Ukázka experimentálního rozhraní SegCAPTCHA testu. V tomto případě představuje řešení označení všech rajčat (aktuálně označeno jedno ze správných řešení).

3.6 Uživatelské testování RoCAPTCHA testu

Účelem uživatelského testování bylo zjistit vhodnost implementovaného řešení z hlediska použitelnosti lidmi. Měřil se především čas, za který je možno test vyřešit a odchylka od správného řešení. Na základě těchto údajů bude poté možno zhodnotit současné řešení a případně navrhnout další úpravy.

V sekci 3.6.1 je blíže popsán systém testování a v následující sekci 3.6.2 vyhodnocení výsledků testování.

3.6.1 Systém testování

Pro uživatelské testování posloužila především webová aplikace rocaptcha.com, na jejích úvodní stránce byla zobrazena funkční ukázka RoCAPTCHA testu. Pro získání více dat posloužily i další webové aplikace, kde byla RoCAPTCHA zabudována.

Obsahem testu bylo otočení jednoho náhodně vygenerovaného obrázku ze sady 25738 obrázků, z čehož bylo 949 obrázků získáno přes API Flickru a 24789 z testovací sady projektu Asirra (původně z webových stránek PetFinder.com). Povolená odchylka úhlu řešení od správné polohy obrázku byla nastavena u všech testů na 15° na každou stranu, celkově tedy 30°. To dává výslednou pravděpodobnost pro náhodné uhádnutí 8,3%, což není z hlediska bezpečnosti vhodné nastavení. Nicméně jak již bylo řečeno, cílem testování bylo především ověřit vhodnost řešení z hlediska použitelnosti lidmi.

V rámci jednoho testu se ukládaly následující údaje:

- datum a čas vykonání,
- uživatelova IP adresa,
- úhel otočení vygenerovaného obrázku,
- zdrojová webová aplikace,
- odchylka úhlu otočení od správného řešení,
- čas řešení,
- odchylka od vzpřímené polohy.

Čas řešení byl měřen od okamžiku vygenerování RoCAPTCHA testu do okamžiku ověření správnosti řešení. Tento čas tedy bude vždy nadsazený, neboť se do něj počítá i doba pro realizaci HTTP požadavků a doba, jež uživatel strávil porozumění způsobu řešení RoCAPTCHA testu. Není zde ošetřen ani případ, kdy uživatel mohl vyřešit test až po delší době po jeho zobrazení. Uživatelé však často řešili více těchto testů po sobě, což umožňuje výsledný čas řešení zpřesnit vyloučením příliš velkých hodnot.

3.6.2 Vyhodnocení

Samotné testování proběhlo od 9.4. do 30.4.2013 a za tuto dobu bylo získáno celkově 3380 statistik řešení. Naprostá většina řešení byla získána skrze ukázku RoCAPTCHA testu, zveřejněnou na testovací webové stránce rocaptcha.com[36] (3332), zbytek byl získán přes registrační formulář webu Draci.info[37] (27), kontaktní formulář na webu autora této práce[38] (18) a návštěvní knihu Světu Sahelu[39] (3). Podle statistik IP adres se tohoto testování zůčastnilo 223 uživatelů, průměrně každý vyřešil cca 15 testů.

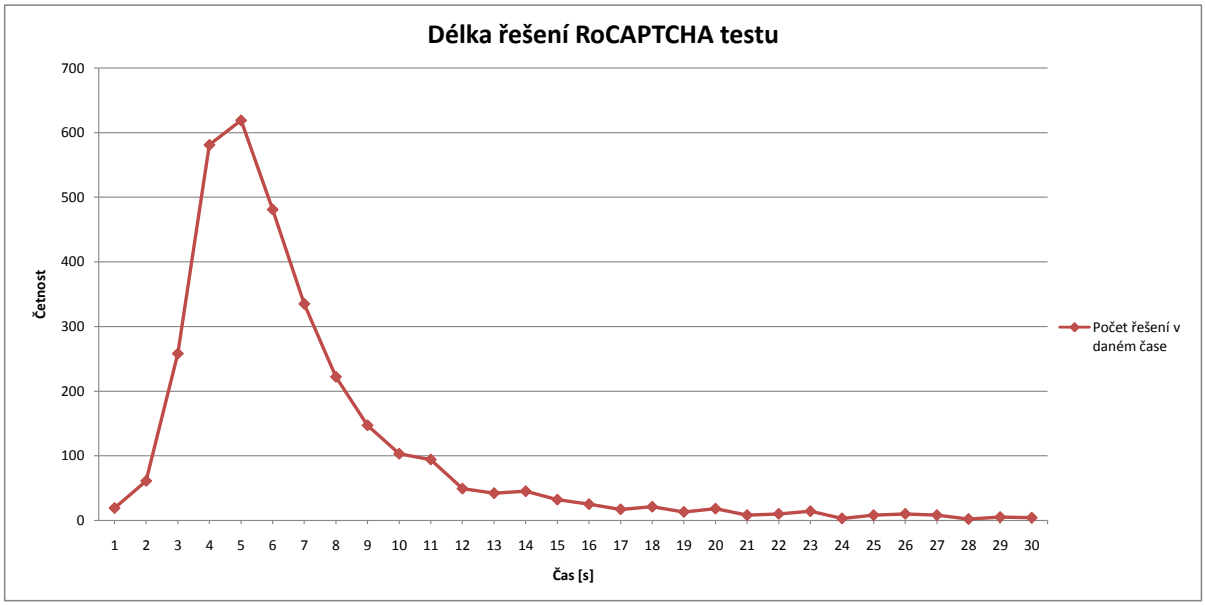
Ačkoli uživatelská spokojenost s RoCAPTCHA testem nebyla exaktně měřena, v průběhu uživatelského testování bylo zaregistrováno poměrně hodně kladných reakcí a minimum negativních. Z toho lze vyvodit, že uživatelé tento test preferují před standardně používaným rozpoznáváním zdeformovaného textu, což dokazuje i provedená studie [12].

Délka řešení RoCAPTCHA testu

Z grafu 3.7 lze vidět, že většina uživatelů byla schopna vyřešit RoCAPTCHA test v čase okolo 5s. Průměrný čas řešení $t_{avg} = 11s$, ovšem je třeba vzít v potaz nepřesnou metodiku měření času, neboť doba řešení byla měřena od času vygenerování po čas ověření správnosti řešení, přičemž uživatelé mohli

[36] http://rocaptcha.com
[37] http://draci.info/index.php?cls=registrace
[38] http://vojtechoram.cz/kontakt/
[39] http://svet-sahelu.cz/guestbook

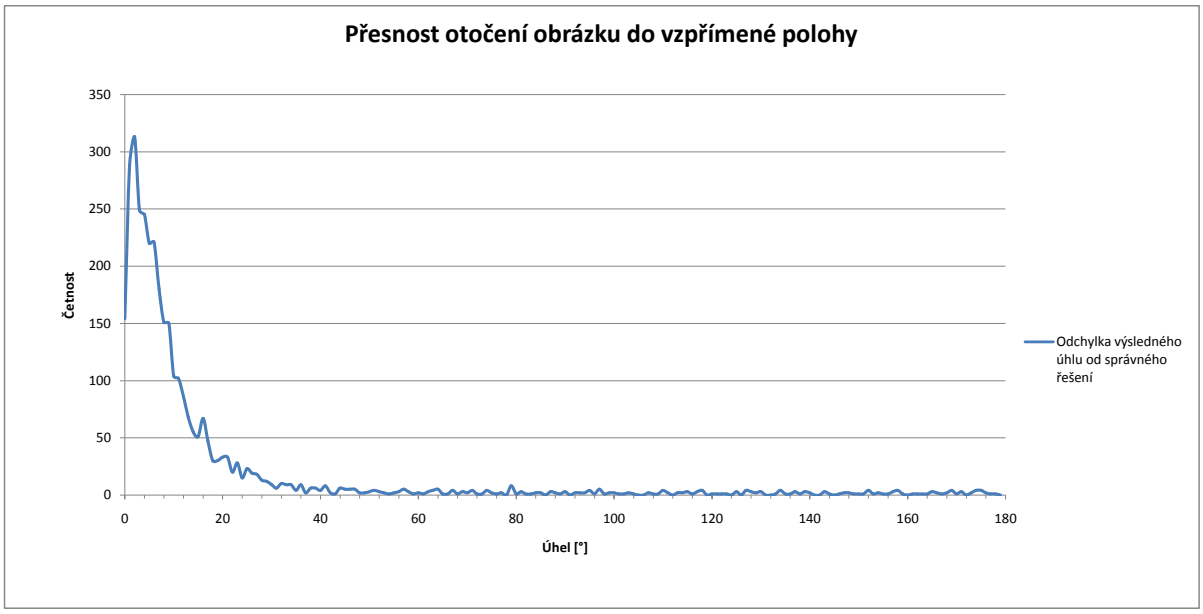
během řešení testu provádět i jiné činnosti. Získaná data vykazují také poměrně velkou variabilitu díky velké směrodatné odchylce $t_{smodch} = 38,4$ a rozptylu $t_{disp} = 1475,54$. Mnohem větší vypovídající hodnotu o době provádění RoCAPTCHA testu proto má medián $t_{med} = 5,25s$. Dále bylo zjištěno, že 90% uživatelů bylo schopno vyřešit tento test v čase $t_{0.9} = 13,65s$.



Obrázek 3.7: Graf četností délek řešení RoCAPTCHA testu.

Přesnost řešení RoCAPTCHA testu

Při otáčení obrázků do správné polohy uživatelé ve většině případů netrefí polohu úplně správně, ale vyskytuje se zde určitá odchylka. Účelem tohoto experimentu bylo zjistit, jako velká tato odchylka je, tedy jak přesně jsou uživatelé schopni odhadnout vzpřímenou polohu obrázku.

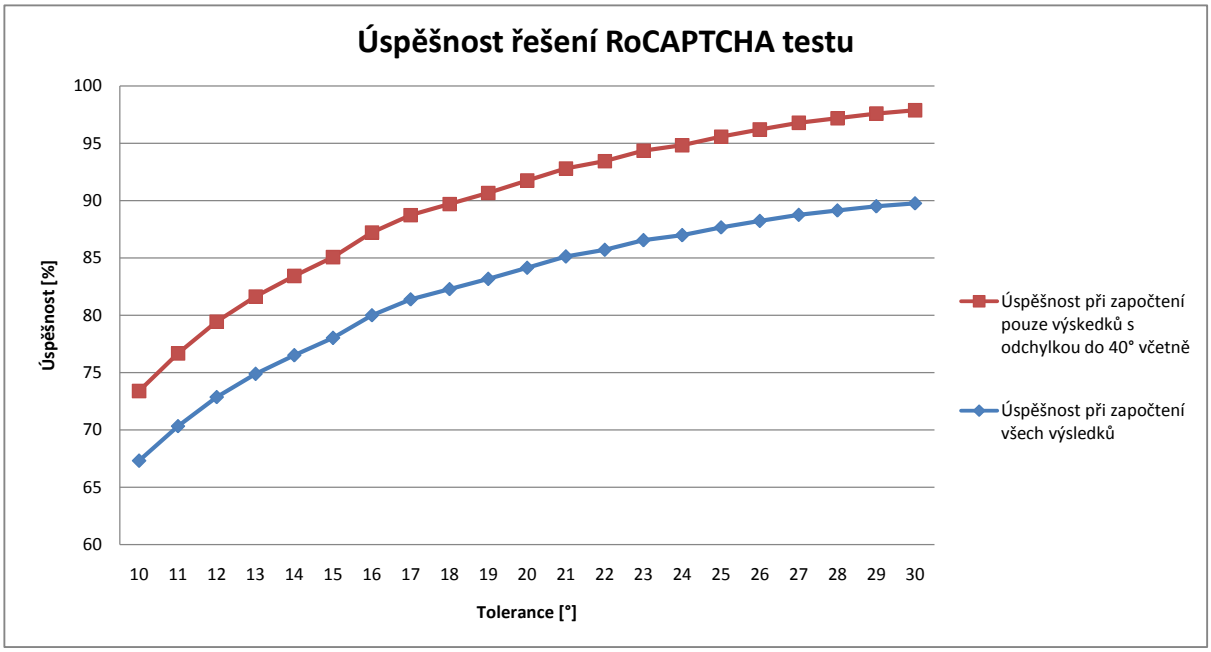


Obrázek 3.8: Graf četností odchylek úhlů od vzpřímené polohy.

Jak lze vidět na grafu 3.8, nejčastěji uživatelů vyřešili RoCAPTCHA test s odchylkou 2° (313 uživatelů). Pokud bereme v potaz všechny výsledky, můžeme odečíst průměrnou odchylku $\omega_{avg} = 15,8°$ a medián $\omega_{med} = 6,5°$. Nicméně je třeba brát v potaz fakt, že některé testy mohli uživatelé odeslat k ověření nevyřešené jen pro otestování, jestli RoCAPTCHA test funguje. Tyto pokusy jsou zachyceny i na grafu v oblasti úhlu cca 40° až 180°. Pokud bychom tedy brali v úvahu jen řešení, kde se uživatelé potenciálně snažili uspět (ořízneme všechny odchylky nad 40°), vyjde nám průměr $\omega_{avg40} = 8,09°$ a medián $\omega_{med40} = 6°$. Stejně jako u délky řešení zde platí, že medián má větší vypovídající hodnotu než průměr, neboť hodnoty nejsou stejnoměrně rozloženy.

Úspěšnost řešení RoCAPTCHA testu

Při samotném testování se u každého testu používala odchylka 15° na každou stranu (30° celkem) pro určení, zda byl uživatel v řešení úspěšný. S tímto nastavením uživatelé dosáhli celkově 78,01% úspěšnosti. Opět je ale třeba brát v potaz testy, jež nebyly vyřešeny správně záměrně. Pokud bereme v potaz pouze výsledné odchylky v toleranci 40° na každou stranu, dostaneme úspěšnost 85,06%.



Obrázek 3.9: Graf úspěšnosti otáčení obrázků do vzpřímené polohy.

Na grafu 3.9 můžeme vidět, jaký vliv má tolerance odchylky od správného řešení na úspěšnost uživatelů. Lehce zjistíme, že 90% uživatelů vyřešilo RoCAPTCHA test s odchylkou 19° (pokud bereme v potaz jen testy, kde se uživatelé potenciálně stažili test vyřešit). Při dalším vývoji by bylo nejlepší pracovat s odchylkou 19° (38° celkově na obě strany), neboť v těchto mezích je schopna vyřešit RoCAPTCHA test většina uživatelů.

Porovnání zjištěných údajů s podobnými CAPTCHA testy

V tabulce 3.2 můžeme vidět srovnání výsledků RoCAPTCHA testu s jinými CAPTCHA testy, založenými na rotaci. Pro lepší porovnání zde byla aproximována i verze RoCAPTCHA testu se 2 obrázky, jež se

pravděpodobností náhodného uhádnutí více blíží k ostatním porovnávaným testům.

Co se týče délky trvání testu, RoCAPTCHA patří při 2 obrázcích a tedy podobné hladině pravděpodobnosti náhodného uhádnutí k nejlepším, ovšem jen díky aplikaci medoty Token Bucket [16]. Bez použití této metody by se musely použít 4 obrázky, aby byla pravděpodobnost náhodného uhádnutí na stejné hladině. Při lepší metodice měření by mohl být výsledný čas řešení ještě o něco lepší.

Tabulka 3.2: Porovnání RoCAPTCHA testu s podobnými CAPTCHA testy (částečně převzato z [18]).

Název testu	**Doba řešení**	**Úspěšnost uživatelů**	**Náhodné hádání**	**Způsob řešení**
RoCAPTCHA	5,25s pro 1 obrázek, cca 10,5s pro 2 obrázky	85%	1,9% pro 1 obrázek a 0,014% pro 2 obrázky (tolerance 15°, Token Bucket)	posunem myši/prstem
What's Up CAPTCHA	nezjištěno	84% pro 3 obrázky	4,44% pro 1 obrázek a 0,009% pro 3 obrázky (tolerance 8°)	slider/posunem myši/klávesy nahoru a dolů
Orientation based image CAPTCHA	14s pro 12 obrázků	92%	50% pro 1 obrázek a 0,02% pro 12 obrázků	jeden klik myši pro 1 obrázek, více pro set
Sketcha	35s pro 10 obrázků	88%	25% pro 1 obrázek a 0,001% pro 8 obrázků	více kliků myši
Multiple SEIMCHA	pod 3,8s pro 1 obrázek, 26s pro 8 obrázků	92%	17,5% pro 1 obrázek a 0,009% pro 8 obrázků se strategií téměř správné odpovědi	jeden klik myši pro 1 obrázek, více pro set

V úspěšnosti dopadá RoCAPTCHA s 85% v porovnání s ostatními podprůměrně. Pro zlepšení úspěšnosti by bylo potřeba více protřídit zdrojové obrázky, a to především ty, kde pro uživatele není snadné rozpoznat správnou polohu. Při větším používání této služby se počítá s automatickým vytříděním obrázků, jež přesáhnout hranici počtu neúspěchů, což by mělo výsledné úspěšnosti pomoci.

Při útoku náhodným hádáním by s jedním obrázkem RoCAPTCHA test patřil k nejlepším, opět zde vydatně pomohl Token Bucket [16]. Při jednom obrázku bez aplikace techniky Token Bucket by pravděpodobnost byla pouze $p = 30/360 \doteq 0,083$, tedy 8,3%. S touto technikou stačí pro přiblížení se k obecně bezpečné hladině pravděpodobnosti test s pouze 2 obrázky. Navíc je zde ještě prostor pro manipulaci s tolerancí odchylky od správného řešení.

Kromě metrik, zachycených v tabulce 3.2 bychom neměli zanedbat také množství prostoru, jež CAPTCHA test zabírá na stránce webové aplikace. Při velké ploše testu nemusí být provozovatelé těchto aplikací ochotni vůbec se tento test implementovat, neboť by jim příliš ubíral z plochy, vyhrazené pro obsah. Na obrázku 1.3 lze vidět, že Sketcha a Orientation based image CAPTCHA zabírají poměrně velké množství prostoru. U Multiple SEIMCHA testu lze předpokládat, že by plocha pro zobrazení byla podobná, neboť při jejím testování bylo použito 8 obrázků [18]. What's Up CAPTCHA naproti tomu

zabírá na stránce poměrně malou plochu, neboť pro jeden test potřebuje pouze 3 obrázky. RoCAPTCHA test má v podstatě stejné rozložení, neboť s What's Up CAPTCHA testem sdílí princip, nicméně s použitím metody Token Bucket stačí pouze dva obrázky. Navíc se obejde i bez slideru, jež je poměrně redundantní.

Pokud bereme v potaz všechny dostupné metriky, RoCAPTCHA se ukazuje jako rovnocenný soupeř v porovnání s ostatními CAPTCHA testy, založenými na rotaci. Ač sdílí princip s What's Up CAPTCHA testem, se zvolením lepších zdrojových obrázků a s aplikací metody Token Bucket poskytuje minimálně stejnou úspěšnost řešení uživateli a lepší úroveň zabezpečení vůči útoku náhodným hádáním.

Závěr

Cílem této diplomové práce bylo vytvořit CAPTCHA test, jež nebude založen na konvenčním způsobu ověření člověka na základě rozpoznání zdeformovaného textu. Na základě provedené rešerše současných 14 CAPTCHA testů byly navrženy a blíže prozkoumány dvě kandidátní řešení s pracovním názvem RoCAPTCHA a SegCAPTCHA.

RoCAPTCHA test je založený na otáčení obrázků do vzpřímené polohy, kombinující v sobě nejlepší vlastnosti CAPTCHA testů Asirra [16] a What's Up CAPTCHA [12]. Toto řešení bylo implementováno a poskytnuto k bezplatnému používání skrze jeho API. V rámci této práce byly rovněž vytvořeny pluginy pro Django a PHP. Uživatelské testování při použití jednoho obrázku ukázalo, že většina uživatelů vyřeší tento test v čase okolo 5s. Průměrná odchylka od správné polohy představuje 8° (medián 6°), přičemž 90% uživatelů je schopno vyřešit tento test do odchylky 19°. Uživatelé dosahují sice pouze 85% úspěšnosti při řešení, nicméně se zavedením lepší metodiky měření a automatickým vyřazením špatně orientovatelných obrázků by mohla být úspěšnost i vyšší. Pro lepší zabezpečení vůči útoku hrubou silou by bylo vhodné použít minimálně dva obrázky v jednom testu. Potenciál RoCAPTCHA testu pokrývá i možnost použití pro reklamu a vylepšení bezpečnosti s použitím sémantických informací („Otočte do vzpřímené polohy obrázek kočky, ne psa.“).

Druhé zkoumané řešení představuje SegCAPTCHA, jež je založená na segmentaci. Toto řešení bylo rovněž implementováno včetně API, nicméně kvůli technickým problémům (náročné generování, obtížné získávání transparentních obrázků) zatím nebylo poskytnuto k používání veřejnosti. Prokázalo poměrně velký potenciál, neboť na rozdíl od RoCAPTCHA testu je možné generovat potenciálně nekonečné množství variant z omezené množiny zdrojových dat a navíc tento princip není v žádném autorovi známém CAPTCHA testu tímto způsobem využit.

V rámci této práce byla vytvořena webová aplikace `http://rocaptcha.com`, jež poskytuje veřejně dostupné API pro RoCAPTCHA test, funkční ukázku tohoto testu, pluginy a dokumentaci.

Přínos mé diplomové práce vidím především v prozkoumání a vytvoření dalších alternativ CAPTCHA testu, jež by potenciálně mohly být uživatelsky přívětivější, než rozpoznávání zdeformovaného textu. Navíc jsem se zdokonalil v práci použitými technologiemi a seznámil s technikami rozpoznávání obrazu, což plánuji využít i v budoucnu.

Literatura

[1] Bigham, J. P.; Cavender, A. C.: Evaluating existing audio CAPTCHAs and an interface optimized for non-visual use. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 2009, s. 1829–1838.

[2] Bushell, D.: In Search Of The Perfect CAPTCHA. [online], Březen 2011, [cit. 2013-05-07].
URL http://coding.smashingmagazine.com/2011/03/04/in-search-of-the-perfect-captcha/

[3] Chanamolu, Y.: *An orientation based image CAPTCHA*. Dizertační práce, Ohio State University, 2009.

[4] Chellapilla, K.; Larson, K.; Simard, P. Y.; aj.: Building segmentation based human-friendly human interaction proofs (HIPs). In *Human Interactive Proofs*, Springer, 2005, s. 1–26.

[5] Chew, M.; Tygar, J. D.: *Image recognition captchas*. Springer, 2004.

[6] Clark Pope, K. K.: Is It Human or Computer? Defending E-Commerce with Captchas. *IT Professional*, ročník 05, 2005.
URL http://www.wiphala.net/research/proposal/papers/is_it_human_or_computer._defending_e-commerce_with_captchas.pdf

[7] Edwards, J.: Beyond CAPTCHA: No Bots Allowed! [online], Březen 2008, [cit. 2013-04-13].
URL http://www.sitepoint.com/captcha-problems-alternatives/

[8] El Ahmad, A. S.; Yan, J.; Tayara, M.: *The Robustness of Google CAPTCHA's*. Computing Science, Newcastle University, 2011.
URL http://www.cs.ncl.ac.uk/publications/trs/papers/1278.pdf

[9] Faessler, F.: minteye captcha - PoC cracking. Youtube, Leden 2013.
URL http://www.youtube.com/watch?v=u0M7gmS5Eg0

[10] Gao, H.; Liu, H.; Yao, D.; aj.: An audio CAPTCHA to distinguish humans from computers. In *Electronic Commerce and Security (ISECS), 2010 Third International Symposium on*, IEEE, 2010, s. 265–269.

[11] Gao, H.; Yao, D.; Liu, H.; aj.: A novel image based CAPTCHA using jigsaw puzzle. In *Computational Science and Engineering (CSE), 2010 IEEE 13th International Conference on*, IEEE, 2010, s. 351–356.

[12] Gossweiler, R.; Kamvar, M.; Baluja, S.: What's up CAPTCHA?: a CAPTCHA based on image orientation. In *Proceedings of the 18th international conference on World wide web*, ACM, 2009, s. 841–850.

[13] Hamarneh, G.; Althoff, K.; Abu-Gharbieh, R.: Automatic line detection. *Project Report for the Computer Vision Course Lund, Simon Fraser University*, 1999.

[14] Hill, S.: Cracking the AreYouAHuman Captcha. [online], Květen 2012, [cit. 2013-04-05].
URL http://spamtech.co.uk/software/bots/cracking-the-areyouhuman-captcha/

[15] Intellect: Automatické rozpoznání Key CAPTCHA testu. [online], Únor 2012, [cit. 2013-03-20].
URL http://intsystem.org/448/vzlom-keycaptcha/

[16] Jeremy Elson, J. H., John R. Douceur; Saul, J.: Asirra: A CAPTCHA that Exploits Interest-Aligned Manual Image Categorization. In *Proceedings of 14th ACM Conference on Computer and Communications Security (CCS)*, Association for Computing Machinery, Inc., 10 2007.
URL http://research.microsoft.com/apps/pubs/?id=74609

[17] Luo, J.; Crandall, D.; Singhal, A.; aj.: Psychophysical study of image orientation perception. In *Electronic Imaging 2003*, International Society for Optics and Photonics, 2003, s. 364–377.

[18] Mehrnejad, M.; Bafghi, A. G.; Harati, A.; aj.: Multiple SEIMCHA: Multiple semantic image CAPTCHA. In *Internet Technology and Secured Transactions (ICITST), 2011 International Conference for*, IEEE, 2011, s. 196–201.

[19] Ministerstvo vnitra České Republiky: *Jak postupovat při plnění povinností vyplývajících ze zákona č. 365/2000 Sb., o informačních systémech veřejné správy a o změněnékterých dalších zákonů, ve znění pozdějších předpisův oblasti informačních systémů veřejné správy*.
URL http://www.mvcr.cz/soubor/metodicke-pokyny-jak-postupovat-pri-plneni-povinnosti-vyplyvajicich-ze-zakona-c-365-2000-sb.aspx

[20] Minkler, J.: Hacking Microsoft ASIRRA Captchas with Google! [online], Duben 2012, [cit. 2013-04-07].
URL http://lazywebdeveloper.blogspot.cz/2012/04/hacking-microsoft-asirra-captchas-with.html

[21] Nanglae, N.; Bhattarakosol, P.: A Study of Human Bio-detection Function under Text-Based CAPTCHA System. In *Computer and Information Science (ICIS), 2012 IEEE/ACIS 11th International Conference on*, IEEE, 2012, s. 139–144.

[22] RolexStrider: Řešení MintEye CaAPTCHA testu v 31 řádcích java kódu, bez otevření obrázku. [online], Leden 2013, [cit. 2013-03-23].
URL http://habrahabr.ru/post/167359/

[23] Ross, S. A.; Halderman, J. A.; Finkelstein, A.: Sketcha: a CAPTCHA based on Line Drawings of 3D Models. In *Proceedings of the 19th international conference on World wide web*, ACM, 2010, s. 821–830.

[24] Sauer, G.; Lazar, J.; Hochheiser, H.; aj.: HIPUU: a Universally Usable Approach to Defeating Automated Bots.

[25] Truong, H. D.; Turner, C. F.; Zou, C. C.: iCAPTCHA: the next generation of CAPTCHA designed to defend against 3rd party human attacks. In *Communications (ICC), 2011 IEEE International Conference on*, IEEE, 2011, s. 1–6.

[26] Vailaya, A.; Zhang, H.; Yang, C.; aj.: Automatic image orientation detection. *Image Processing, IEEE Transactions on*, ročník 11, č. 7, 2002: s. 746–755.

[27] Von Ahn, L.; Blum, M.; Langford, J.: Telling humans and computers apart automatically. *Communications of the ACM*, ročník 47, č. 2, 2004: s. 56–60.
URL http://pcfly.info/doc/Computers/14.pdf

[28] Wang, L.; Liu, X.; Xia, L.; aj.: Image orientation detection with integrated human perception cues (or which way is up). In *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, ročník 2, IEEE, 2003, s. II–539.

[29] Wikipedia: Hough transform — Wikipedia, The Free Encyclopedia. [online], 2013, [cit. 2013-04-28].
URL
http://en.wikipedia.org/w/index.php?title=Hough_transform&oldid=541909233

[30] Wismer, A. J.; Madathil, K. C.; Koikkara, R.; aj.: Evaluating the usability of CAPTCHAs on a mobile device with voice and touch input. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, ročník 56, SAGE Publications, 2012, s. 1228–1232.

[31] Yang, M.-H.; Kriegman, D. J.; Ahuja, N.: Detecting faces in images: A survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, ročník 24, č. 1, 2002: s. 34–58.

Seznam příloh

A Ukázka vzhledu aplikace **70**

B ER diagramy **71**

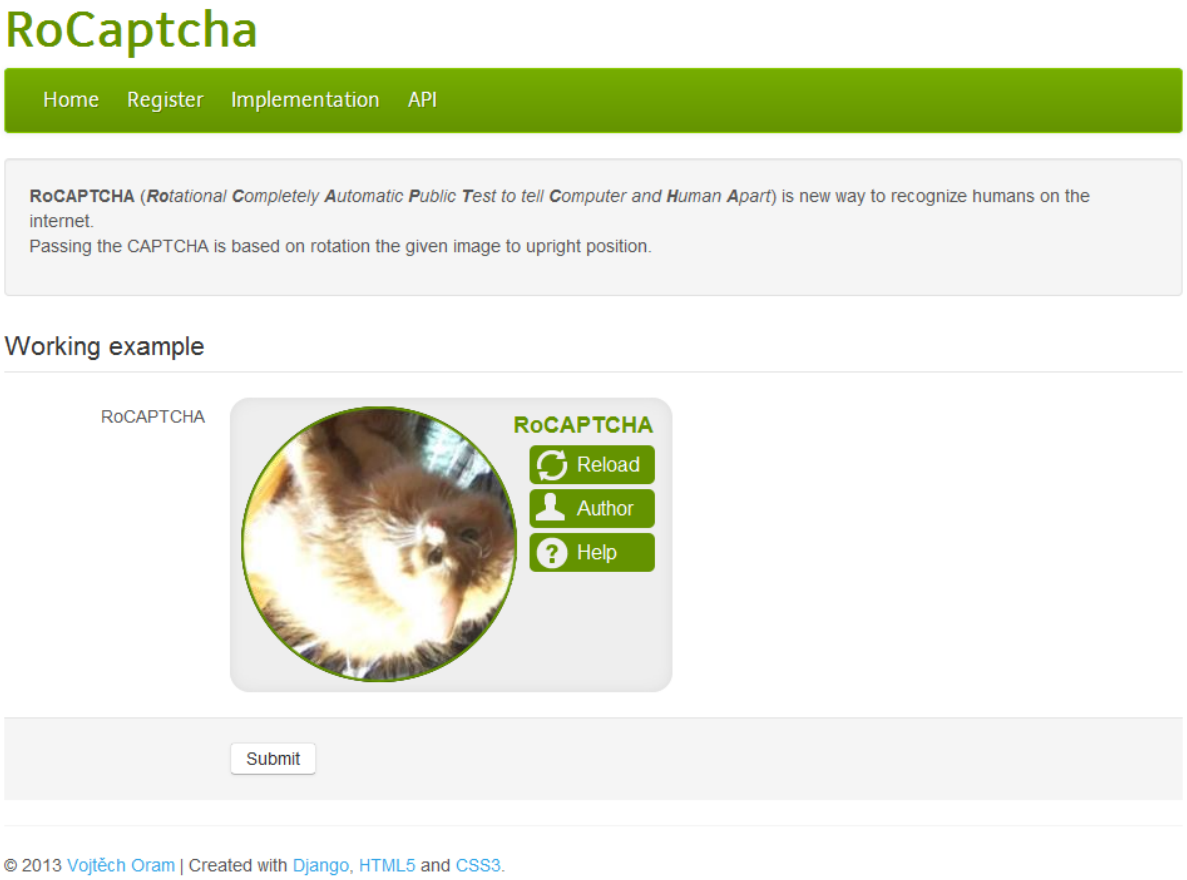
C Diagramy aktivit API **73**

D Stavy a statusy API **75**

E Verifikace SegCAPTCHA testu **76**

Příloha A

Ukázka vzhledu aplikace

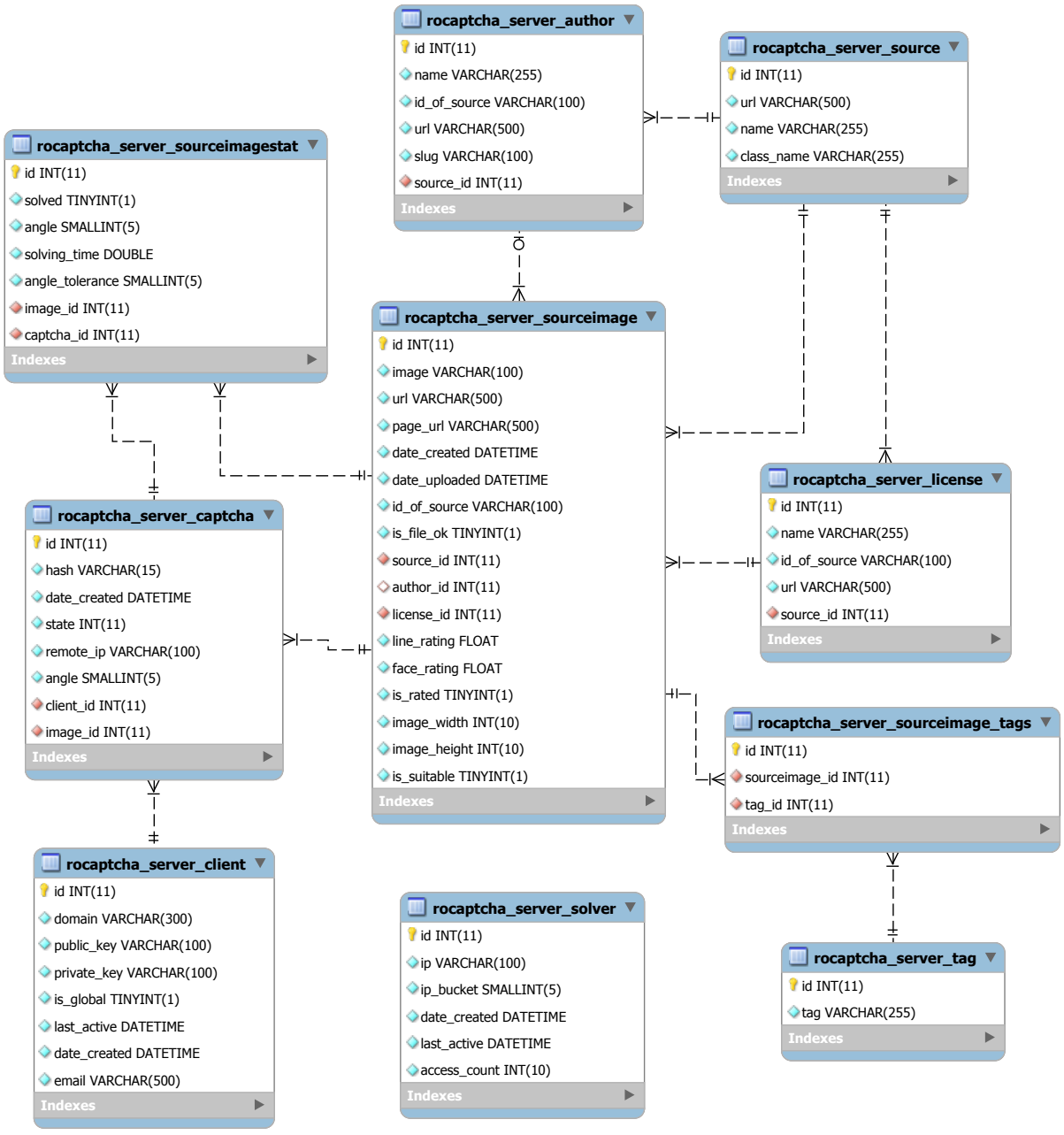


Obrázek A.1: Ukázka vzhledu vytvořené webové aplikace.

Příloha B

ER diagramy

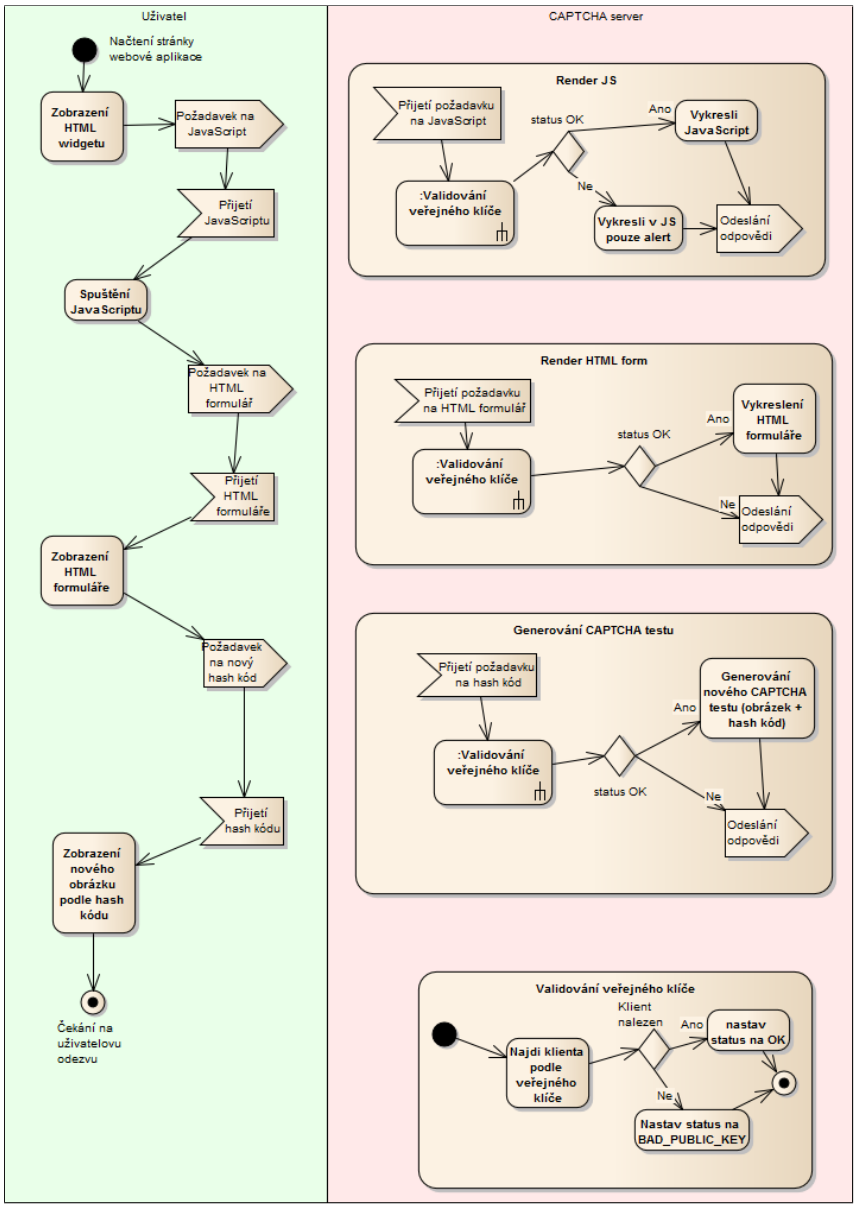
Obrázek B.1: ER diagram SegCAPTCHA testu. Nejsou zde zobrazeny tabulky, jež se generují automaticky pro každý Django projekt.



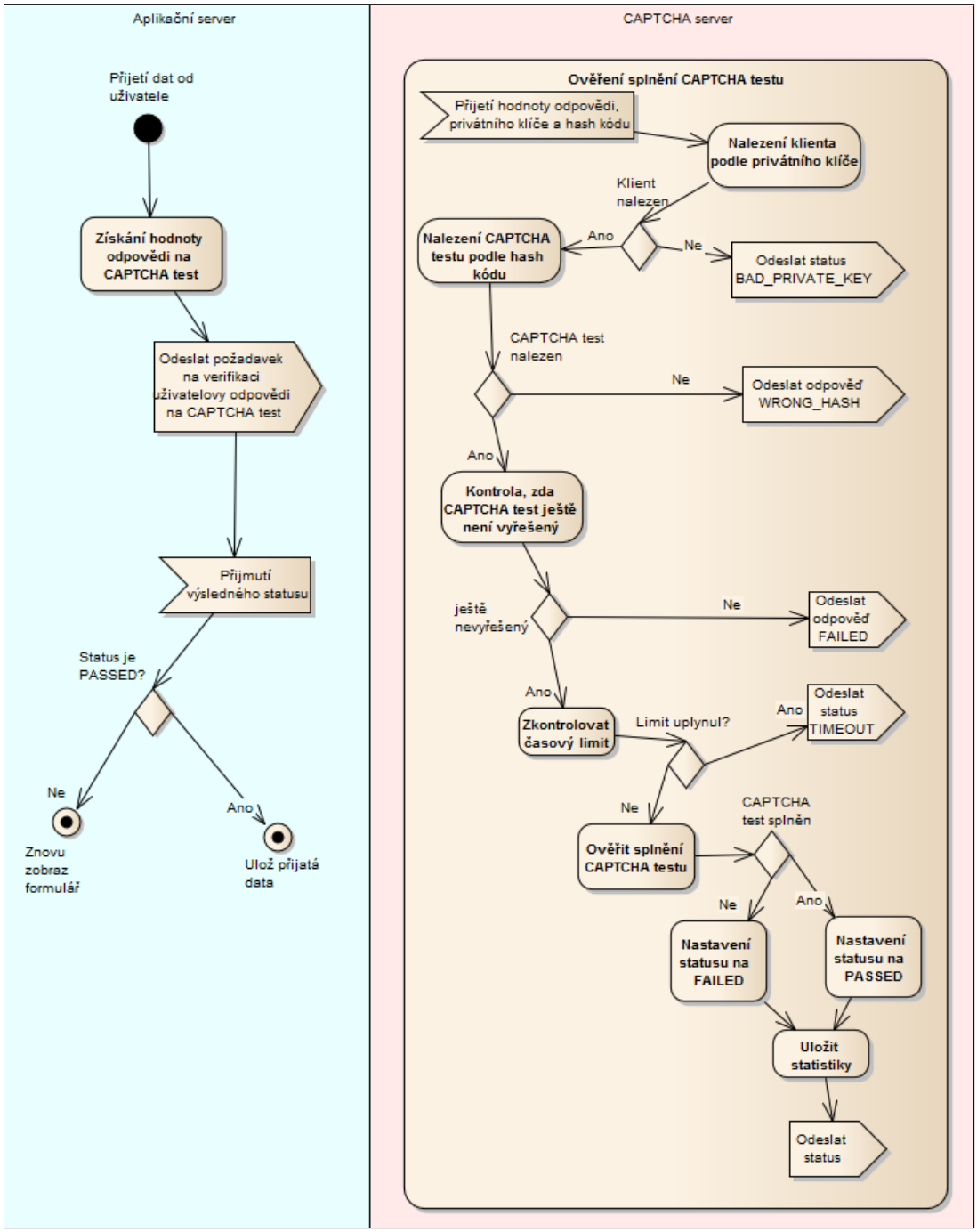
Obrázek B.2: ER diagram RoCAPTCHA testu. Nejsou zde zobrazeny tabulky, jež se generují automaticky pro každý Django projekt.

Příloha C

Diagramy aktivit API



Obrázek C.1: Diagram aktivit, modelující proces zobrazení CAPTCHA testu u klienta.



Obrázek C.2: Diagram aktivit, modelující proces verifikace řešení CAPTCHA testu.

Příloha D

Stavy a statusy API

Tabulka D.1: Statusové kódy, jež mohou být vráceny CAPTCHA serverem v odpovědi.

OK	Vše proběhlo v pořádku.
ERROR	Chyba na straně serveru.
PASSED	CAPTCHA test byl úspěšně vyřešen.
FAILED	CAPTCHA test nebyl úspěšně vyřešen.
BAD_PUBLIC_KEY	Špatný veřejný klíč.
BAD_PRIVATE_KEY	Špatný privátní klíč.
TIMEOUT	Čas pro řešení CAPTCHA testu již uplynul.
WRONG_RESPONSE	Špatná hodnota odpovědi.
WRONG_SESSION	Nezadaná hodnota klíče sezení (session).
WRONG_HASH	Špatný identifikační hash kód.

Tabulka D.2: Stavy, jež může CAPTCHA test nabývat.

SOLVING	Tento test se právě nachází v průběhu řešení.
PASSED	CAPTCHA test byl úspěšně vyřešen.
FAILED	CAPTCHA test nebyl vyřešen úspěšně.
ABORTED	Tento test byl v průběhu řešení zrušen a místo něj byl načten jiný.
BLOCKED	Při verifikaci byl test označen za neplatný (ve vědru sezení uživatele nebyl žádný token).

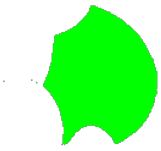
Příloha E

Verifikace SegCAPTCHA testu

(a)

(b)

(c)



(d)

(e)

Obrázek E.1: Ukázka verifikačních obrázků. Obrázek E.1a je původní obrázek, ostatní představují masky pro jednotlivé viditelné části rajčat na původním obrázku.