

Zabezpečení obsahu internetových stránek

Securing Web Content

Petr Knap

Bakalářská práce
2012



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2011/2012

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Petr KNAP**
Osobní číslo: **A09125**
Studijní program: **B 3902 Inženýrská informatika**
Studijní obor: **Informační a řídicí technologie**

Téma práce: **Zabezpečení obsahu internetových stránek**

Zásady pro vypracování:

1. Vypracujte literární rešerši na téma zabezpečení digitálního obsahu na webových stránkách.
2. Provedte analýzu současných metod, užívaných k zabezpečení dat před nežádoucími uživateli, útočníky a roboty.
3. Navrhněte skript nebo aplikaci, která by umožnila zabezpečit pouze malou část obsahu webové stránky nezávisle na zbytku obsahu této stránky.
4. Provedte porovnání zabezpečení obsahu pomocí aplikace z předešlého bodu a běžně užívaného přihlašování.
5. Vyhodnoťte dostatečnost současných metod zabezpečení obsahu webových stránek.

Rozsah bakalářské práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. Schneier, B. **Applied Cryptography, Protocols, Algorithms and source code in C..** John Wiley, 1996. ISBN 0-471-11709-9.
2. Wayner, P. **Disappearing cryptography : information hiding : steganography & watermarking.** 2nd ed. Amsterdam : MK/Morgan Kaufmann Publishers, 2002. 413 s. ISBN 1-55860-769-2.
3. Stallings, W. **Cryptography and Network Security-principles and practices..** Prentice Hall, 2005. ISBN 0-13-111502-2.
4. Bishop. M. **Introduction to Computer Security..** Addison Wesley, 2005. ISBN 0-321-24744-2.
5. Stison, D.R. **Cryptography – Theory and Practice. ,** ISBN. Chapman & Hall, 2002. ISBN 1-58488-206-9.
6. Menezes A. J., Oorschot P. C., Vanstone S. A. **Handbook of Applied Cryptography.** <http://www.math.uwaterloo.ca/hac>. CRC Press, 2001.

Vedoucí bakalářské práce:

Ing. Roman Šenkeřík, Ph.D.

Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce:

24. února 2012

Termín odevzdání bakalářské práce:

8. června 2012

Ve Zlíně dne 24. února 2012

prof. Ing. Vladimír Vašek, CSc.
děkan



prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

ABSTRAKT

Tato bakalářská práce se zabývá zabezpečením obsahu umístěného na webových stránkách. Obsah bakalářské práce je rozdělen na dvě hlavní části. První, teoretická část přináší obecné informace o možnostech zabezpečení obsahu webových stránek. Zvláštní pozornost je věnována běžně používaným metodám, které využívají převážně steganografie, kryptografie a autentifikace. Druhá, praktická část je věnována implementaci vlastního řešení, které přináší další vrstvu zabezpečení, která může být nasazena nad nezabezpečené i zabezpečené webové stránky. Tato implementace, z důvodu možnosti nasazení pro výuku, využívá pouze triviálních kryptografických metod. Zdrojový kód je navržen a napsán tak, aby jej bylo snadné pochopit a v případě potřeby v budoucnu doplnit o další možnosti.

Klíčová slova: webové stránky, zabezpečení, steganografie, kryptografie, Internet, autentifikace uživatele

ABSTRACT

This bachelor thesis deals with the securing content of the web pages. Content of this thesis is divided into two parts. The first, theoretic part presents general information about the possibilities of content security. This part mainly deals with methods commonly used for securing content, like steganography, cryptography and user authentication. The second, practical part contains my own implementation. This implementation can be used as the independent security layer as easy as possible. The implementation uses a simple cipher, because only then can be used as learning material. Source code of this implementation is designed for easy modification by anyone else in the future.

Keywords: web pages, security, steganography, cryptography, Internet, user authentication

The art of war teaches us to rely not on the likelihood of the enemy's not coming, but on our own readiness to receive him; not on the chance of his not attacking, but rather on the fact that we have made our position unassailable.

THE ART OF WAR, SUN TZU

Tímto bych chtěl poděkovat vedoucímu mé bakalářské práce, panu Romanu Šenkeříkovi, za poskytnuté cenné rady a ochotu zaštitit bakalářskou práci pod sebe. Poděkování dále patří panu Michalu Valáškovvi, jehož odborné rady a cenné zkušenosti, jenž mi předal, umožnily vzniknout této práci v takové podobě, v jaké si ji níže můžete přečíst.

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....

podpis diplomanta

OBSAH

ÚVOD.....	9
I TEORETICKÁ ČÁST.....	10
1 ÚVOD DO PROBLEMATIKY ZABEZPEČENÍ OBSAHU WEBU	11
1.1 PROČ OBSAH ZABEZPEČIT?	11
1.2 PROTI KOMU OBSAH ZABEZPEČIT?.....	11
1.2.1 SPAM roboti	11
1.2.2 Nepovolané osoby	11
1.2.3 Vyhledávače	12
1.3 PROČ JE VÝHODNÉ ZABEZPEČIT POUZE MALOU ČÁST OBSAHU A NE VŠE?	12
1.3.1 Dopad zabezpečení veškerého obsahu	12
1.3.2 Výhody zabezpečení malé části webové stránky	12
2 NEJPOUŽÍVANĚJŠÍ METODY ZABEZPEČENÍ OBSAHU WEBOVÝCH STRÁNEK	14
2.1 STEGANOGRAFIE	14
2.2 CAPTCHA	15
2.3 VODOZNAK	16
2.3.1 Viditelný vodoznak	16
2.3.2 Neviditelný vodoznak	17
3 PŘIHLAŠOVÁNÍ.....	18
3.1 CO JE A JAK FUNGUJE PŘIHLAŠOVÁNÍ?	18
3.2 VÍCESTUPŇOVÉ PŘIHLÁŠENÍ	18
3.2.1 Náhodné ověřovací kódy.....	19
3.2.2 Ověřovací kódy závislé na parametrech	19
3.3 MOŽNOSTI ZABEZPEČENÍ HESEL PŘI UCHOVÁVÁNÍ V DATABÁZI.....	19
3.3.1 Jednosměrné algoritmy	20
3.3.1.1 MD5	21
3.3.1.2 SHA-1	23
3.3.2 Jednosměrné algoritmy se solí	25
3.3.3 Šifrování.....	25
3.3.3.1 DES	26
4 SSL/TLS NA WEBU	29
4.1 HTTPS.....	29
4.2 RSA.....	29
4.2.1 DSA.....	30
4.3 CERTIFIKÁTY.....	31
4.4 CERTIFIKAČNÍ AUTORITY	31
4.4.1 Důvěryhodné certifikační autority	31
4.4.2 Nedůvěryhodné certifikační autority.....	32
4.5 SLABÉ ČLÁNKY SSL/TLS	32
II PRAKTICKÁ ČÁST	34
5 POUŽITÉ TECHNOLOGIE.....	35

5.1	PHP.....	35
5.2	JAVASCRIPT	35
5.2.1	jQuery.....	35
5.3	SYMETRICKÁ ŠIFRA AUTOKLÍČ	36
5.4	ŠESTNÁCTKOVÁ (HEXADECIMÁLNÍ) SOUSTAVA.....	37
6	ZABEZPEČENÍ VYBRANÝCH ČÁSTÍ WEBOVÉ STRÁNKY.....	38
6.1	REALIZACE.....	38
6.1.1	Aplikace na straně serveru	38
6.1.2	Aplikace na straně klienta	40
6.2	PRAKTICKÉ UKÁZKY POUŽITÍ	41
6.2.1	Soukromá zpráva na webu	41
6.2.2	CAPTCHA	43
6.2.3	Zabezpečená komunikace mezi klientem a serverem	45
	ZÁVĚR	50
	ZÁVĚR V ANGLIČTINĚ.....	51
	SEZNAM POUŽITÉ LITERATURY.....	52
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....	55
	SEZNAM OBRÁZKŮ	56
	SEZNAM TABULEK.....	57

ÚVOD

Internet je dnes všudypřítomnou oporou každé vyspělé civilizace. Na Internetu se tak vedle relativně zbytečných podvodných reklamních sdělení nachází i digitální verze různých důležitých dokumentů. Jinak řečeno, Internet ve velkém začíná nahrazovat klasické listovní služby i telekomunikační sítě. Jeho potenciál je dokonce tak velký, že je schopen zcela nahradit jakékoliv služby poskytující informace (např. noviny) a do velké míry změnit i přístup k obchodování, ať už mluvíme o obchodování na burze cenných papírů, nebo zůstaneme u obyčejného nákupu jídla.

Tento proces sjednocování z podstaty různých služeb vede k nižším nákladům pro jejich provozovatele. Místo několika sítí se buduje, rozšiřuje a spravuje pouze jedna konsolidovaná počítačová síť, Internet. Z tohoto důvodu je Internet stále rozšířenější, avšak Internet sám o sobě není uživatelsky přívětivý. Aby bylo možno Internet dostat mezi lidi, bylo nutno mu poskytnout „tvář“. Touto tváří se v průběhu let stal web.

Přes web, respektive webové stránky, dnes můžeme poslouchat rozhlasové a sledovat televizní vysílání, komunikovat s přáteli, vyplňovat daňová přiznání, spravovat své finance, nakupovat potraviny a spoustu dalších věcí.

Díky tomu se web, potažmo Internet obecně, stává pomyslnou Achillovou patou každé vyspělé civilizace. Nedávná historie dokonce ukázala, že digitální válka na Internetu může mít daleko větší dopady, než válka reálná. Proto jsou v oblasti zabezpečení internetu a webu vyvíjeny stále nové a nové technologie. Většina z nich se opírá o léty ověřené metody, které zpravidla vznikly v období druhé světové války a následně studené války.

Mezi nejdůležitější úkoly zabezpečení webu dnes patří ochrana digitálních informací před krádeží, zneužitím a pozměněním. Jedním z dalších důležitých úkolů je i jednoznačná autentifikace jednotlivce na Internetu. Tyto stěžejní úkoly zabezpečení a jejich realizace v prostředí webu jsou obsahem následující, teoretické části této bakalářské práce.

I. TEORETICKÁ ČÁST

1 ÚVOD DO PROBLEMATIKY ZABEZPEČENÍ OBSAHU WEBU

Tato kapitola je věnována přiblížení základní problematiky zabezpečení obsahu webu. Všechny následující kapitoly se o níže zmíněná fakta opírají.

1.1 Proč obsah zabezpečit?

Důvodů, proč zabezpečit online publikovaný obsah, existuje nepřeberné množství, avšak zpravidla jde o prevenci krádeže díla chráněného autorským zákonem nebo o odepření přístupu k administrátorským funkcím webu. Je celkem nepodstatné, zda zabezpečujeme administrační rozhraní, textové informace, fotografie, videa nebo jakýkoliv jiný obsah. Nikdy nemůžeme obsah dostupný online zabezpečit na 100 %, můžeme pouze velice efektivně ztížit jeho získání, popř. zneužití (jde-li o přístup k administračním funkcím).

1.2 Proti komu obsah zabezpečit?

Nejdůležitější otázkou, kterou bychom si měli zodpovědět jako první, je, proti komu chceme obsah zabezpečit. Od této odpovědi se bude odvíjet druh zabezpečení, který si zvolíme. Různé metody zabezpečení obsahu webových stránek si ukážeme až v následující kapitole nazvané Nejpoužívanější metody zabezpečení obsahu webových stránek na straně 14. Nyní se podíváme na nejčastější odpovědi na otázku, proti komu obsah zabezpečit.

1.2.1 SPAM roboti

SPAM robot je v tomto kontextu chápán jako program, který prohledává webové stránky a snaží se přímo, nebo nepřímo šířit nevyžádaná sdělení. [1] Činní tak buď shromažďováním e-mailových adres z internetových stránek pro pozdější použití, nebo vkládáním nevyžádaných sdělení přímo na stránky skrze obsažené formuláře.

1.2.2 Nepovolané osoby

Nepovolanou osobou je na webových stránkách prakticky každý návštěvník stránek, který je v dané části stránek nežádoucí. Pro lepší pochopení si představme webové stránky nabízející možnost vytvoření a publikování inzerátu. Takovéto webové stránky, podobně jako většina moderních webů, se skládají z několika sekcí. V první řadě nutná veřejná sekce, kam mohou nahlížet naprosto všichni. Dále je přítomna uživatelská sekce, ve které je každý nepřihlášený uživatel nepovolanou osobou. A nakonec se zde nachází administrace, v níž jsou nepovolanými osobami všichni vyjma administrátorů.

1.2.3 Vyhledávače

V dnešní době je nutné obsah chránit i před vyhledávači, převážně fulltextovými (např. Bing), které se na první pohled tváří velice podobně jako SPAM roboti, avšak jejich přítomnost je na rozdíl od SPAM robotů částečně žádoucí. Díky činnosti vyhledávačů (tzv. indexování) je možné najít prakticky jakoukoliv webovou stránku na Internetu. Je proto nezbytně nutné umožnit vyhledávačům přístup k veřejné části webu a současně je nepustit do žádné jiné části.

1.3 Proč je výhodné zabezpečit pouze malou část obsahu a ne vše?

Je výhodnější se nejprve zaměřit na to, co ztratíme zabezpečením veškerého obsahu.

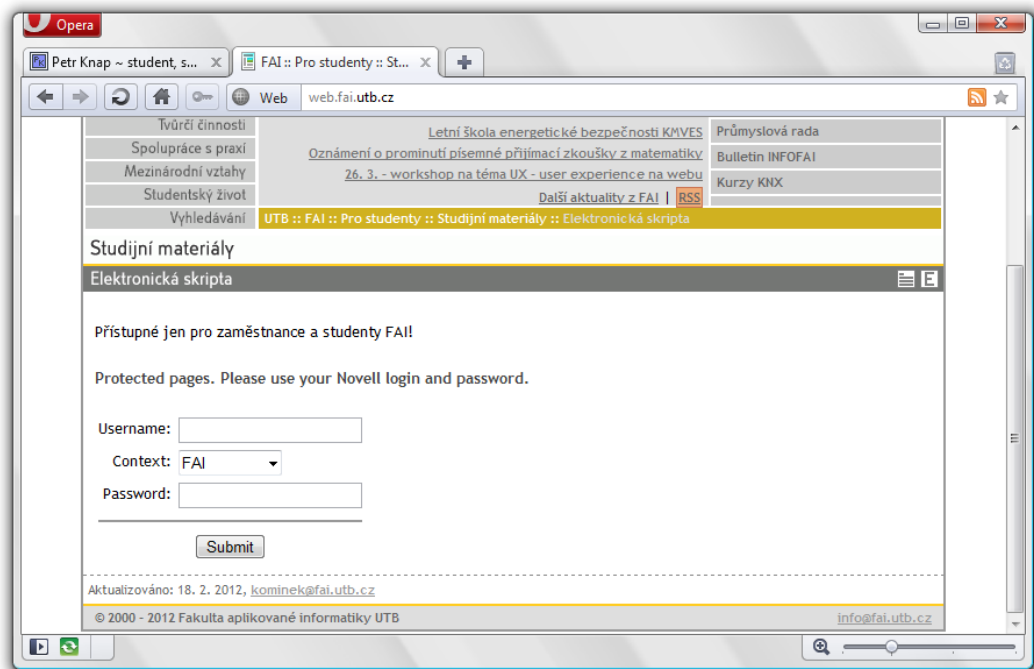
1.3.1 Dopad zabezpečení veškerého obsahu

Není nemožné, ba je dokonce snazší, zabezpečit celý web – pár řádků v konfiguraci webu a zabezpečení pomocí přihlašování je kompletní. Podobným způsobem řeší zabezpečení prakticky všechny domácí routery s webovým administračním rozhraním. Na první pohled jde o jasnou volbu a není třeba vymýšlet nic víc.

Při podrobnějším pohledu ale zjistíme, že tento způsob zabezpečení není pro webové stránky nabízející jakýkoliv obsah vhodný. Aby se o naší službě někdo dozvěděl, museli bychom vytvořit výjimku pro vyhledávače nebo investovat peníze do propagace. Nicméně to by nebyl konec našeho trápení, dále bychom se také museli postarat o to, abychom každému návštěvníkovi zvláště dodali přihlašovací údaje. A ani tehdy bychom neměli klidné spaní, protože bychom umožnili vstup komukoliv, tedy bychom nic nezabezpečili.

1.3.2 Výhody zabezpečení malé části webové stránky

Pokud zabezpečíme pouze malou část webové stránky, potom můžeme návštěvníky rozřadit do několika skupin. Nejčastěji jde o anonymní, registrované a administrátory. Každá tato skupina může následně na webové stránce obdržet jiný obsah, popřípadě může být anonymním uživatelům část obsahu skryta.



Obr. 1 – Elektronická skripta

Přímo skvělou ukázkou výše popsaného jsou weby fakult. Každý (anonymní) návštěvník si může prohlížet drtivou většinu obsahu webu, avšak do jistých sekcí jsou vpuštěni pouze studenti a do administrace smí pouze správci webové stránky.

Pokud bychom na takovou stránku aplikovali plošné zabezpečení pomocí přihlašování, pak by se k jejímu obsahu dostali pouze registrovaní uživatelé, tedy studenti a správci.

2 NEJPOUŽÍVANĚJŠÍ METODY ZABEZPEČENÍ OBSAHU WEBOVÝCH STRÁNEK

V této kapitole si představíme nejpoužívanější metody zabezpečení obsahu webových stránek. Zaměříme se na jejich výhody, nevýhody a použitelnost. Úplně nejpoužívanější metodu zabezpečení obsahu webových stránek, přihlašování, si však pro její rozsah necháme na zvláštní kapitole.

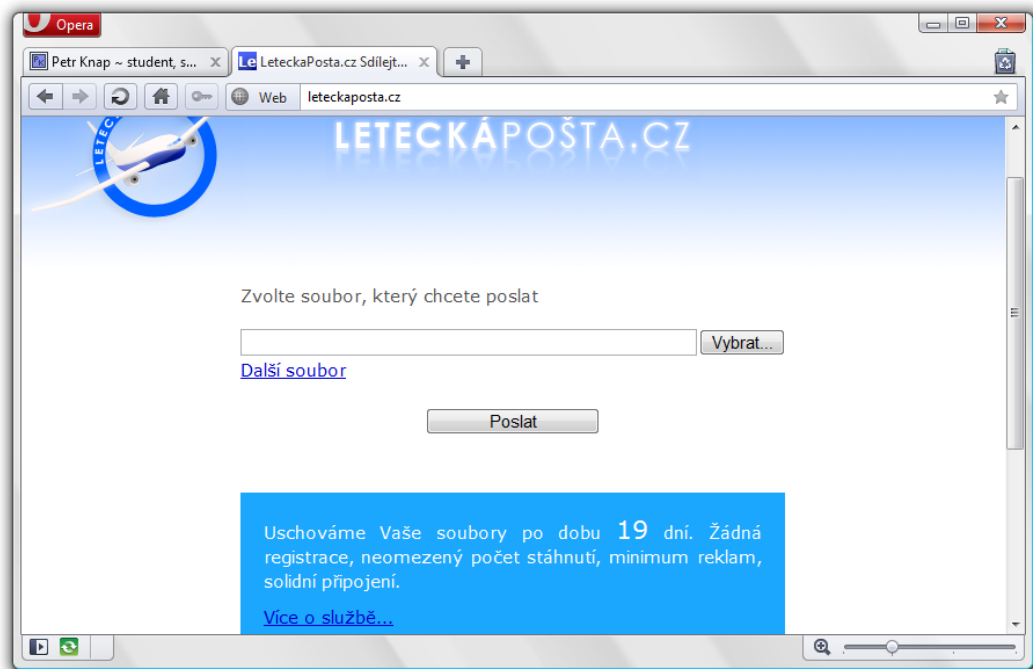
2.1 Steganografie

Slovo „steganós“ pochází z řečtiny a znamená skrytý. Steganografie je umělecký a vědní obor zabývající se psaním skrytých zpráv takovým způsobem, aby se o nich nepovolane osoby nedozvěděly, navzdory tomu, že si je prohlíží. Steganografie ovšem obsah zprávy nijak nezabezpečuje. [2] Proto je považována za opak kryptografie, která se snaží zprávu zabezpečit, ovšem nijak se ji nesnaží skrýt. Z toho důvodu se steganografie a kryptografie používají současně. [3]

Ukázka využití steganografie na webu

Webové služby nejčastěji využívají steganografii v podobě skrytých odkazů. Ty nalezneme například na všech anonymních úložištích souborů. Princip těchto úložišť je následující:

Uživatelka Alice chce poslat uživateli Bob velký soubor, který se nevejde do e-mailu, proto si ve fulltextovém internetovém vyhledávači vyhledá „služba pro zaslání velkých souborů e-mailem“.



Obr. 2 – Webová služba LeteckáPošta.cz

Alice se po chvíli hledání dostala na stránku, kde vidí vstupní pole pro přiložení svého souboru a tlačítko pro jeho nahrání na server. Jakmile soubor přiloží a odešle, služba jí vygeneruje tajný odkaz.

Alice tento odkaz pošle Bobovi, který se na základě znalosti tohoto odkazu dostane k souboru od Alice. Bohužel se k souboru dostane i kdokoliv jiný, kdo odkaz jakýmkoliv způsobem získá či uhodne, proto se vývojáři podobných služeb snaží o generování špatně uhodnutelných odkazů.

2.2 CAPTCHA

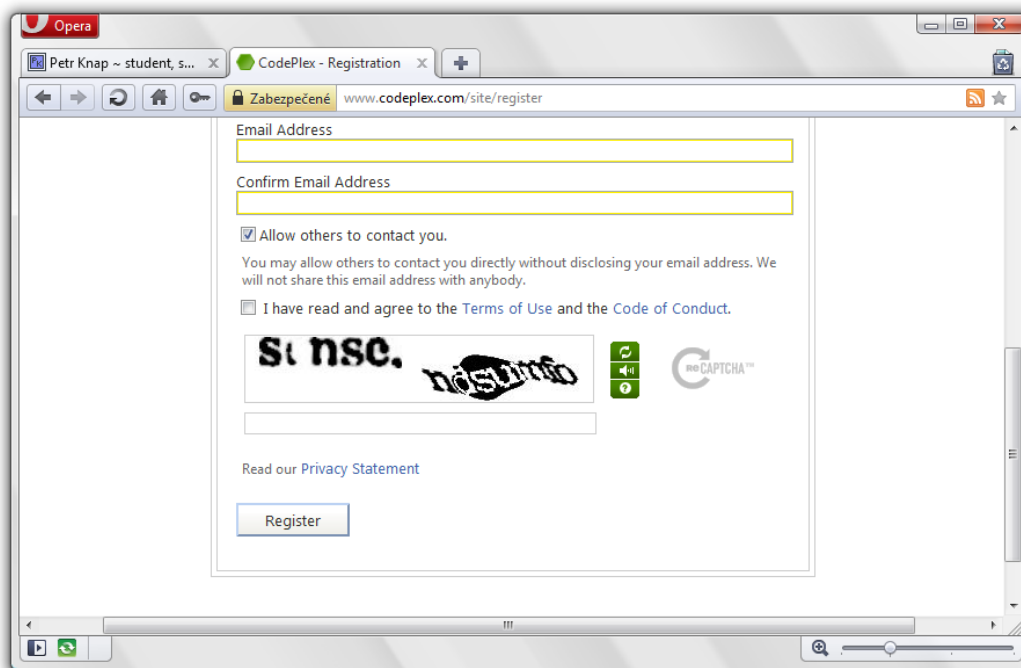
CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) je typ testu, který s dostatečnou přesností dokáže rozhodnout o tom, zda jej vyplnil počítač, nebo člověk. Termín CAPTCHA v roce 2000 vytvořili Luis von Ahn, Manuel Blum, Nicholas J. Hopper a John Langford. [4]

CAPTCHA vyžaduje jeden počítač (server), který se dotazuje, a uživatele, který odpovídá. Ačkoliv je počítač schopný generovat a hodnotit odpovědi na své otázky, není schopen si na ně sám odpovědět. [4] Detekce člověka je proto postavena na předpokladu, že pouze člověk, nikoliv stroj, zvládne správně odpovědět.

Nejpoužívanější typ CAPTCHA vyžaduje, aby uživatel opsal znaky ze zkresleného obrázku. Někdy se pro ztížení vyžaduje například opsání pouze znaků jedné barvy. [4]

Ukázka využití CAPTCHA na webu

CAPTCHA je zpravidla používána při odesílání veřejně dostupných webových formulářů, kde se snaží zabránit robotům (viz SPAM roboti) v uložení dat na server. Nejčastěji lze CAPTCHA najít v otevřených diskusních systémech a při registracích.



Obr. 3 – Registrace na serveru CodePlex.com

2.3 Vodoznak

Vodoznaky umožňují skrytí informací o autorských právech nebo jiných verifikačních informací do digitálních audio, video nebo obrazových souborů. Digitální vodoznak může být před pozorovateli skryt, ale může být i viditelný. [5]

2.3.1 Viditelný vodoznak

Nejúčinnější ochranou vizuálních dat je viditelný vodoznak, který překrývá původní obraz a vytváří tak zcela nová vizuální data. Pozorovateli zpravidla nečiní problém si data překrytá vodoznakem domyslet, ovšem je nemožné je zcela rekonstruovat. Proto jsou viditelné vodoznaky hojně využívány například při publikování fotografií nebo v televizním vysílání. Avšak lze je použít pouze na vizuální data. [5]



Obr. 4 – Vodoznak České televize v levém horním rohu vysílání

2.3.2 Neviditelný vodoznak

Neviditelný vodoznak nijak zásadně neovlivňuje výstupní signál, neboť ovlivňuje jeho nejméně podstatné části. Například je ukryt v nejméně důležitých bitech daného souboru, což je steganografická metoda (viz Steganografie). [5] Ačkoliv je vodoznak neviditelný, lze jej snadno využít ke zjištění původního vlastníka.

Velkou výhodou neviditelných vodoznaků je možnost použití na jakákoliv binární data. Neviditelný vodoznak proto může chránit jakékoliv binární soubory. [5] Bohužel existuje spousta snadných metod, jak tyto vodoznaky z původních dat odstranit. Zpravidla stačí soubor „uložit jako“ – neviditelný vodoznak spolehlivě ustojí pouze binární kopírování.

Informace obsažené v digitálním vodoznaku je možné získat pouze s pomocí specializovaného softwaru. Ten zprávu ze souboru bit po bitu vyextrahuje, překóduje a uživateli ji předá v podobě prostého textu. [5]

3 PŘIHLAŠOVÁNÍ

Přihlašování je kapitola sama o sobě, proto bylo z předešlé kapitoly vytrženo. Prakticky jde o nejznámější a nejrozšířenější způsob ochrany obsahu a autentifikace uživatele na webu v jednom.

3.1 Co je a jak funguje přihlašování?

Velké množství webových služeb používá jako prostředek k ověření identity uživatele uživatelské jméno a heslo. Uživatel se tak při vstupu identifikuje svým uživatelským jménem a svým uživatelským heslem. Tomuto procesu se říká přihlášení. [6]

Výhodou přihlašování je fakt, že je ověření uživatele založeno na pouhé abstraktní znalosti informace (jména a hesla), kterou zná pouze daný uživatel. Přihlášení se k webové službě je proto pro uživatele velice jednoduché a lze jej provést na jakémkoliv místě bez nutnosti s sebou cokoli přenášet. [7]

Nevýhodou přihlašování je, že tajná informace může být snadno zjištěna anebo zapomenuta [7], navíc je zde statická povaha této informace - při každém přihlášení je použito stejné uživatelské jméno i heslo. [6] Citlivější aplikace (např. elektronické bankovníctví) proto k přihlašování přidávají ještě další stupeň ověření, zpravidla s využitím jednorázového kódu zasláného přes SMS na mobilní telefon majitele účtu (viz Vícestupňové přihlášení).

name	pass
Adam	Nagano1998
Bob	bobek
Eva	MojeKockaJeMicka

Tab. 1 – Tabulka s daty pro přihlášení

Implementace samotného přihlášení je velice jednoduchá. Uživatel předloží systému své tajné heslo společně se svou identitou (uživatelským jménem). Systém porovná autentizační údaje s databází, a pokud se údaje shodují s údaji z databáze, je identita uživatele ověřena. [7]

3.2 Vícestupňové přihlášení

Jak jsem již zmínil výše, pro lepší zabezpečení mohou posloužit další stupně ověření uživatele. Nejrozšířenějším z vícestupňových přihlášení je přihlášení dvoustupňové, které

je založené na samotném přihlášení a zadání jednorázového kódu¹. Existují v podstatě dva hlavní způsoby generování kódu: náhodné a závislé na parametrech.

3.2.1 Náhodné ověřovací kódy

Generování náhodného ověřovacího kódu je spuštěno až poté, co se uživateli podaří provést první krok přihlášení. Kód se následně vygeneruje s pomocí generátoru náhodných čísel a uloží do databáze.

V druhé fázi se z databáze vybere vygenerovaný kód společně s kontaktem na uživatele. Skript následně neudělá nic složitějšího, než že kód na tento kontakt zašle. Uživatel tento kód obdrží jiným komunikačním kanálem a zadá jej do systému.

Nejčastěji se můžeme setkat s přenosem kódu s využitím SMS (v případě přihlášení), e-mailu (v případě obnovy hesla či potvrzení účtu) a dopisu (v případě potvrzení adresy). Všechny tyto alternativní komunikační kanály ovšem proces autentifikace zpomalují a jsou proto používány jen ve velmi specifických případech.

3.2.2 Ověřovací kódy závislé na parametrech

Tyto kódy musí být generovány na základě určitých parametrů² a prozatím se masově k přihlašování na webové služby moc nevyužívají. O prozatím poslední vlašťovku se postarala společnost Google, která vyvinula aplikaci Google Authenticator. Ta na základě některých parametrů účtu Google a použitého kapesního zařízení je schopná vygenerovat stejný časově závislý kód, jaký ve stejném čase vygeneruje server. [8]

Výhodou ověřovacích kódů závislých na parametrech je skutečnost, že zde nevzniká nepříjemná prodleva mezi zadáním uživatelského jména, hesla a kódu.

3.3 Možnosti zabezpečení hesel při uchovávání v databázi

Většina autorů webových služeb klade uživatelům na srdce, že jejich hesla musí mít nejméně osm znaků, nesmí být založena na slovníkovém pojmu, musí obsahovat velká písmena, malá písmena, číslice a nejméně jeden nealfanumerický znak a každých šest týdnů si ho musí změnit. [9] Uživatelé proto na oplátku od autorů požadují, aby byla jejich

¹ Jednorázový kód je platný pouze jednou, tzn. pro každý pokus o přihlášení je vygenerován znovu.

² Několik různých strojů musí být schopno na základě stejných parametrů ve stejný čas vygenerovat stejný kód způsobem, který není snadno předpověditelný.

nesmyslně složitá a špatně zapamatovatelná hesla uložena bezpečně, tedy v nečitelné podobě.

name	pass
Adam	c94cd5e89d61c9cca7f436b228be511a18aec618
Bob	7c0a8873c20e6ee5b36d3484d1b58f7f2545b16f
Eva	ec3d3170dc5a31f22cf5cd47b5bbd6e3c934902d

Tab. 2 – Tabulka se zabezpečenými daty

K zabezpečení hesel lze využít velké množství různých metod, nicméně nejčastěji se lze setkat s jednosměrnými algoritmy a v ojedinělých případech je možné zahlédnout i klasické šifrování. [7]

3.3.1 Jednosměrné algoritmy

Ve většině případů přihlašování nepotřebujeme znát přesné heslo uživatele. Potřebujeme mít pouze možnost ověřit si, že heslo je shodné se správným heslem. K tomuto účelu můžeme nasadit a využívat jednosměrné algoritmy. [10] Jejich činnost se nazývá hashování.

Hashování je v podstatě výpočet kontrolního součtu. Z daných dat vždy snadno spočteme stejný kontrolní součet (hash), ovšem neexistuje snadný způsob, jak z tohoto součtu rekonstruovat zpět původní data. [10]

Nejznámější a nejrozšířenější hashovací algoritmy jsou SHA-1 a MD5. Je nutné zmínit, že již na přelomu let 2005 a 2006 proběhla médii zpráva, že tyto algoritmy byly prolomeny. Objevené postupy naštěstí neumožňují rekonstruovat z hashe původní text, umožňují pouze nalezení dvojic dat, která mají stejný kontrolní součet. [10] Na základě takovýchto dvojic je možné se přihlásit bez znalosti hesla uživatele [2], tedy heslo zůstane utajeno.

Celková idea použití hashů pro ukládání hesel je taková, že v databázi uchováváme pouze hashe – kontrolní součty. Ověření hesla pak probíhá tím způsobem, že vypočítáme hash zadané hodnoty a výsledek se musí shodovat s uloženou hodnotou. Pokud se následně naše databáze dostane do nepovolaných rukou, nezíská útočník čitelná hesla, ale pouze jejich hashe, které jsou mu teoreticky k ničemu, protože by je musel prolomit. [10]

Prakticky však zneužít hashe lze. Oproti obecným bezpečnostním zásadám a doporučením uživatelé používají hesla nedostatečné délky. Velice často jde o hesla naležitelná ve slovnících – na Internetu jsou volně ke stažení soubory, které obsahují seznamy obvyklých

slov a hesel a k nim odpovídající hashe. Útočníkovi pak stačí porovnat tyto hashe s databází získanou útokem a slabá hesla odhalí. [10]

3.3.1.1 MD5

MD5 je algoritmus, který na libovolně dlouhý binární vstup vrátí výstup délky 128 bitů, tzv. hash. Je navržen tak, aby na dvě velice podobné zprávy vrátil diametrálně odlišný hash. [11]

MD5 je primárně určen pro podepisování digitálních dat, kde je nutnost, aby podpisy velkých souborů byly „komprimovány“ bezpečným způsobem a následně mohly být zašifrovány asymetrickou šifrou, např. RSA. [11]

MD5 je navržen s ohledem na 32 bitové procesory, kde se řadí mezi nejrychlejší jednosměrné algoritmy vůbec. Kromě toho MD5 nevyžaduje velké substituční tabulky, čímž šetří nejenom další čas, ale i operační paměť. [11]

MD5 je rozšířením dříve používaného MD4. MD5 je lehce pomalejší než MD4, ale je „konzervativnějšího“ návrhu. Důvodem vzniku MD5 byl pocit, že MD4 není dostatečně bezpečný. Návrh MD4 umožňoval extrémně rychlý chod na úkor bezpečnosti, což se mu s nástupem rychlejších procesorů stalo osudným. MD5 byl navržen s opačnými podmínkami – bylo žádoucí malé zpomalení na úkor větší bezpečnosti. [11]

Pohled MD5 na zprávu

MD5 předpokládá, že má n bitovou zprávu jako vstup, a že chce najít hash. n je libovolné nezáporné celé číslo, může to být nula a nemusí jít o násobek čísla osm. Vstupní zprávu si tedy rozdělí na bity označené jako m_0 až m_{n-1} . [11]

Přidání „vycpávkových“ bitů

V prvním kroku je zpráva rozšířena tak, aby její délka (v bitech) byla shodná s 448, při modulu 512 - po celočíselném dělení délky číslem 512 musí zůstat zbytek 64. Rozšíření se provede i v případě, kdy zpráva na začátku tuto podmínku splňuje. [11]

Rozšíření se provádí tak, že se přidá jeden bit hodnoty 1 a následně jsou přidávány bity hodnoty 0, dokud nedojde ke splnění výše popsané podmínky. [11]

Doplnění délky

K výsledku předchozího kroku je připojena 64 bitová reprezentace n . Pokud by došlo k výjimce a bitová reprezentace n by byla delší, pak je použito pouze spodních 64 bitů. Těchto 64 bitů je připojeno v podobě dvou 32 bitových slov. [11]

V tomto okamžiku má výsledná zpráva bitovou délku, která je přesným násobkem čísla 512. Z toho plyne, že je zpráva rozdělena po 16 slovech délky 32 bitů. [11]

Inicializace MD zásobníku

K výpočtu hashe zprávy je použito zásobníku o délce čtyř 32 bitových slov. Tato slova jsou naplněna následujícími hodnotami 01 23 45 67, 89 AB CD EF, FE DC BA 98 a 76 54 32 10, zapsány jsou v šestnáctkové soustavě (viz Šestnáctková (hexadecimální) soustava). [11]

Zpracování zprávy

Za účelem zpracování zprávy jsou definovány čtyři funkce, které ze tří vstupních slov vrátí jedno slovo výstupní. Jedná se o funkce $F(X, Y, Z) = XY + \bar{X}Z$, $G(X, Y, Z) = XZ + Y\bar{Z}$, $H(X, Y, Z) = X \oplus Y \oplus Z$ a $I(X, Y, Z) = Y \oplus (X + \bar{Z})$. Povšimněte si, že funkce H vrací paritu svých vstupů. [11]

Nyní je zapotřebí vygenerovat vektor T délky 64 prvků. Jeho prvky jsou $T[i] = 4294967296 \cdot |\sin i|$, kde i je v radiánech. Po vygenerování tohoto vektoru již stačí zpracovat každých 16 slov jako jeden blok. [11] Postup zpracování jednotlivých bloků je k nalezení v RFC 1321³ na stranách 4 a 5.

Výstup

Výstupem MD5 jsou čtyři slova o délkách 32 bitů, tzn. hash má celkem délku 128 bitů bez ohledu na délku vstupu. [11] Základní vlastnost výstupu je velká citlivost na malou změnu vstupu, tedy změna jednoho libovolného bitu na vstupu zajistí změnu na výstupu nejméně v jednom bitu.

³ RFC 1321 je veřejně k dispozici na adrese <http://tools.ietf.org/html/rfc1321>

3.3.1.2 SHA-1

SHA-1 je součástí SHS společně s SHA-224, SHA-256, SHA-384 a SHA-512. Všechny tyto algoritmy slouží k výpočtu hash ze zprávy a vrací hashe s celkovou délkou od 160 bitů do 512 bitů. SHA jsou zpravidla používány při digitálním podepisování, generování náhodných čísel a při generování autorizačních tokenů. [12]

SHA nesou v názvu slovo „zabezpečené“, protože pro daný algoritmus je výpočetně náročné zjistit z hashe původní zprávu a najít jinou zprávu, která by měla stejný hash. Důvodem je fakt, že velmi malá změna na vstupu způsobí velmi velkou změnu na výstupu. [12]

Logické funkce

SHA-1 používá řadu logických funkcí f_0, f_1, \dots, f_{79} . Každá z funkcí f_t , kde $0 \leq t < 79$, pracuje se třemi slovy délky 32 bitů (X, Y a Z) a vrací výstup v podobě jednoho 32 bitového slova. Funkce $f_t(X, Y, Z)$ je definována následovně: [12]

$$f_t(X, Y, Z) = \begin{cases} XY \oplus \bar{X}Z & 0 \leq t \leq 19 \\ X \oplus Y \oplus Z & 20 \leq t \leq 39 \\ XY \oplus XZ \oplus YZ & 40 \leq t \leq 59 \\ X \oplus Y \oplus Z & 60 \leq t \leq 79 \end{cases} \quad (1)$$

Konstanty

SHA-1 dále využívá řadu 32 bitových konstant K_0, K_1, \dots, K_{79} , které jsou definovány následovně: [12]

$$K_t = \begin{cases} 5A827999 & 0 \leq t \leq 19 \\ 6ED9EBA1 & 20 \leq t \leq 39 \\ 8F1BBCDC & 40 \leq t \leq 59 \\ CA62C1D6 & 60 \leq t \leq 79 \end{cases}$$

Hodnoty jsou zapsány v šestnáctkové soustavě (viz Šestnáctková (hexadecimální) soustava).

Přidání „vycpávkových“ bitů

Zprávu m je nutno doplnit tak, aby její bitová délka byla beze zbytku dělitelná číslem 512. [12] Přidání bitů je totožné s algoritmem MD5, viz Přidání „vycpávkových“ bitů u MD5.

Zpracování zprávy

Po doplnění zprávy je již možné začít zprávu zpracovávat. Zpráva je rozdělena na n bloků dlouhých 512 bitů, tj. m^0, m^1, \dots, m^{n-1} . Pro zrychlení výpočtu je využito toho, že 512 bitové bloky mohou být rozděleny na 16 32 bitových slov, tato slova značíme $m_0^i, m_1^i, \dots, m_{15}^i$, kde i je číslo bloku. [12]

Inicializace hodnoty hashe

Hash (výstup) je značen h a před začátkem výpočtu je nastaven následovně: [12]

$$h_0^0 = 67452301$$

$$h_1^0 = \text{EFCDAB89}$$

$$h_2^0 = 98BADCFE$$

$$h_3^0 = 10325476$$

$$h_4^0 = \text{C3D2E1F0}$$

Hodnoty jsou zapsány v šestnáctkové soustavě (viz Šestnáctková (hexadecimální) soustava).

Výpočet hashe

Po dokončení předzpracování zprávy dochází k postupnému zpracování všech bloků m^0 až m^{n-1} . [12] Postup zpracování jednotlivých bloků je k nalezení ve FIPS PUB 180-3⁴ na stranách 17 až 20.

Výstup

Výstupem SHA-1 je 5 slov o délce 32 bitů, tzn. hash má celkem 160 bitů bez ohledu na délku vstupu. [12] Základní vlastnost výstupu je velká citlivost na malou změnu vstupu, tedy změna libovolného jednoho bitu na vstupu zajistí změnu na výstupu nejméně v jednom bitu.

⁴ FIPS PUB 180-3 je veřejně k dispozici na adrese http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf

3.3.2 Jednosměrné algoritmy se solí

Z důvodu možnosti praktického zneužití hashe (viz Jednosměrné algoritmy) se obvykle doporučuje používat takzvaný hash se solí. Tato metoda spočívá v tom, že se k heslu přidá ještě nějaký další text, tzv. sůl, a teprve z toho se spočítá hash. Na rozdíl od šifrovacího klíče nevyžaduje sůl zvláštní ochranu ani utajení. Sůl sama o sobě nenesou žádnou informaci, slouží pouze k zamezení možnosti použít předpočítané slovníky. [10]

Existují dva základní způsoby, jimiž lze sůl použít:

1. Hodnota soli je pro všechny uživatele v rámci aplikace stejná. Lze tak efektivně zabránit použití obecných slovníků předpočítaných hesel. Pokud si ale útočník dá tu práci a spočítá si hesla s naší solí, může pak takto vytvořený slovník použít proti všem uživatelským účtům. [10]
2. Sůl je generována zvlášť pro každého uživatele. Náročnost prolomení jednoho konkrétního účtu tím sice zůstane nezměněna, ale případný úspěch nijak nepomůže útočníkovi v útoku na další uživatele. [10]

3.3.3 Šifrování

Nasazení symetrického šifrování⁵ na uživatelská hesla není zcela běžné, neboť je zpravidla pomalejší než jednosměrné algoritmy. [10] Mluvíme-li o webových službách, potom musíme brát v úvahu fakt, že proces nebude na serveru prováděn jednou za čas, ale že bude prováděn neustále v několika tisících vláknech paralelně. Šifrování uživatelských hesel v databázi je proto zpravidla nasazováno ve velice specifických případech.

Tato možnost je jediná, kterou můžeme použít v případě, že námi používaný autentizační systém vyžaduje na serveru znalost hesla v otevřeném tvaru (například Digest Authentication). S její implementací jsou ovšem spojeny značné obtíže, spočívající především v nezbytnosti bezpečně uložit klíč, který k šifrování používáme. Bohužel, v podmínkách běžného komerčního hostingu je bezpečné uložení šifrovacího klíče v podstatě nemožné. [10]

⁵ Symetrické šifrování je takové, které pro šifrování i dešifrování využívá stejný klíč.

3.3.3.1 DES

DES je jedním z relativně rychlých symetrických šifrovacích standardů s širokou softwarovou podporou a je relativně snadné jej nasadit ve webových aplikacích (např. v PHP verze 5 jej můžeme používat skrz API nazvané mcrypt). Navíc je DES využíván i protokolem HTTPS, který bude zmíněn v následující kapitole. Nejprve si však představme samotný DES.

DES ve zkratce

DES se skládá ze dvou základních algoritmů, ty naleznete v dokumentu FIPS PUB 46-3⁶ na stranách 8 až 16. První slouží k šifrování a druhý k dešifrování a navzájem jsou inverzní. [13] Oba algoritmy pracují nad bloky dat délky 64 bitů, na které aplikují klíč délky 56 bitů a navrací data délky 64 bitů. [2] To umožňuje DES aplikovat dvěma směry.

1. Data můžeme prohnat procesem šifrování, následně je jako nesmyslná data přeneseme po komunikačním kanálu a druhá strana je pomocí stejného klíče dešifruje do původní podoby. [13]
2. Data můžeme prohnat procesem dešifrování, následně je jako nesmyslná data přeneseme po komunikačním kanálu a druhá strana je pomocí stejného klíče zašifruje do původní podoby. [13]

Tohoto faktu bylo využito po prolomení v roce 1998, kdy se bezpečnost zvedla tak, že byl zaveden standard využívající trojitý proces. To znamená, že se DES aplikuje na stejná data třikrát s použitím tří různých klíčů. Tato metoda se nazývá Triple DES (značí se TDES nebo 3DES). [2] Pro snazší značení se za název šifry přidává trojice písmen. Jedná se o písmena *E* (šifrování) a *D* (dešifrování) a označují pořadí, v jakém byly operace prováděny.

Např. *3DES EDE* je zkrácený zápis pro následující vztahy otevřeného a šifrovaného textu:

$$CT = DES_{encrypt_{K_1}} \left(DES_{decrypt_{K_2}} \left(DES_{encrypt_{K_3}} (OT) \right) \right) \quad (2)$$

$$OT = DES_{decrypt_{K_3}} \left(DES_{encrypt_{K_2}} \left(DES_{decrypt_{K_1}} (CT) \right) \right) \quad (3)$$

⁶ FIPS PUB 46-3 je veřejně k dispozici na adrese <http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf>

Ačkoliv je Triple DES dostatečně rychlý pro šifrování komunikace, jeho rychlost je již pro nasazení nad interní databázi nedostatečná. Východiskem je nasazení již překonaného DES anebo modernějšího algoritmu. Zvolit můžeme například AES, který ovšem ke svému chodu vyžaduje větší množství paměti. [2]

Silná stránka DES

Jak již bylo zmíněno, DES již byl zlomen, ovšem to neznamena, že je jeho zlomení triviální. DES obsahuje jisté postupy, které se snaží znemožnit inteligentnější prolomení než prolomení hrubou silou.

Podstata DES algoritmu

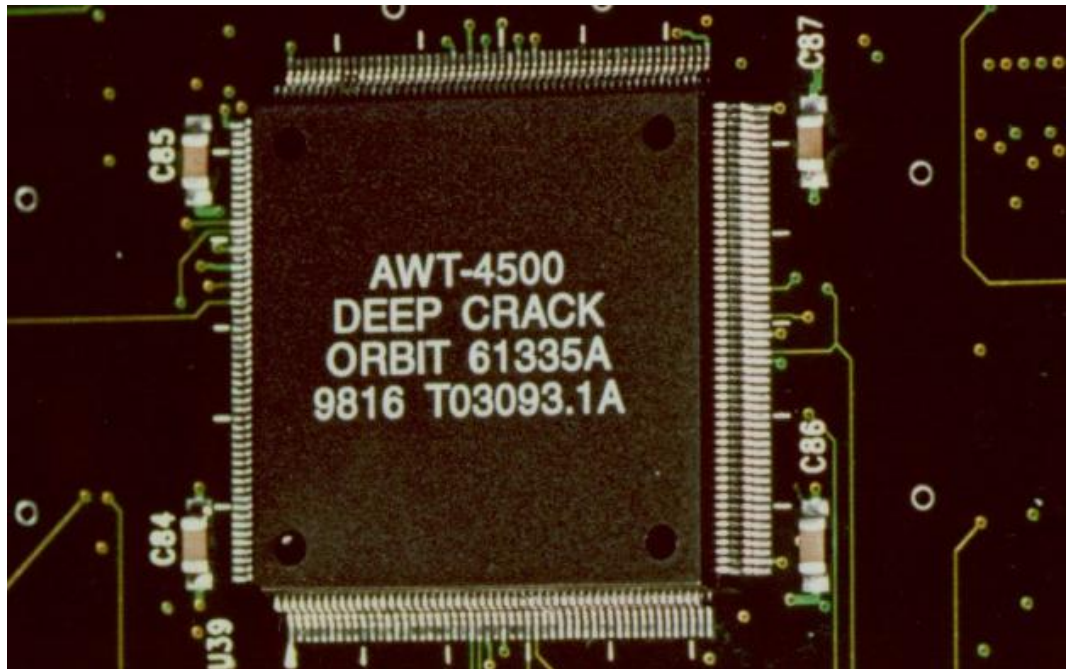
Inteligentnějším útokům brání už samotná podstata DES. Hlavním rysem šifry DES je využívání substitučních tabulek, též zvaných S-boxy. V každé iteraci je použito 8 substitučních tabulek, které nejsou veřejně zjistitelné (generují se). To znemožňuje použití kryptoanalýzy a tím pádem i urychlení prolomení. [2]

Použití klíčů délky 56 bitů

Pro klíče délky 56 bitů existuje 2^{56} možných kombinací, to je zhruba $7,2 \cdot 10^{16}$ klíčů. I kdybychom měli k dispozici stroj, který zvládne provést dešifrování DES za $1 \mu s$, stejně by nám útok hrubou silou trval stovky let. [2]

V roce 1977 pánové Diffie a Hellman postulovali, že výše zmíněné platí pouze pro sériové výpočty. Došli k názoru, že kdybychom měli k dispozici stroj, který zvládne provést dešifrování DES za $1 \mu s$ a zvládne paralelně provádět 1 milion vláken, pak by prolomení trvalo pouhých 10 hodin. Autoři určili cenu takového stroje na 20 000 000 \$. [2]

Reálného prolomení se šifra DES dočkala v roce 1998. Cena potřebného hardwaru do té doby klesla na méně než 250 000 \$. Prolomení provedl stroj nazvaný „DES cracker“ sdružení Electronic Frontier Foundation. [2]



Obr. 5 – Jeden z mnoha výpočetních čipů DES cracker

Odolnost proti časovým útokům

Časové útoky využívají faktu, že některé algoritmy pro různé klíče vyžadují různou dobu zpracování. Na základě reakčního času sledovaného stroje jsou pak schopny odhadnout klíč. DES je proti tomuto druhu útoků odolný, protože délka jeho vykonávání závisí na délce dat, nikoliv na obsahu dat nebo klíče. [2]

4 SSL/TLS NA WEBU

Tato kapitola je věnována zabezpečení komunikace mezi serverem a klientem, protože když se reklamy vychloubají „bezpečnými servery“, většinou mluví o tom, že tyto webové servery umí šifrovat komunikaci [6] s využitím SSL nebo TLS nad protokolem HTTP, tzv. HTTPS.

4.1 HTTPS

Je důležité vědět, že protokol HTTPS chrání pouze komunikaci mezi serverem a klientem. Útočník může snadno napadnout jak server, tak klienta. [6]

Komunikaci HTTPS zahajuje handshake (potřesení rukou) zabezpečený s pomocí asymetrického šifrování⁷ (nejčastěji RSA, viz níže), ve kterém proběhnou tyto kroky: [6]

1. Klient obdrží od serveru certifikát, kterým ověří server (viz Certifikáty). [6]
2. Klient se dohodne se serverem na šifrovacích a hashovacích algoritmech, které budou používat. [6]
3. Oba se domluví na symetrickém šifrovacím klíči. [6]
4. Komunikace je dále již šifrována symetrickou šifrou (např. 3DES). [2]

V průběhu handshake se může klient serveru identifikovat zasláním certifikátu. Po ukončení handshake je komunikace již šifrována s využitím domluvených parametrů a probíhá přes obyčejné HTTP. [6]

4.2 RSA

RSA je masivně používanou asymetrickou šifrou, která, jako každá asymetrická šifra, je značně pomalejší, než symetrické šifry, proto se zpravidla využívá pouze při inicializaci spojení a to převážně k přenosu klíčů pro následující symetrické šifrování. [6]

RSA zprávu chápe jako číslo M , veřejný klíč jako vektor čísel $\{n, e\}$ a privátní klíč jako vektor čísel $\{n, d\}$. [14] Dlouhé bloky dat je nutno rozdělit tak, aby jejich bitová velikost k splňovala nerovnost $2^k < n \leq 2^{k+1}$. [2] Zpráva se následně šifruje i dešifruje blokově.

⁷ Asymetrické šifrování je takové, které k šifrování a dešifrování používá dva různé klíče (tzv. klíčový pár).

Při šifrování i dešifrování se využívá vztahu $M_2 = M_1^{k_2} \bmod k_1$, kde M_1 je vstupní blok zprávy, M_2 je výstupní blok zprávy, k_1 je první část klíče (tj. n), k_2 je druhá část klíče (tj. e , nebo d) a mod je modulo – zbytek po celočíselném dělení. [14] Tento vztah je platný pouze při zachování daného postupu generování klíčového páru ze dvou prvočísel, viz A Method for Obtaining Digital Signatures and Public-Key Cryptosystems⁸ strany 6 až 10.

Příklad: Mějme veřejný klíč {187, 7}, privátní klíč {187, 23} a zprávu 20. Nyní můžeme zprávu zašifrovat:

$$C = M^e \bmod n = 20^7 \bmod 187 = 1\,280\,000\,000 \bmod 187 = 147$$

Pro ověření si zprávu podobným způsobem dešifrujme:

$$\begin{aligned} M &= C^d \bmod n = 147^{23} \bmod 187 \\ &= 70\,517\,790\,972\,121\,409\,091\,230\,735\,429\,871\,287\,686\,482\,512\,769\,723 \bmod 187 \\ &= 20 \end{aligned}$$

Po dešifrování jsme obdrželi původní zprávu, tedy vše proběhlo v pořádku.

4.2.1 DSA

Algoritmus digitálních podpisů (DSA) je určen pro aplikace, které vyžadují ověření identity autora dat. Jedná se tak o digitální podobu podpisu psaného rukou. [15]

Digitální podpis v podstatě není nic víc, než hash (viz Jednosměrné algoritmy) podepisovaných dat, který je zašifrován s pomocí asymetrické šifry, např. RSA. Pro účely digitálních podpisů je však nutno používat veřejný klíč ve smyslu privátního klíče a privátní klíč ve smyslu veřejného klíče. Z toho plyne, že kdokoliv může hash dešifrovat, ale zašifrovat jej může pouze majitel podpisu. [15]

Digitální podepisování probíhá ve dvou krocích:

1. Autor zprávy zprávu nejprve napíše, před uložením/odesláním spočte její hash a tento hash zašifruje svým privátním klíčem, který nikdo nezná, a tento zašifrovaný hash ke zprávě přiloží. [7]

⁸ A Method for Obtaining Digital Signatures and Public-Key Cryptosystems je veřejně k dispozici na adrese <http://people.csail.mit.edu/rivest/Rsapaper.pdf>

2. Příjemce zprávy zprávu přijme včetně zašifrovaného hashu. Hash dešifruje a z obdržené zprávy vypočte vlastní hash. Pokud se dešifrovaný hash shoduje s hashem obdržené zprávy, pak nebyla zpráva nijak upravena a jejím autorem je majitel privátního klíče. [7]

4.3 Certifikáty

Certifikát je digitálně podepsaná datová struktura, která mimo jiné obsahuje jméno vlastníka veřejného klíče a samotný veřejný klíč. [6] Hlavním účelem certifikátu je kryptografické spojení veřejného klíče s identitou daného majitele. [7]

Ověření identity subjektu je založeno na faktu, že je každý veřejný klíč matematicky svázán s privátním klíčem v rámci klíčového páru a každý certifikát musí být podepsán certifikační autoritou. [7]

4.4 Certifikační autority

Certifikační autoritou se automaticky stává každý, kdo digitálně podepíše certifikát. Z tohoto důvodu se certifikační autority rozdělují na důvěryhodné a nedůvěryhodné.

4.4.1 Důvěryhodné certifikační autority

Důvěryhodné certifikační autority jsou hierarchicky uspořádány do stromové struktury, kde je jasně dáno, která certifikační autorita je nadřizená/podřizená které certifikační autoritě. Takovýchto stromů existuje několik a každý uživatel se může rozhodnout, kterému stromu, od jaké úrovně a ve které větvi bude důvěřovat. [7] Pro usnadnění situace obsahuje každý komplexnější operační systém již v základu seznam důvěryhodných certifikačních autorit. Jedná se zpravidla o stromy, které zaštiťují velké nadnárodní korporace a sdružení s dlouholetou tradicí vystavování certifikátů.

Tyto defaultně důvěryhodné certifikační autority si však za podepisování certifikátů účtují jisté poplatky a velice často požadují i ověření identity žadatele – čím výše se certifikační autorita nachází v důvěryhodném stromu, tím je důvěryhodnější a tím větší je poplatek za podpis certifikátu. Takto podepsaný certifikát navíc jednoznačně identifikuje majitele. Pro lepší pochopení si uveďme ukázkou z praktického života:

- Alice: „Ahoj, já jsem Alice, tady je můj občanský průkaz se jménem Alice.“
- Eva: „Ahoj, já jsem Alice, tady je můj občanský průkaz se jménem Eva.“
- Bob: „Ahoj, já jsem Bob a chci mluvit s Alicí.“

Bob si na základě předložených občanských průkazů snadno zkontroluje, kdo je skutečnou Alicí a baví se pouze s ní.

Akreditované certifikační autority

Jednou z podskupin důvěryhodných certifikačních autorit jsou takzvané akreditované certifikační autority. Jedná se o certifikační autority, které prošly akreditačním procesem ze strany státních orgánů, u nás například Česká Pošta. Akreditované certifikační autority proto mají postavení kvalifikované instituce s obecně uznávanou důvěryhodností a využitím především pro orgány státní správy (např. při podepisování digitálních formulářů). [7]

4.4.2 Nedůvěryhodné certifikační autority

Vylučovací metodou dojdeme k jednoznačnému závěru, že každá certifikační autorita, která není důvěryhodná, je automaticky nedůvěryhodná. [7]

Nedůvěryhodné certifikační autority jsou zakládány především za účelem snížení nákladů za podepisování certifikátů. Zejména neziskové organizace a školy využívají ve velkém takzvaných certifikátů podepsaných „sám sebou“ (sami se stávají certifikační autoritou). Tyto certifikáty umožňují kryptograficky naprosto stejné zabezpečení jako certifikáty od důvěryhodných autorit, avšak nejsou podepsány nikým důvěryhodným. Uživatele tak staví do situace, kterou můžeme převést do běžného života následovně:

- Alice: „Ahoj, já jsem Alice.“
- Eva: „Ahoj, já jsem Alice.“
- Bob: „Ahoj, já jsem Bob a chci mluvit s Alicí.“

V takovéto situaci Bob neví, kdo je pravou Alicí a kdo ne. Východiskem je stav, kdy Bob Alici zná a proto jí důvěřuje více než Evě, kterou v životě neviděl. Jinak řečeno Bob si přidal Alici mezi důvěryhodné certifikační autority.

4.5 Slabé články SSL/TLS

Ani SSL/TLS není magickým proutkem, jehož mávnutí by dokonale vyřešilo zabezpečení dat. Avšak jedná se o v současné době nejlepší z masově používaných řešení. [6]

Uživatelé

Pokud uživatel aplikace upozorní, že platnost certifikátu vypršela nebo že je certifikát podepsán nedůvěryhodnou certifikační autoritou, většina uživatelů nebude ničemu z toho rozumět a klikne na „pokračovat“, aby mohli provést to, co zamýšleli. [6] V tomto momentě je celý proces ověřování identity a podepisování certifikátů důvěryhodnými certifikačními autoritami zcela zbytečný.

Certifikační autority

Jak historie ukázala, nejslabším článkem SSL/TLS jsou samotné certifikační autority. Například v roce 2001 se podařilo společnosti VeriSign (jedné z nejznámějších certifikačních autorit) vystavit certifikáty společnosti Microsoft Corporation někomu, kdo s touto firmou neměl vůbec nic společného. [6]

Jednotlivé implementace

U většiny nejpoužívanějších internetových prohlížečů se ještě stále nachází chyby v implementaci SSL/TLS, díky kterým je možné kontrolu certifikátu zcela obejít. S takovými chybami je SSL/TLS neúčinné. [6]

II. PRAKTICKÁ ČÁST

5 POUŽITÉ TECHNOLOGIE

Před představením samotné realizace by bylo vhodné si v této kapitole krátce představit technologie, které byly k realizaci využity a odůvodnit, proč byly použity právě tyto technologie a ne jiné.

5.1 PHP

PHP je široce rozšířený skriptovací jazyk určený pro obecné použití. Je vhodný především pro tvorbu webových stránek a služeb, protože může být vepsán přímo do HTML kódu. Syntaxe jazyka PHP vychází z jazyků C, Java a Perl. Hlavním cílem jazyka PHP je poskytnutí webovým vývojářům možnost tvorby rychlých dynamických webových stránek, avšak PHP toho zvládne mnohem více. [16]

PHP bylo zvoleno právě pro svou dobrou rozšířenost a obecně snadno srozumitelnou syntaxi.

5.2 JavaScript

JavaScript je v současné době jediný použitelný skriptovací jazyk určený pro web na straně klienta. Je využíván prakticky na všech dynamických stránkách a podporován všemi rozšířenými internetovými prohlížeči. Na jinak statické webové stránky přináší JavaScript funkce ryze dynamické (např. kontrolu a předzpracování vstupních dat). [17]

JavaScript byl zvolen, protože se jedná prakticky o jediný jazyk použitelný k těmto účelům.

5.2.1 jQuery

jQuery je rychlá a stručná JavaScriptová knihovna, která zjednodušuje traverzování HTML dokumentu, zpracování událostí a přidává AJAX interface pro rychlejší a pohodlnější vývoj webových stránek a aplikací. jQuery je navrženo tak, aby změnilo cestu, jakou v současné době píšeme JavaScript. [18]

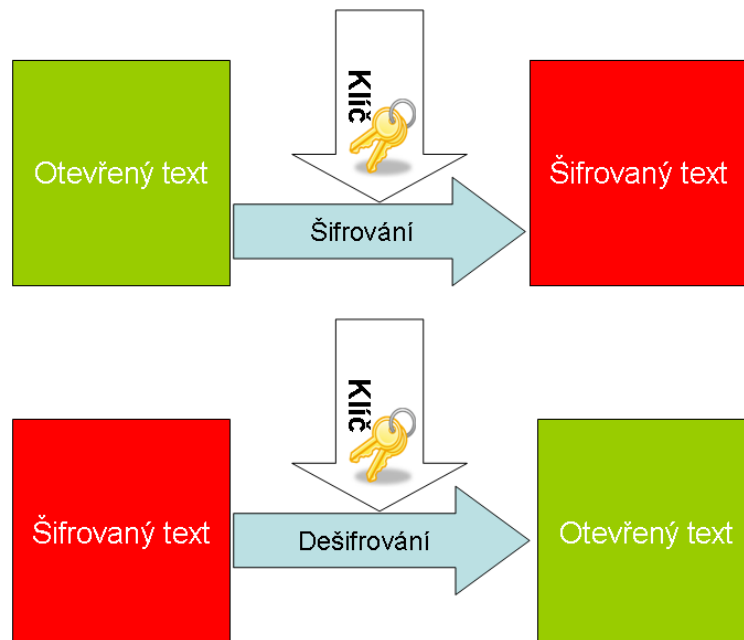
Samotná realizace jQuery nijak nevyužívá, avšak je jej využito ke zjednodušení ukázkových kódů.

5.3 Symetrická šifra Autoklíč

Autoklíč (také známý jako Autoclave) je šifra, která klíč nastavuje vlastní zprávou. [19] Existují dvě základní implementace nastavování klíče:

1. Po dosažení konce klíče jsou postupně přidávány znaky z šifrovaného textu.
2. Po dosažení konce klíče jsou postupně přidávány znaky z otevřeného textu.

Za autora šifry Autoklíč je považován Girolamo Cardano. [19] Tato šifra obsahuje slabinu, která ji umožňuje velice rychle zlomit. Šifru následně přepracoval Blaise de Vigenère. [2] V moderní kryptologii vychází z šifry Autoklíč všechny proudové šifry. [19]



Obr. 6 – Princip symetrické kryptografie

Tato šifra patří mezi monoalfabetické šifry⁹ a to je důvod, proč byla zvolena – každý si může snadno na papíře zkontrolovat, zda šifra funguje správně, a nemělo by být nijak složité pochopit její algoritmus ze zdrojového kódu.

⁹ Monoalfabetické šifry pracují pouze se znaky A až Z a patří k jedněm z nejsnazších na pochopení. [2]

5.4 Šestnáctková (hexadecimální) soustava

Šestnáctková (hexadecimální) soustava je číselná soustava o základu 16 určená k vyjadřování celých čísel. K běžně známým arabským číslicím přidává následující znaky: *A* s hodnotou 10, *B* s hodnotou 11, *C* s hodnotou 12, *D* s hodnotou 13, *E* s hodnotou 14 a *F* s hodnotou 15. Hlavní využití šestnáctkové soustavy je v počítačovém programování. Slouží k základní kompresi zápisu bitů – každé 4 bity mohou být zapsány jedním šestnáctkovým číslem. [20]

Šestnáctková soustava byla vybrána za účelem zkrácení výstupního kódu. Ten by jinak obsahoval až tříznakové skupiny číslic, s použitím šestnáctkové soustavy obsahuje pouze dvouznakové skupiny číslic a písmen.

6 ZABEZPEČENÍ VYBRANÝCH ČÁSTÍ WEBOVÉ STRÁNKY

V této kapitole již je na čase představit mé řešení zabezpečení části obsahu webové stránky. Toto řešení se z důvodu snazšího pochopení zaměří na ochranu pouze textových dat a to na straně klienta, serveru i komunikačního kanálu.

6.1 Realizace

Při realizaci byl kladen důraz na následující věci:

1. Snadnost pochopení řešení.
2. Výpočetní nenáročnost.
3. Využitelnost v rámci výuky.

6.1.1 Aplikace na straně serveru

Na straně serveru je vše realizováno jednou PHP třídou jménem *Crypt*. Třída byla psána tak, aby nebylo nijak složité ji pochopit a rozšířit o jiný šifrovací algoritmus, popřípadě doplnit o kompresní či kódovací algoritmus.

Tato třída poskytuje jednoduché aplikační rozhraní, které dovoluje provádět veškeré úkony nutné k jejímu využívání. Při jejím využívání mějte na paměti, že třída *Crypt* v příložené implementaci pracuje s monoalfabetickou šifrou, tedy využívá pouze spodní část ASCII tabulky¹⁰. Z tohoto důvodu si všechny texty interně převádí na 8 bitovou znakovou sadu.

Inicializace

O inicializaci se stará PHP samotné. Novou instanci třídy *Crypt* vytvoříme pouhým zavoláním konstruktoru.

```
$crypt = new Crypt();
```

Nastavení klíče

Nastavení klíče obstarává magický setter proměnné *SecretKey*. Chceme-li nastavit klíč na hodnotu „TajnyKlicUzivatele“, provedeme to následovně:

```
$crypt->SecretKey = 'TajnyKlicUzivatele';
```

¹⁰ ASCII tabulka je kódová tabulka, která definuje 7 bitové kódy pro znaky anglické abecedy, číslice, speciální znaky a řídicí znaky. [21]

Klíč by měl obsahovat znaky ze spodní poloviny ASCII tabulky. Jiné znaky jsou ignorovány z důvodu jejich nejednoznačného číselného vyjádření.

Nastavení otevřeného textu

Nastavení otevřeného textu obstarává magický setter proměnné *OpenText*. Chceme-li nastavit otevřený text na hodnotu „TajnyTextUzivatele“, provedeme to následovně:

```
$crypt->OpenText = 'TajnyTextUzivatele';
```

Otevřený text by měl obsahovat znaky ze spodní poloviny ASCII tabulky, jiné znaky jsou ignorovány z důvodu jejich nejednoznačného číselného vyjádření.

Zašifrování otevřeného textu

Zašifrování otevřeného textu obstarává veřejná metoda *encrypt*. Stačí ji zavolat a ona nám vrátí zašifrovaný text. Proměnná *CryptText* se nemění.

```
$CT = $crypt->encrypt();
```

Výstupem je formátovaný řetězec. Pro snazší pochopení se výstup skládá z čísel 0 až 127 zapsaných v šestnáctkové soustavě, čísla oddělují mezery. Toto řešení je zvoleno proto, že spodní část ASCII tabulky obsahuje i speciální znaky, které by bylo nutno substituovat za znaky jiné. Respektive by bylo nutno nasadit kompletní kódování (např. Base64) a to by vedlo k zneřehlednění výstupu třídy *Crypt*, což je pro účely využití ve výuce nežádoucí.

Nastavení šifrovaného textu

Nastavení šifrovaného textu obstarává magický setter proměnné *CryptText*. Chceme-li nastavit šifrovaný text na hodnotu „28 42 54 5c 72 1f 51 61 57 2a 74 52 6c 42 68 4a 58 4a“, provedeme to následovně:

```
$crypt->CryptText = '28 42 54 5c 72 1f 51 61 57 2a 74 52 6c 42 68  
4a 58 4a';
```

Šifrovaný text je formátovaný řetězec, viz *Zašifrování otevřeného textu*.

Dešifrování šifrovaného textu

Dešifrování zašifrovaného textu obstarává veřejná metoda *decrypt*. Stačí ji zavolat a ona nám vrátí otevřený text. Proměnná *OpenText* se nemění.

```
$OT = $crypt->decrypt();
```

6.1.2 Aplikace na straně klienta

Na straně klienta je vše realizováno jednou JavaScriptovou třídou jménem *Crypt*. Třída byla psána tak, aby nebylo nijak složité ji pochopit a rozšířit o jiný šifrovací algoritmus, popřípadě doplnit o kompresní či kódovací algoritmus.

Tato třída je z převážné části symetrická s třídou na straně serveru. Je tomu tak z důvodu snazšího vývoje – zajišťuje to lepší udržitelnost kódu.

Inicializace

O inicializaci se stará JavaScript samotný. Novou instanci třídy *Crypt* vytvoříme pouhým zavoláním konstruktoru.

```
var crypt = new Crypt();
```

Nastavení klíče

Nastavení klíče obstarává magický setter proměnné *SecretKey*. Chceme-li nastavit klíč na hodnotu „TajnyKlicUzivatele“, provedeme to následovně:

```
crypt.SecretKey = 'TajnyKlicUzivatele';
```

Klíč by měl obsahovat znaky ze spodní poloviny ASCII tabulky. Jiné znaky jsou ignorovány z důvodu jejich nejednoznačného číselného vyjádření.

Nastavení otevřeného textu

Nastavení otevřeného textu obstarává magický setter proměnné *OpenText*. Chceme-li nastavit otevřený text na hodnotu „TajnyTextUzivatele“, provedeme to následovně:

```
crypt.OpenText = 'TajnyTextUzivatele';
```

Otevřený text by měl obsahovat znaky ze spodní poloviny ASCII tabulky, jiné znaky jsou ignorovány z důvodu jejich nejednoznačného číselného vyjádření.

Zašifrování otevřeného textu

Zašifrování otevřeného textu obstarává veřejná metoda *encrypt*. Stačí ji zavolat a ona nám vrátí zašifrovaný text. Proměnná *CryptText* se nemění.

```
var CT = crypt.encrypt();
```

Výstupem je formátovaný řetězec, viz *Zašifrování otevřeného textu* v sekci *Aplikace na straně serveru*.

Nastavení šifrovaného textu

Nastavení šifrovaného textu obstarává magický setter proměnné *CryptText*. Chceme-li nastavit šifrovaný text na hodnotu „28 42 54 5c 72 1f 51 61 57 2a 74 52 6c 42 68 4a 58 4a“, provedeme to následovně:

```
crypt.CryptText = '28 42 54 5c 72 1f 51 61 57 2a 74 52 6c 42 68 4a
58 4a';
```

Šifrovaný text je formátovaný řetězec, viz *Zašifrování otevřeného textu* v sekci *Aplikace* na straně serveru.

Dešifrování šifrovaného textu

Dešifrování zašifrovaného textu obstarává veřejná metoda *decrypt*. Stačí ji zavolat a ona nám vrátí otevřený text. Proměnná *OpenText* se nemění.

```
var OT = crypt.decrypt();
```

6.2 Praktické ukázky použití

Jelikož jsem třídu *Crypt* napsal tak, aby byla práce s ní velice jednoduchá a intuitivní, rozhodl jsem se přímo do této práce vměstnat i několik praktických ukázek.

6.2.1 Soukromá zpráva na webu

Chceme-li na webovou stránku umístit tajnou zprávu, můžeme ji s využitím třídy *Crypt* zašifrovat. Nejprve je nutno na serveru zprávu zašifrovat a vložit do stránky jako obsah snadno přístupného elementu (například tagu s ID).

```
<div id="encrypted"><?php
    $PM = new Crypt();
    $PM->OpenText = 'Ahoj pepo, k obedu bude svickova.';
    $PM->SecretKey = 'MujTajnyKlic';
    print_r($PM->encrypt());
?></div>
```

Protože budeme dešifrovat na straně klienta, musíme si připravit ještě jeden datový blok na dešifrovaná data. To proto, abychom při zadání chybného klíče nepoškodili původní zprávu. A rovnou si do kódu můžeme vypsát i vstupní pole pro zadání klíče a odkaz pro dešifrování.

```
<div id="decrypted"></div>
<input type="password" id="key" name="key">
<a href="#desifruj" id="encrypt">dešifruj</a>
```

Na závěr už nám pouze stačí přidat akci po kliknutí na odkaz. Tato akce vezme data z bloku dat, klíč přečte ze vstupního pole, data dešifruje a výsledek vloží do výstupního elementu.

```
<script>
    $(function() {
        $("#encrypt").click(function() {
            var PM = new Crypt();
            PM.CryptText = $("#encrypted").html();
            PM.SecretKey = $("#key").val();
            $("#decrypted").html(PM.decrypt());
            return false;
        });
    });
</script>
```

Zde by mohly vzniknout drobné nejasnosti kolem navracení hodnoty *false*. Tu navracíme pro zrušení efektu události *click*. To znamená, že se provede JavaScriptová obsluha události, ale samotná událost se již neprovede.

SOUKROMÁ ZPRÁVA

5a 7f 77 5e 1d 5a 2c 06 55 75 39 55 03 78 í 2a 78 63 32 7f 79 77 37 30 65 52 73 14 1d 69 59
55 2b 1d 5a 32 61 53 61 5b 1f 16 59 44 26 1d é 14 6c 5f 60 á 34 19 22 6e 74 2a 2f 10 38 7d
7c 0d 00 35 1e 4b 17 46 38 34 30 02 72 ý 2a 2b 54 31 01 6b 7f 29 55 15 5c 61 0a 5d é 13 30
46 63 23 5d 10 66 á 60 67 3c 4f 21 52 2b 6c 07 7a 2e 2f 66 37 49 39 72 77

Tato zpráva je soukromá, zde zadejte svůj tajný klíč a [klikněte zde](#).

Obr. 7 – Ukázka implementace soukromé zprávy na webu

SOUKROMÁ ZPRÁVA

První **odstavec** tajné zprávy.

Druhý odstavec tajné zprávy.

Tato zpráva je soukromá, zde zadejte svůj tajný klíč a [klikněte zde](#).

Obr. 8 – Ukázka implementace soukromé zprávy na webu (správný klíč)

SOUKROMÁ ZPRÁVA

!! 1qlyKh&\$! /i_+=# mJwX (D-J & 9-g| + P→ 4t -Mt ié f!láK|T b>/E gt+B-Y ,OR
r*)FýX b•X% Q73U e^é}qZ> + áo ~P◀H +}z|^ L?v|

Tato zpráva je soukromá, zde zadejte svůj tajný klíč a [klikněte zde](#).

Obr. 9 – Ukázka implementace soukromé zprávy na webu (chybný klíč)

Porovnání s přihlašováním

Šifrování má oproti přihlašování jednu neocenitelnou výhodu. Pokud chceme zveřejnit data předem neznámému počtu uživatelů, pak stačí vybraným osobám předat klíč a vše je hotovo. Jedná se tak o snadnou možnost sdílení dat předem neznámému počtu osob.

V případě přihlašování bychom museli vytvořit jeden účet sdílený všemi vybranými osobami, nebo každé z vybraných osob vytvořit unikátní profil. To je neocenitelné v případě, kdy chceme uživatele odlišit (např. při distribuci prémiového obsahu), nebo kdy chceme uživatelům poskytovat obsah upravený právě pro ně (např. cílenou reklamu).

6.2.2 CAPTCHA

Ať už se rozhodneme provést CAPTCHA jakkoliv, vždy potřebujeme generátor náhodného kódu, který bude uživatel opisovat. Pro zdejší účely budeme uvažovat, že generátor je funkce *RandomString(\$StrLen)* a její parametr *StrLen* určuje délku řetězce ve znacích.

Nejjednodušší implementace CAPTCHA s využitím třídy *Crypt* nepotřebuje žádný generátor obrázků a obejde se i bez *cookies*¹¹ či *sessions*¹². Jako otevřený text si zvolme hash identifikující klienta a jako klíč náhodný řetězec.

```
<?php
    $hash = sha1($_SERVER['HTTP_USER_AGENT']);
    $hash.= md5($_SERVER['REMOTE_ADDR']);
    $captcha = new Crypt();
    $captcha->OpenText = $hash;
```

¹¹ Cookies jsou prostředkem k uchování dat na straně klienta za účelem identifikace nebo sledování uživatele. [22]

¹² Session slouží k přenosu dat mezi jednotlivými požadavky klienta na server, jsou uchovávány na serveru a klient se k nim hlásí za pomocí ID uloženého v cookie. [23]

```

    $captcha->SecretKey = RandomString(5);
?>

```

Nyní si připravme vstupní pole pro klíč, snadno přístupný element (například tag s ID) pro umístění zašifrovaného hashe a odkaz pro provedení akce.

```

<div id="encrypted"><?php
    print($captcha->encrypt());
?></div>
<code><?php print_r($captcha->SecretKey); ?></code>
<input type="text" name="key" id="key" />
<a href="#captcha" id="link">OK</a>

```

Na závěr nám již stačí pouze vytvořit obsluhu velice podobnou jako u Soukromá zpráva na webu. Jedinou změnou bude podmínka v PHP, která porovná hash se zasláným hashem a rozhodne o projití, nebo neprojití CAPTCHA.

```

<script type="text/javascript">
    $(function(){
        $("#link").click(function(){
            var captcha = new Crypt();
            captcha.CryptText = $("#encrypted").html();
            captcha.SecretKey = $("#key").val();
            $("#link").attr("href","./?captcha=" +
captcha.decrypt());
            return true;
        });
    });
</script>
<?php
    if(isset($_GET['captcha'])) {
        if($_GET['captcha'] == $hash) print_r('prošel');
        else print_r('neprošel');
    }
?>

```

V této obsluze si povšimněte, že návratová hodnota je *true*, tedy obsluha nám změní cíl odkazu a následně klienta na tuto adresu odkáže. Díky tomu může server porovnat, zda zasláný otisk klientem je správný, nebo nikoliv.

CAPTCHA

Opište kontrolní kód "Lh8GE" do tohoto pole a klikněte zde.

Obr. 10 – Ukázka implementace CAPTCHA

CAPTCHA

Kontrolní kód souhlasí.

Opište kontrolní kód "j412e" do tohoto pole a klikněte zde.

Obr. 11 – Ukázka implementace CAPTCHA (správný kód)

CAPTCHA

Kontrolní kód nesouhlasí.

Opište kontrolní kód "t33KG" do tohoto pole a klikněte zde.

Obr. 12 – Ukázka implementace CAPTCHA (chybný kód)

Porovnání s běžnou implementací CAPTCHA

Oproti běžným implementacím je zde výhoda, že je před stroj položena navíc ještě úloha provedení třídy *Crypt*. SPAM roboti jsou převážně specializované programy, které implementují OCR pro případ výskytu CAPTCHA využívající vtisknutí kódu do obrázku. Pouze ve velice vzácných případech si proto dokáží poradit se složitějším JavaScriptem, který však běžným prohlížečům (včetně mobilních verzí) nedělá žádné problémy.

6.2.3 Zabezpečená komunikace mezi klientem a serverem

Asi nejlepší ukázkou využití třídy *Crypt* je zabezpečení komunikace mezi klientem a serverem a naopak. Ideální by bylo, kdyby komunikující strana k zašifrovaným datům přidala klíč pro další relaci. Pro první výměnu dat bohužel musíme použít předem známý klíč, např. uživatelský klíč uložený v databázi. Pro snazší pochopení ukázky kódu si toto předávání klíčů popíšeme pouze slovně (viz Výměna klíčů při komunikaci) a v ukázce kódu jej vypustíme. V případě zájmu je implementace zabezpečené obousměrné komunikace s výměnou klíčů k nahlédnutí v příložených zdrojových souborech.

Protože je API třídy *Crypt* totožné pro server i klienta, ukážeme si pouze jednosměrnou komunikaci. Druhý směr by byl totožný, pouze by se otočila role serveru a klienta.

Nejprve je nutno na serveru správně inicializovat třídu *Crypt*. Pro výměnu dat si určíme například textové pole *data* a dešifrovaná data si uložíme do proměnné *data*.

```
<?php
    $SDF = new Crypt();
    $SDF->SecretKey = 'TajnyKlicUzivatele';
    if(isset($_POST['data'])) {
        $SDF->CryptText = $_POST[data];
        $data = $SDF->decrypt();
    }
?>
```

Nyní je vhodné připravit pro klienta vstupní rozhraní. Pro zjednodušení necháme uživatele zadat klíč ručně. V reálných podmínkách by byl klíč uchovávan v proměnné *sessionStorage*¹³. Pro náš jednoduchý příklad si tedy připravíme textové pole pro zadávání dat, vstupní pole pro zadání klíče a tlačítko pro odeslání dat. Pro lepší kompaktnost vše zabalíme do formuláře.

```
<form method="post" action="." id="SDFform">
    <div>
        <label for="data">Data:</label>
        <textarea name="data" id="data" rows="10"
cols="50"></textarea>
    </div>
    <div>
        <label for="key">Klíč:</label>
        <input type="text" name="key" id="key" />
    </div>
    <div>
        <input name="button" id="button" type="submit"
value="odešli data" />
    </div>
</form>
```

¹³ *sessionStorage* je objekt určený k uchování dat na straně klienta, tato data jsou přístupná pouze v rámci jednoho vlákna prohlížeče a nikdy nejsou odesílána bez výslovného vyžádání JavaScriptem. [24]

I zde je nakonec nutné vytvořit obsluhu na straně klienta, obdobně jako v předešlých příkladech.

```
<script type="text/javascript">
    $(function() {
        $("#SDFform").submit(function() {
            var SDF = new Crypt();
            SDF.OpenText = $("#data").val();
            SDF.SecretKey = $("#key").val();
            $("#key").val("");
            $("#data").val(SDF.encrypt());
            return true;
        });
    });
</script>
```

Obsluha je volána při odesílání formuláře, ovšem v odeslání nijak nebrání. Obsluha pouze zajišťuje, že se před odesláním data zašifrují a klíč se vymaže. To nám zajistí, aby z klienta neodešla data v čitelné podobě, především tedy klíč. Výměna statického klíče mezi serverem a klientem je nejenom nežádoucí, ale navíc jde i o velice nebezpečnou bezpečnostní trhlinu.

ZABEZPEČENÝ PŘENOS DAT

Data:

Klíč:

Obr. 13 – Ukázka implementace zabezpečení komunikace

ZABEZPEČENÝ PŘENOS DAT

Server obdržel následující data:

```
Array
(
    [secure_data_transfer_data] => 15 49 59 58 19 3b 51 59 52 1 1a 54 16 50 56 4a 50 5a 61 4a 64 4e 5 10 58 66 58 f b 5a 16 50 10
    [secure_data_transfer_key] =>
    [secure_data_transfer_button] => odešli data
)
```

Po dešifrování datové části byla nalezena následující zpráva:

Ahoj pepo, k obědu bude svickova.

Data:

Klíč:

Obr. 14 – Ukázka implementace zabezpečení komunikace (správný klíč)

ZABEZPEČENÝ PŘENOS DAT

Server obdržel následující data:

```
Array
(
    [secure_data_transfer_data] => 15 57 57 56 5 10 4f 55 f 16 9 59 20 f 4d 51 64 75 41 23 5d 53 4f 40 63 5b 59 52 17 f 61 1 1d
    [secure_data_transfer_key] =>
    [secure_data_transfer_button] => odešli data
)
```

Po dešifrování datové části byla nalezena následující zpráva:

AvmhEcl, A&#p*.Ylx†-pkC{o- 7SD

Data:

Klíč:

Obr. 15 – Ukázka implementace zabezpečení komunikace (chybný klíč)

Výměna klíčů při komunikaci

1. Klient nechá uživatele zadat přihlašovací údaje a odešle je serveru zašifrovaná předem známým klíčem.
2. Server data obdrží, dešifruje je předem známým klíčem a data zpracuje.
3. Server vygeneruje odpověď a náhodný klíč.
4. Server odpověď zašifruje vygenerovaným klíčem a vygenerovaný klíč zašifruje klíčem z bodu 2.
5. Server obě zašifrované zprávy spojí a pošle klientovi.
6. Klient nejprve dešifruje nový klíč klíčem z bodu 2 a následně novým klíčem dešifruje zbytek dat.

7. Klient uživateli předloží data, vygeneruje nový náhodný klíč a čeká na akci uživatele.
8. Klient akci uživatele zpracuje, výsledná data zašifruje novým klíčem a nový klíč zašifruje klíčem z bodu 6.
9. Klient obě zprávy spojí a pošle je serveru.

Dále již komunikace navazuje na bod 2, pouze místo předem známého klíče je použit předchozí náhodný klíč.

Fungující implementace je součástí přiložených zdrojových kódů. Pro přenos dat je využíván JSON a komunikaci obstarává AJAX.

Porovnání s HTTPS

Oproti HTTPS má třída *Crypt* tu výhodu, že k ní není zapotřebí generovat žádné certifikáty, nicméně ve všem ostatním je HTTPS na zabezpečení komunikace vhodnější. Nejvíce užítku lze získat nasazením třídy *Crypt* nad samotný protokol HTTPS. V tomto případě by třída *Crypt* vytvořila zabezpečení dat na stranách klienta a serveru, které samotný protokol HTTPS neposkytuje.

ZÁVĚR

V této bakalářské práci jsou popsány běžné metody zabezpečení obsahu webových stránek. Jsou zde dále uvedeny vybrané kryptografické i steganografické postupy, které se při zabezpečování obsahu webových stránek používají, jejich silné i slabé stránky a je zde ukázáno, kde se s nimi můžeme setkat.

Nejsilnější zbraní při obraně digitálních dat, jak bakalářská práce několikrát lehce naznačila, je bezesporu kryptografie. Avšak ani ona není všemohoucí a je pouze otázkou času, za jak dlouho ta či ona kryptografická metoda selže. Díky tomu je potřeba neustálého vývoje. Bohužel i zde platí pravidlo, že útočníci jsou vždy o krok napřed.

Obsahem této práce byl i návrh a realizace vlastního kryptografického řešení. To je postaveno výlučně pro využití na webových stránkách, a proto se opírá o technologie podporované napříč velkým spektrem webových prohlížečů i serverů. Aby byla zachována možnost využití dané implementace při výuce, byla zvolena jednoduchá šifra autoklíč a pro kompresi výstupních dat byl zvolen pouhý převod z desítkové do šestnáctkové soustavy.

Při porovnání se současnými metodami si řešení zmíněné v této práci obecně, tj. při použití složitější šifry a komprese, nevedlo o moc hůře. Nicméně dle předpokladů si našlo své hlavní místo jako nadstavba nad některou z již používaných metod.

Závěrem lze konstatovat, že zabezpečení používaná v současné době jsou v současné době dostatečná. Ovšem nelze „usnout na vavřínech“, je nutno tyto technologie a znalosti neustále vylepšovat.

ZÁVĚR V ANGLIČTINĚ

This bachelor thesis describes commonly used methods for securing web page content. Thesis focuses on selected cryptographic and steganographic processes used for securing content on the Internet. By the way thesis shows strengths, weakness and real examples of these methods.

As thesis show, cryptography provides the best protection for digital data. But even cryptography doesn't perfect. Every securing method is secure only in limited time. It means that every method can be overcome in unlimited time. Therefore it is important to continually developing the methods – attackers are a step ahead.

The second part of this thesis contains design and implementation of my own cryptographic solution. The solution is suitable for use on web pages – the solution using only technologies with great support in each web browser and on each web server. The solution use simple Autokey cipher and its output convert to hexadecimal numeral system, so it can be used as an education material.

As shown the solution with stronger cipher and better compression can be comparable with commonly used methods. The best usage for the solution is as the independent security layer over another security layer.

We can conclude that security methods used in the present are secure just in the present. But development must continue – it is important to promote our technologies and knowledge.

SEZNAM POUŽITÉ LITERATURY

1. Spambot: Definition from Answers.com. In: *Answers.com: ReferenceAnswers* [online]. 8.3.2012 [cit. 2012-03-26]. Dostupné z: <http://answers.com/topic/spambot>
2. STALLINGS, William. *Cryptography and network security: principles and practice*. 3rd ed. Upper Saddle River: Prentice Hall, c1999, xiv, 681 s. ISBN 01-311-1502-2.
3. Steganography. In: *CryptoDox* [online]. 2005, 29.1.2007 [cit. 2012-04-01]. Dostupné z: <http://cryptodox.com/Steganography>
4. CAPTCHA. In: *CryptoDox* [online]. 12.10.2007 [cit. 2012-03-31]. Dostupné z: <http://cryptodox.com/CAPTCHA>
5. Digital watermarking. In: *CryptoDox* [online]. 12.10.2007 [cit. 2012-03-31]. Dostupné z: http://cryptodox.com/Digital_watermarking
6. HUSEBY, Sverre H. *Zranitelný kód*. Vyd. 1. Brno: Computer Press, 2006, 207 s. ISBN 80-251-1180-6.
7. MATYÁŠ, Vašek a Jan KRHOVJÁK. *Autorizace elektronických transakcí a autentizace dat i uživatelů*. Brno: Masarykova univerzita, 2008, 125 s. ISBN 978-802-1045-569.
8. Google Authenticator. In: *Nápověda Google* [online]. 2012 [cit. 2012-04-15]. Dostupné z: <http://support.google.com/a/bin/answer.py?hl=en&answer=1037451>
9. VALÁŠEK, Michal. Michal Altair Valášek: Všechno, co víte o heslech, je špatně. *Tech - iHNed.cz: Ekonomický zpravodajský server Hospodářských novin* [online]. 28.3.2012 [cit. 2012-04-01]. ISSN 1213-7693. Dostupné z: <http://tech.ihned.cz/geekosfera/c1-55195370-michal-altair-valasek-vsechno-co-vite-o-heslech-je-spatne>
10. VALÁŠEK, Michal. Uchovávání hesel ve webových aplikacích. *ASPNET.CZ - in technology we trust* [online]. 17.5.2006 [cit. 2012-04-01]. ISSN 1801-9447. Dostupné z: <http://www.aspnet.cz/articles/93-uchovavani-hesel-ve-webovych-aplikacich>
11. RIVEST, Ronald L. RFC 1321 - The MD5 Message-Digest Algorithm. In: *Internet Engineering Task Force (IETF)* [online]. Cambridge (MA), 8.4.1992 [cit. 2012-04-

- 01]. Dostupné z: <http://tools.ietf.org/html/rfc1321>
12. NIST COMPUTER SECURITY DIVISION. Federal Information Processing Standards Publication: Secure Hash Standard (SHS). In: NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. *NIST.gov - Computer Security Division - Computer Security Resource Center* [online]. Patrick Gallagher. Cita Furlani. Carlos M. Gutierrez. Gaithersburg (MD), 6.11.2008, s. 32 [cit. 2012-04-02]. FIPS PUB 180-3. Dostupné z: http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf
13. NIST COMPUTER SECURITY DIVISION. Federal Information Processing Standards Publication: Data Encryption Standard (DES). In: NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. *NIST.gov - Computer Security Division - Computer Security Resource Center* [online]. William M. Daley. Raymond G. Kammer. William Mehuron. Gaithersburg (MD), 25.10.1999, s. 26 [cit. 2012-04-22]. FIPS PUB 46-3. Dostupné z: <http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf>
14. RIVEST, Ronald L., A. SHAMIR a L. ADLEMAN. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. In: *MIT Computer Science and Artificial Intelligence Laboratory* [online]. Cambridge (MA), 4.4.1977 [cit. 2012-04-13]. Dostupné z: <http://people.csail.mit.edu/rivest/Rsapaper.pdf>
15. NIST INFORMATION TECHNOLOGY LABORATORY. Federal Information Processing Standards Publication: Digital Signature Standard (DSS). In: NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. *Information Technology Laboratory Homepage* [online]. Ronald H. Brown. James H. Burrows. Arati Prabhakar. Gaithersburg (MD), 19.5.1994 [cit. 2012-04-13]. FIPS PUB 186. Dostupné z: <http://itl.nist.gov/fipspubs/fip186.htm>
16. PHP Manual: Preface. THE PHP GROUP. *PHP.net* [online]. 20.3.2012 [cit. 2012-03-31]. Dostupné z: <http://php.net/manual/en/preface.php>
17. JavaScript Tutorial. REFSNES DATA. *W3Schools Online Web Tutorials* [online]. © 1999-2012 [cit. 2012-03-31]. Dostupné z: <http://w3schools.com/js/default.asp>
18. THE JQUERY PROJECT. *jQuery: The Write Less, Do More, JavaScript Library* [online]. 21.3.2012 [cit. 2012-03-31]. Dostupné z: <http://jquery.com/>

19. Autokey cipher. In: *CryptoDox* [online]. 12.10.2007, 17.10.2007 [cit. 2012-03-31].
Dostupné z: http://cryptodox.com/Autokey_cipher
20. WEISSTEIN, Eric W. Hexadecimal. In: *MathWorld: A Wolfram Web Resource* [online]. © 1999-2012 [cit. 2012-03-31]. Dostupné z:
<http://mathworld.wolfram.com/Hexadecimal.html>
21. PŘISPĚVATELÉ WIKIPEDIE. ASCII. In: *Wikipedia: the free encyclopedia* [online].
San Francisco (CA): Wikimedia Foundation, 30.4.2009, 2.4.2012 [cit. 2012-04-23].
Dostupné z: <http://cs.wikipedia.org/w/index.php?title=ASCII&oldid=8340983>
22. PHP: Cookies. THE PHP GROUP. *PHP.net* [online]. 27.4.2012 [cit. 2012-04-28].
Dostupné z: <http://cz2.php.net/cookies>
23. PHP: Sessions - Introduction. THE PHP GROUP. *PHP.net* [online]. 27.4.2012 [cit.
2012-04-28]. Dostupné z: <http://cz2.php.net/manual/en/intro.session.php>
24. HTML5 Web Storage. REFSNES DATA. *W3Schools Online Web Tutorials* [online].
© 2009-2012 [cit. 2012-04-28]. Dostupné z:
http://www.w3schools.com/html5/html5_webstorage.asp

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

AES	Advanced Encryption Standard
AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
CA	certifikační autorita
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart
DES	Data encryption standard
DSA	Digital Signature Algorithm
HTTP	Hypertext Transform Protocol
HTTPS	Hypertext Transform Protocol Secure
JS	JavaScript
JSON	JavaScript Object Notation
MD	Message-Digest
OCR	Optical Character Recognition
PHP	Personal Home Page
RSA	iniciály autorů asymetrické šifry (Rivest, Shamir, Adleman)
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SPAM	přenesený výraz pro nevyžádanou věc (původně spiced ham)
SSL	Secure Socket Layer
TLS	Transport Layer Security

SEZNAM OBRÁZKŮ

Obr. 1 – Elektronická skripta.....	13
Obr. 2 – Webová služba LeteckáPosta.cz.....	15
Obr. 3 – Registrace na serveru CodePlex.com	16
Obr. 4 – Vodoznak České televize v levém horním rohu vysílání	17
Obr. 5 – Jeden z mnoha výpočetních čipů DES cracker.....	28
Obr. 6 – Princip symetrické kryptografie	36
Obr. 7 – Ukázka implementace soukromé zprávy na webu	42
Obr. 8 – Ukázka implementace soukromé zprávy na webu (správný klíč)	42
Obr. 9 – Ukázka implementace soukromé zprávy na webu (chybný klíč).....	43
Obr. 10 – Ukázka implementace CAPTCHA.....	45
Obr. 11 – Ukázka implementace CAPTCHA (správný kód).....	45
Obr. 12 – Ukázka implementace CAPTCHA (chybný kód)	45
Obr. 13 – Ukázka implementace zabezpečení komunikace	47
Obr. 14 – Ukázka implementace zabezpečení komunikace (správný klíč)	48
Obr. 15 – Ukázka implementace zabezpečení komunikace (chybný klíč).....	48

SEZNAM TABULEK

Tab. 1 – Tabulka s daty pro přihlášení.....	18
Tab. 2 – Tabulka se zabezpečenými daty	20