

Vysoká škola ekonomická v Praze

Fakulta informatiky a statistiky

Katedra systémové analýzy

Studijní program: Aplikovaná informatika

Obor: Informatika

Webové aplikace a vliv šifrování na jejich rychlost

BAKALÁŘSKÁ PRÁCE

Vypracoval: Jan Podavka

Vedoucí práce: Ing. Jaromír Veber, Ph.D.

Oponent práce: Ing. Tomáš Klíma

Rok vypracování: 2014

Čestné prohlášení:

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Veškeré použité podklady, ze kterých jsem čerpal informace, jsou uvedeny v seznamu použité literatury a citovány v textu podle normy ČSN ISO 690.

V Praze dne

Podpis:

Poděkování:

Chtěl bych poděkovat Ing. Jaromíru Veberovi za všechny rady a doporučení k vytvoření této práce. Oceňuji jeho věcné a odborné připomínky k této práci a vstřícné jednání. Dále bych chtěl poděkovat všem, kteří mi poskytli morální podporu při studiu zdrojů a následném psaní práce.

Abstrakt

Tato bakalářská práce řeší otázku, jaký vliv má šifrování dat na webových stránkách na velikost databáze a reakci systému. Tato otázka je řešena pomocí testování šifrovacích metod na systému Raspberry Pi. Práce v teoretické části popisuje šifrovací algoritmy. Dále popisuje, jakým způsobem zabezpečit web, kde použít šifrování na webu a jaké použít. Praktická část je zaměřena na vliv šifrování na systém. Výsledkem je zhodnocení rychlosti algoritmů v různých situacích a také porovnání velikosti databáze s šifrovanými a nešifrovanými daty.

Klíčová slova

Rychlost šifrování, kryptografie, šifry, Raspberry Pi

Abstract

This bachelor thesis deals with how the size of the database and response of the system is influenced by an encryption of the data on the websites. This issue is solved by means of testing of encryption methods on the Raspberry Pi. Theoretical part of the thesis describes the encryption algorithms. It also describes how to secure the website and where to use the encryption on the web and what kind of encryption to use. The practical part focuses on the impact of the encryption on the system. As a result, the speed of the algorithms in different situations is evaluated, and also the size of the database with cypher text and plain text is compared.

Keywords

Encryption speed, cryptography, encryption, Raspberry Pi

1. Obsah práce

1.	Obsah práce	1
2.	Úvod	4
2.1.	Důvod výběru tématu a jeho vymezení	4
3.	Cíle práce.....	5
3.1.	Vymezení práce.....	5
3.2.	Omezení práce	5
4.	Stav problematiky	6
5.	Šifrování (Kryptografie)	7
5.1.	Základní pojmy	7
5.1.1.	XOR.....	7
5.1.2.	Runda (Rounds).....	7
5.2.	Cíle.....	7
5.3.	Symetrické šifry.....	8
5.3.1.	Rozdělení šifer	8
5.3.2.	Bezpečnost	9
5.3.3.	Šifry	10
5.3.4.	Módy:	13
5.4.	Asymetrické šifry.....	16
5.5.	Hashovací funkce	17
5.6.	Vernamova šifra	19
5.7.	Délka klíče	19
5.8.	Zásady tvorby hesel	21
5.9.	Bezpečnost přenosu dat	22
5.9.1.	HTTPS	22
5.9.2.	SSL/TLS	22

5.9.3.	Certifikační autority	22
5.10.	Bezpečnost webové aplikace	22
5.10.1.	SQL injection	23
5.10.2.	Cross site Scripting (XSS)	23
5.10.3.	Cross-Site Request Forgery (CSRF)	23
6.	Uživatelský komfort	24
6.1.	Šifrování společným klíčem	24
6.1.1.	Hodnocení	25
6.2.	Šifrování vlastním klíčem	25
6.3.	Šifrování zadávaným klíčem	26
6.4.	Soukromý a veřejný klíč	26
6.4.1.	Hodnocení	27
6.5.	Osobní databáze	27
6.6.	Databáze uživatelů	27
6.7.	MLM systém prohlížení	27
6.8.	Sociální síť, diskuze	28
6.9.	Email	28
6.9.1.	Vlastní síť	28
6.9.2.	Cizí síť	29
7.	Testovací systém Raspberry Pi	31
7.1.	Specifikace	31
7.2.	Nastavení	31
7.3.	Instalace webového serveru	32
7.4.	Změna adresáře na plochu	32
7.5.	Metodika testu	33
7.5.1.	Šifrovací metody	33
7.5.2.	Vstupní data	34

7.5.3.	Šifrovací klíč/dešifrovací klíč	35
7.5.4.	Mód	35
7.5.5.	Inicializační vektor	35
7.5.6.	Typy testů	35
7.6.	Vlastní měření	36
7.6.1.	Velikost databáze	37
7.6.2.	Rychlost šifrování - jeden uživatel	37
7.6.3.	Rychlost šifrování - více uživatelů	41
7.6.4.	Vyhodnocení testů	44
8.	Závěr	46
8.1.	Zhodnocení šifrování dat ve webové aplikaci	48
8.2.	Přínosy práce	48
9.	Seznam obrázků	49
10.	Seznam tabulek	50
11.	Seznam grafů	51
12.	Bibliografie	52
13.	Příloha 1 - Verze balíčků PHP, MySQL a Apache	57
14.	Příloha 2 - SQL kódy	59
15.	Příloha 3 - Zdrojový kód	59
16.	Příloha 4 - Konfigurace PHP	59

2. Úvod

Problematika zabezpečení dat na internetu je z mého pohledu velmi podceňovaná záležitost. Ve svém okolí se setkávám s názorem, že kdo nic špatného nedělá, nemusí nic skrývat. Dalším názorem, se kterým se setkávám, je názor: „proč by šli zrovna po mně“ a odůvodňují to slovy: „o mě se zajímat nebudou“. Dodají, že když se budou chovat na internetu zdrženlivě a nebudou o sobě dávat informace jako většina, budou pro ně podezřelí a zaměří se na ně.

Podle listiny základních práv a svobod v článku 13: je napsáno: *„Nikdo nesmí porušit listovní tajemství ani tajemství jiných písemností a záznamů, ať již uchovávaných v soukromí, nebo zasílaných poštou anebo jiným způsobem, s výjimkou případů a způsobem, které stanoví zákon. Stejně se zaručuje tajemství zpráv podávaných telefonem, telegrafem nebo jiným podobným zařízením.“* (1) Z tohoto textu vyplývá, že se listovní tajemství týká i emailové komunikace nebo jiného osobního sdělení posílaného pomocí internetu.

Jak nám pan Edward Snowden z Národní bezpečnostní agentury Spojených států amerických ukázal, podle informací, které vynesl, USA sledují občany nejen ze Spojených států amerických a bez soudního rozhodnutí. Obávám se, že tato taktika nebude jen v Americe. A bylo by bláhové si myslet, že vyjmou osobní komunikaci ze sledování.

Dalším příkladem jsou nové smluvní podmínky Googlu, o kterých jsem se dozvěděl ze serveru CDR. (2) V těchto podmínkách je napsáno toto: „Váš obsah (včetně e-mailů) analyzují naše automatizované systémy, abychom vám mohli nabídnout funkce služeb relevantní přímo pro vás, například přizpůsobené výsledky vyhledávání, personalizované reklamy a detekci spamu a malwaru. K této analýze dochází při odeslání, přijetí a uložení obsahu.“ (3) Tato část smluvních podmínek z mého pohledu odporuje Listině základních práv a svobod České republiky. Jenže přestat používat služby Googlu je v dnešní době skoro vyloučené, protože mobilní telefony s operačním systémem Android vyžadují Google účet pro stažení aplikací.

2.1. Důvod výběru tématu a jeho vymezení

Při výběru bakalářské práce jsem se snažil zaměřit na témata bezpečnosti nebo webu, a když jsem našel téma zabezpečení dat na internetu, věděl jsem, že bych mohl tato dvě témata skloubit dohromady a vytvořit bakalářskou práci o vlivu šifrování na webový server a zjistit rozdíly mezi šifrovanou a nešifrovanou stránkou.

3. Cíle práce

Cílem práce je vybrat vhodné šifrovací techniky na webu a v praktické části zhodnotit vliv šifrování dat do databáze na reakci webového serveru a velikost databáze.

3.1.Vymezení práce

V této práci se v teoretické části zaměřím na různé druhy šifrování dat. Uvedu jednotlivé typy šifer a dále uvedu problémy, které se týkají uživatelského komfortu na stránkách se zabezpečením.

V úvodu praktické části uvedu nastavení počítače Raspberry Pi a metodiku testu a následně na něm provedu měření vlivu různých šifrovacích metod na rychlost webového serveru a jak velký bude rozdíl ve velikosti databáze.

3.2.Omezení práce

Okruh bezpečnosti je obsáhlý a proto je v této práci důležité vyjmout z něj pouze oblast zabezpečení webu a šifrovacích technik a zaměřit se na ně.

Protože specifikace šifrovacích algoritmů jsou popsány na desítkách stran, nebudu v této práci popisovat práci jednotlivých algoritmů.

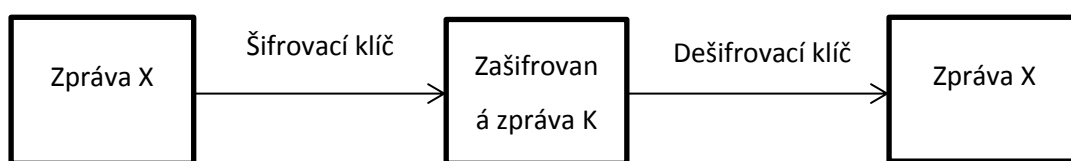
4. Stav problematiky

O Problematice rychlosti šifrování bylo napsáno mnoho článků například zde: (4). nebo v Javě (5). V České republice se studenti ve svých bakalářských pracích zaměřili i na zabezpečení cloudových služeb (6). Robert Ganian se zabýval ve své bakalářské práci testováním rychlosti proudových šifer (7). Výzkum výkonu šifrování na Raspberry Pi jsem nezaznamenal.

5. Šifrování (Kryptografie)

Definice šifrování je z učebnice Matematické základy informatiky „Šifrováním rozumíme způsob, jak převést text zprávy čitelný pro nezúčastněného pozorovatele na text jiný, zašifrovaný, nečitelný bez znalosti další informace. Tato další informace nutná pro přečtení zašifrovaného textu zprávy se nazývá dešifrovací klíč. Šifrovací klíč užívající se k zašifrování textu, se nemusí shodovat se dešifrovacím klíčem.“ (8 str. 128)

Schéma šifrování je vidět na obrázku 1.



5.1. Základní pojmy

5.1.1. XOR

Logická operace nad binárními daty, kdy ze dvou vstupů se vytvoří jeden výstup. Pokud jsou vstupy rozdílné, výstup je pravda, jinak je nepravda. Vše je vidět v tabulce 1

VSTUPY		VÝSTUP
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Tabulka 1- XOR; Zdroj: (9)

5.1.2. Runda (Rounds)

Šifrovací algoritmus je založen na provedení určitých operací. Tyto operace se opakují. Jeden průchod těmito operacemi se nazývá runda. (10)

5.2. Cíle

Moderní kryptografie si klade za cíl plnit následující cíle:

Důvěrnost dat - Utajení informací před neoprávněnými uživateli. Kromě zašifrování dat lze použít i řízení přístupu k těmto informacím. (11 str. 11)

Integrita dat - Zabezpečení nezměnitelnosti informací, ať už náhodnou nebo úmyslnou změnou. (11 str. 11)

Autentizace entit - Ověření identity uživatele, počítače, atd. (11 str. 11)

Autentizace dat - Ověření informací v datech jako je obsah, datum vzniku, autor, atd. Ve chvíli, kdy jsou všechna data ověřena, pak data mají integritu. (11 str. 11)

Nepopiratelnost - Nepopiratelnost zajišťuje zpětnou dohledatelnost informací o manipulaci s informacemi, jako je jejich vytvoření, přenos, atd. (11 str. 11)

Řízení přístupu - Zajištění přístupu znamená, že se k informacím dostane pouze oprávněný subjekt. (11 str. 11)

5.3.Symetrické šifry

Jednotlivé šifry můžeme rozdělit do několika skupin. V případě, kdy je šifrovací i dešifrovací klíč stejný, mluvíme o symetrických šifrách.

5.3.1. Rozdělení šifer

Symetrické šifry se dělí na dvě skupiny. První a větší z nich jsou blokové šifry a druhou skupinou jsou proudové šifry. Částečný seznam šifer a jejich rozdělení je v tabulce 2 na další straně.

Blokové šifry rozdělí text na bloky textu a v závislosti na šifrovací metodě jsou tyto bloky různě velké. Většinou se bloky pohybují od 64 bitů po 128 bitů. Proudové šifry zpracovávají text po jednotlivých znacích. (12 str. 9)

Šifra	Typ
A5/1	Proudová šifra
AES (Rijndael)	Bloková šifra
Blowfish	Bloková šifra
CAST-128	Bloková šifra
DES, 3DES	Bloková šifra
IDEA	Bloková šifra
FISH	Proudová šifra
GOST	Bloková šifra
LOKI97	Bloková šifra
RC2	Bloková šifra
RC4	Proudová šifra
RC5	Bloková šifra
RC6	Bloková šifra
Serpent	Bloková šifra
Twofish	Bloková šifra

Tabulka 2 - Symetrické šifry; Zdroj: (13)

Další šifrovací algoritmy naleznete například na Wikipedii http://en.wikipedia.org/wiki/Symmetric-key_algorithm nebo na stránkách Národního institutu standardů a technologií <http://csrc.nist.gov/projects/crypto.html>.

5.3.2. Bezpečnost

Bezpečnost blokových šifer je na dobré úrovni, pokud mají dostatečně dlouhý šifrovací klíč. Kryptoanalytici se snaží šifry prolomit a používají k tomu různé techniky.

Known plain text - Útočník má jak nezašifrovaný, tak zašifrovaný text. Díky této znalosti může získat šifrovací klíč a vlastnosti algoritmu. Standard AES není zranitelný vůči tomuto typu útoku. (14)

Chosen plain text - Útočník zašifruje vlastní text a získá tak nejen zašifrovaný text, ale i informace o samotné šifře. (15)

Diferenciální kryptoanalýza - Při použití diferenční kryptoanalýzy se hledají rozdíly, jak se změní výstup šifry při změně vstupu. (16)

Lineární kryptoanalýza - Při použití lineární kryptoanalýzy se hledají lineární závislosti mezi akcemi šifry. (17)

Boomerang - Útok je založen na diferenční kryptoanalýze a umožňuje zaútočit na šifry, které byly bezpečné proti diferenční kryptoanalýze. (18)

Kvantový počítač - Budoucí bezpečnostní hrozbou k šifrám, tak jak je dnes známe, je kvantový počítač, který pracuje odlišně než dnešní počítače a asymetrické šifrování bude kvantovým počítačem prolomeno během okamžiku. (19)

5.3.3. Šifry

5.3.3.1.DES, 3DES

Šifrovací standard americké vlády pro neklasifikované informace ve státní správě, vytvořený na začátku 70. let 20. století. DES používá algoritmus se 64 bitovým šifrovacím klíčem, z něhož 8 bitů je kontrolních, takže efektivní velikost klíče je 56 bitů. Počet klíčů je tedy 2^{56} . Tento počet klíčů byl vyzkoušen útokem hrubou silou v roce 1998 pomocí specializovaného stroje DES-cracker. Vyzkoušení všech klíčů trvá maximálně 9 dní, ale v soutěži DES-Challenge II-2 byl klíč zjištěn za 56 hodin. Proto byl DES aktualizován a vznikl standard 3DES. (20 str. 5) (21) (22)

3DES (Triple DES nebo TDEA) může mít 3 efektivní velikosti klíčů: 168 bitů, 112 bitů nebo 56 bitů. 3DES funguje tak, že DES proběhne 3x. Zašifruje klíčem 1, dešifruje klíčem 2 a zašifruje klíčem 3. Pokud jsou klíče stejné, má velikost klíče 56 bitů a jedná se o stejné zabezpečení jako v případě standardu DES. Druhou možností je, že klíč 1 a 3 jsou stejné a klíč 2 se liší. Velikost klíče bude 112 bitů. Nebo jsou všechny klíče rozdílné a velikost klíče bude 168 bitů. (23 stránky 6, 13, 21)

Protože DES měl slabý šifrovací klíč a neplnil funkci důvěrnosti dat a jeho nástupce 3DES je neefektivní z důvodu tří průchodů, bylo od něj upuštěno a byl nahrazen standardem AES.

5.3.3.2.AES

AES (Advanced Encryption Standard) je standard, který nahradil předchozí standard DES.

Standard AES splňuje následující kritéria, která jsem převzal z informačního sešitu GCUCMP Crypto-World: „vybraný standard má být velice flexibilní, lehce implementovatelný, má pracovat s 32-bitovým mikroprocesorem, 64-bitovým procesorem, ale i 8-bitovým (v tzv. režimu smart card). AES má být 128-bitová bloková šifra, musí podporovat klíče délky 128, 192 a 256 bitů.“ (24 str. 2)

NIST (National Institute of Standards and Technology) vypsál výběrové řízení podle shora uvedených kritérií pro symetrický šifrovací standard pro 21. století. Přihlásilo se 15 šifrovacích

metod (CAST-256, Crypton, DEAL, DFC, E2, Frog, HPC, LOKI97, Magenta, MARS, RC6, Rijndael, SAFER+, Serpent, Twofish). U šifry LOKI97 byly nalezeny 2 chyby. U šifer Magenta a Frog byly nalezeny teoretické chyby. Do užšího výběru se dostalo 5 šifer (MARS, RC6, Rijndael, Serpent, Twofish). NIST 2. října 2000 ohlásil vítěze, jímž se stala šifra Rijndael. (25) (26)

Rijndael

Rijndael je bloková šifra, kterou společně vytvořili Joan Deamen a Vincent Rijmen. Tento algoritmus je volně šiřitelný a nemůže být patentován. Rijndael má volitelnou délku klíče 128, 192 nebo 256 bitů a velikost bloku 128 bitů, algoritmus má 10, 12 nebo 14 rund podle zvolené délky klíče. Vzhledem k tomu, že je AES nejpoužívanější šifrou, jsou také největší snahy tuto šifru prolomit. Až v roce 2009 se začaly objevovat první útoky na tuto šifru. (25) (27) (28)

Specifikace této šifry je dostupná na adrese: <http://csrc.nist.gov/archive/aes/rijndael/Rijndael-ammended.pdf>

5.3.3.3. Blowfish

Blowfish je bloková šifra vytvořená Brucem Schneierem v roce 1993. Schneier vytvořil tuto šifru jako volně šiřitelnou náhradu za algoritmy DES nebo IDEA. Blowfish má variabilní délku klíče od 32 do 448 bitů a velikost bloku 64 bitů, algoritmus má 16 rund. Pro šifru existují slabé klíče, při kterých je snazší šifru prolomit. Dnes se už nedoporučuje Blowfish používat a doporučuje se nahradit ji novějšími algoritmy jako je AES nebo nástupce Blowfishe - Twofish nebo Threefish. Sám autor v roce 2007 doporučil použít algoritmus Twofish. Při šifrování souborů se nedoporučuje šifrovat větší soubory než 4 Gb. (29) (30)

5.3.3.4. CAST-128

CAST-128 je bloková šifra, kterou společně vytvořili Carlisle Adams a Stafford Tavares v roce 1996. Tento algoritmus je volně šiřitelný. CAST-128 má volitelnou délku klíče 40-128 bitů, přírůstky po 8 bitech a velikost bloku 64 bitů, algoritmus má 12 nebo 16 rund. 16 rund má, pokud je klíč delší než 80 bitů. (31)

Rizikem této šifry je vytvoření krátkého klíče, který umožní relativně rychlé získání nešifrovaného textu hrubou silou

Specifikace této šifry je dostupná na adrese: <http://tools.ietf.org/html/rfc2144>

5.3.3.5.CAST-256

CAST-256 je bloková šifra vycházející z CAST-128, vytvořená pro výběrové řízení na AES, kterou společně vytvořili Carlisle Adams, Stafford Tavares, Howard Heys a Michael Wiener v roce 1998. Tento algoritmus je volně šiřitelný. CAST-256 má volitelnou délku klíče 128 do 256 bitů, přírůsteky po 32 bitech a velikost bloku 128 bitů, algoritmus má 48 rund. 16 rund má, pokud je klíč delší než 80 bitů. (32)

Specifikace této šifry je dostupná na adrese: <http://tools.ietf.org/html/rfc2612>

5.3.3.6.Serpent

Serpent je bloková šifra, kterou společně vytvořili Ross Anderson, Eli Biham a Lars Knudsen v roce 1998. Tento algoritmus je volně šiřitelný a nemůže být patentován. Serpent má volitelnou délku klíče 128, 192 nebo 256 bitů a velikost bloku 128 bitů, algoritmus má 32 rund. Na šifru Serpent byl proveden útok Meet-in-the-middle, Boomerang nebo pomocí lineární kryptoanalýzy. Podařilo se prolomit maximálně 12 z 32 rund algoritmu. Specifikace této šifry je dostupná na adrese: <http://www.cl.cam.ac.uk/~rja14/Papers/serpent.pdf>. (33) (34)

5.3.3.7.Twofish

Twofish je bloková šifra vytvořená Bruce Schneierem v roce 1998. Stejně jako blowfish je tento algoritmus volně šiřitelný. Twofish má variabilní délku klíče od 128 do 256 bitů a velikost bloku 128 bitů, algoritmus má 16 rund. I když v teoretické rovině byl vymyšlen útok, Twofish zatím nebyl ani částečně prolomen. Nejlepší analýzu této šifry provedl Shiho Moriai a Yiquin Lisa Yin v roce 2000, dostupná je na adrese:

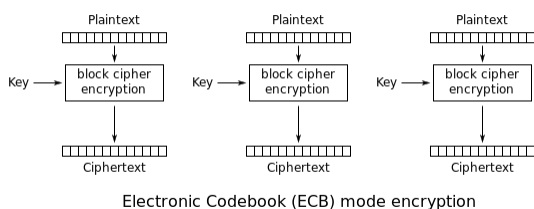
<https://www.schneier.com/twofish-analysis-shiho.pdf>. (35) (36)

5.3.4. Módy:

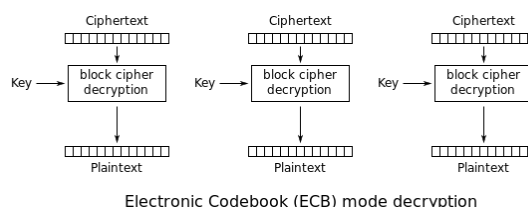
Symetrické šifry používají různé módy šifrování. Jejich výběr má vliv nejen na rychlost zašifrování nebo dešifrování, ale i na bezpečnost webové aplikace

5.3.4.1. ECB (Electronic Codebook Mode)

Zpráva je rozdělena na jednotlivé bloky o stejné velikosti. Každý blok je potom zašifrován algoritmem. Každý blok textu má svůj zašifrovaný blok, proto je možné vytvořit tabulku, kde budou hodnoty k nezašifrovanému i zašifrovanému bloku textu. Jednotlivé bloky se šifrují nebo dešifrují samostatně. (10)



Obrázek 2 - ECB schéma zašifrování Zdroj: (62)



Obrázek 3 - ECB Schéma dešifrování Zdroj: (61)

Výhody

Šifrování i dešifrování textu je možné paralelizovat díky nezávislosti jednotlivých bloků mezi sebou. Také je možné je šifrovat nebo dešifrovat v jakémkoliv pořadí. (10)

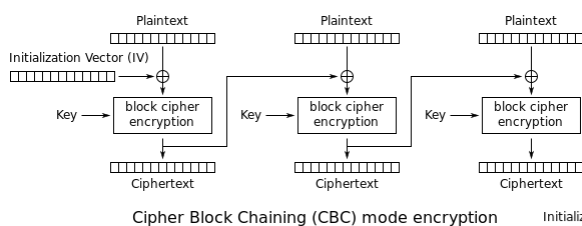
Nevýhody

Stejný blok nešifrovaného textu je vždy převeden do stejného bloku zašifrovaného textu. Tabulku se zašifrovanými bloky je možné sestavit bez znalosti klíče. Díky tomu může útočník změnit obsah zprávy, aniž by to příjemce poznal. (10)

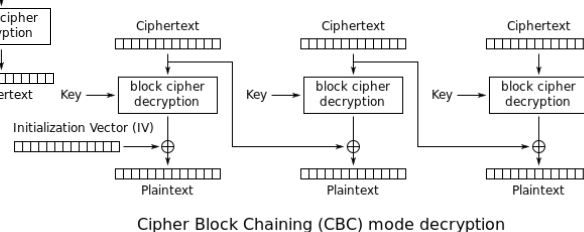
5.3.4.2.CBC (Cipher Block Chaining Mode)

V CBC módu je každý blok před zašifrováním nejdříve XORován předcházejícím zašifrovaným textem. Protože při prvním zašifrování neexistuje předchozí zašifrovaný text, je potřeba vytvořit Inicializační vektor, který by měl být pro každou zprávu znovu generovaný. Zašifrovaný text je nejen použit na výstupu, ale i znovu použit pro XOR dalšího bloku textu. (10)

Dešifrování pracuje tak, že je nejdříve zašifrovaný text dešifrován a poté XORován předchozím zašifrovaným blokem, nebo v případě prvního bloku je použit Inicializační vektor. (10)



Obrázek 4 - CBC schéma zašifrování Zdroj: (60)



Obrázek 5 - CBC schéma dešifrování Zdroj: (59)

Výhody

Dešifrování textu může být paralelizováno, protože není závislé na přechozím dešifrovaném textu.

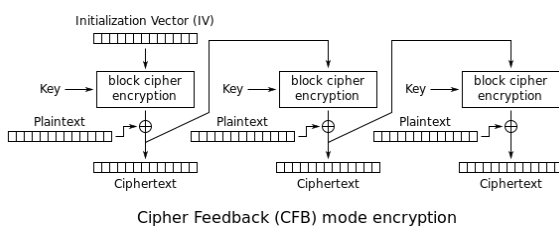
Když dojde ke změně bitu v zašifrovaném bloku textu, bude při dešifrování nečitelný pouze tento a následující blok. Všechny ostatní bloky budou v pořádku. (10)

Nevýhody

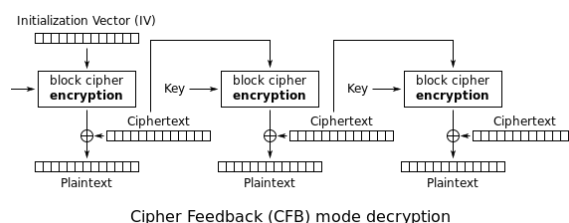
Šifrování je závislé na předchozím zašifrovaném textu, proto ho nelze paralelizovat. Naštěstí se na webových stránkách data zadávají většinou jednou a pak se mnohokrát čtou, takže nemožnost paralelizace šifrování není takový problém.

5.3.4.3.CFB (Cipher Feedback Mode)

Mód CFB je kombinací módu CBC a proudové šifry. Pro začátek šifrování je potřeba mít načtena data celého bloku určeného k zašifrování. Šifrování však nezašifruje celý blok, ale jen jeho část. Předcházející blok textu (Inicializační vektor) je zašifrován a XORován s nezašifrovaným textem. Takto zašifrovaný text jde na výstup a také se použije jako vstup pro další zašifrování. (37) (38)



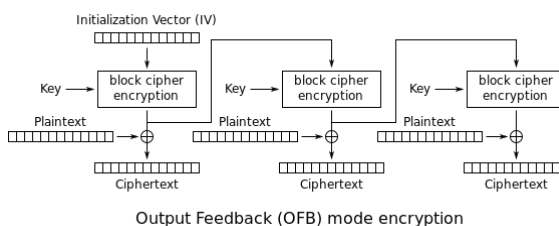
Obrázek 6 - CFB schéma zašifrování; Zdroj: (63)



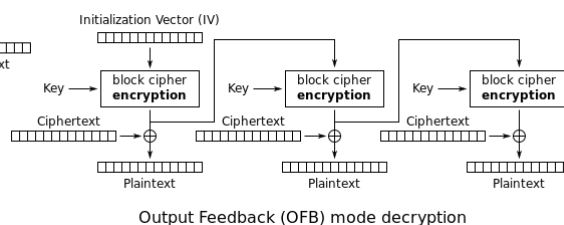
Obrázek 7 - CFB schéma dešifrování; Zdroj: (64)

5.3.4.4.OFB (Output Feedback Mode)

Mód OFB vytváří z blokové šifry šifru proudovou. Pracuje prakticky stejně jako mód CFB, jen pro další vstup se použijí znaky, které ještě nebyly XORovány. (38)



Obrázek 8 - OFB schéma šifrování; Zdroj: (65)



Obrázek 9 - OFB schéma dešifrování; Zdroj: (66)

Módy	Paralelizace šifrování	Paralelizace dešifrování
ECB	Ano	Ano
CBC	Ne	Ano
CFB	Ne	Ano
OFB	Ne	Ne

Tabulka 3 - Módy blokových šifer; Zdroj: (37)

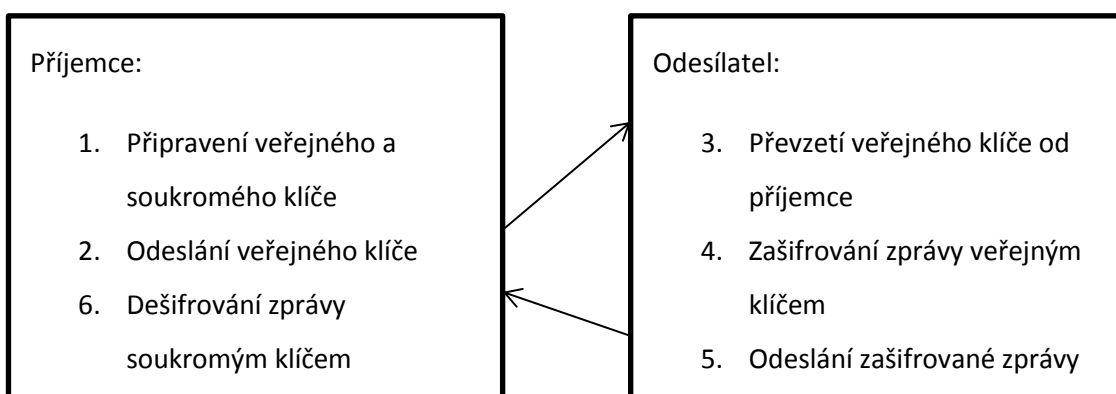
Existují i další módy, pro bližší informace viz anglická wikipedie http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation

5.4.Asymetrické šifry

U asymetrických šifer se používají dva klíče. Jeden veřejný klíč pro zašifrování zprávy a druhý soukromý (tajný) klíč pro její dešifrování. Asymetrické šifrování se používá pro komunikaci po síti mezi osobním počítačem (klient) a serverem. Jak klient, tak server si vygenerují vlastní soukromý a veřejný klíč. Veřejné klíče se předávají nešifrovaným spojením.

Postup předání klíčů funguje následovně. Příjemce zprávy si připraví svůj soukromý a veřejný klíč, veřejný klíč odešle odesílateli zprávy. Ten zprávu zašifruje pomocí veřejného klíče příjemce. Příjemce pomocí soukromého klíče dešifruje zprávu a získá tak text zprávy. (8 str. 128)

Samozřejmě na internetu bývá komunikace obousměrná, takže předání veřejného klíče provedou obě strany. Tento postup je znázorněn na obrázku 10.



Obrázek 10 - Schéma asymetrického šifrování; Zdroj: (8); Úprava: Autor

5.4.1.1.RSA

Autoři šifrovací metody RSA, podle kterých je metoda pojmenována jsou Rivest, Shamir, Adleman. Při šifrování zprávy pomocí RSA dochází k jejímu umocnění veřejným klíčem a při dešifrování k jejímu odmocnění soukromým klíčem.

V knize Matematické základy Informatiky je metoda RSA popsána takto: „Pro libovolná dvě různá prvočísla p a q , číslo r nesoudělné s číslem $(p - 1)(q - 1)$ a číslo $X < p \cdot q$ platí:

$$\text{Je-li } K = [X^r]_{p \cdot q}, \text{ pak } X = [X^s]_{p \cdot q}, \text{ kde } s = [r]^{-1}_{(p-1) \cdot (q-1)}$$

Tvrzení je založeno na tzv. malé Fermatově větě¹ a jejím zobecnění, Eulerově větě².“

5.4.1.2. Eliptické kryptosystémy

Prvními autory, kteří navrhli využití eliptických křivek pro asymetrickou kryptografii, byli Victor Miller a Neal Koblitz v polovině 80. let minulého století. Bezpečnost eliptických systémů závisí na výpočetní složitosti problému řešení diskretního logaritmu pro eliptické křivky. Diskretní logaritmy eliptických křivek je možné vyřešit, pokud je klíč dostatečně krátký. V tabulce 4 na str. 19 je vidět porovnání velikostí klíče mezi různými druhy šifrování. (39 str. 2)

Složitost řešení úlohy eliptického diskretního logaritmu vysvětluje Jaroslav Pinkava v informačním sešitu GCUCMP Crypto-World:

„Zde existuje několik algoritmů, jejichž výpočetní složitost lze popsat ve tvaru druhé odmocniny z N , kde N je počet bodů příslušné eliptické křivky. Jsou to zejména Pollardova p -metoda a Pollardova λ -metoda. Složitost z nich nejefektivnější Pollardovy p -metody je daná výrazem $(\pi N/4)^{1/2}$.“ (39 str. 2)

5.5. Hashovací funkce

Hashovací funkce pracují pouze jedním směrem. Z textu vytvoří například 32 znakový hash (otisk), ze kterého není možné zrekonstruovat předchozí zprávu, proto se hashovací funkce používají pro zašifrování hesla pro přístup do webových aplikací. Při přihlášení se stejným postupem zahashuje zadané heslo a porovná s otiskem v databázi.

Výhodou tohoto řešení je velmi rychlé vytvoření otisku z textu a nemožnost získat původní text z hashe. Přesto je možné získat text, který má stejný otisk.

¹ Pro každé prvočíslu p a každé celé číslo α nesoudělné s p platí $\alpha^{p-1} \equiv 1 \pmod{p}$

² Pro libovolná dvě přirozená nesoudělná čísla α a n platí $\alpha^{\phi(n)} \equiv 1 \pmod{n}$, kde $\phi(n)$ je Eulerova funkce, která pro $n > 1$ udává počet celých kladných čísel menších než n a nesoudělných s n ; $\phi(1) = 1$.

Je-li n prvočíslu, pak $\phi(n) = n - 1$, je-li $n = p \cdot q$, kde p a q jsou různá prvočísla, pak $\phi(n) = (p - 1)(q - 1)$.

Problémem hashových funkcí je možnost vytvoření stejného hashe (kolize) z různých vstupních dat. Vzhledem k tomu, že lze vytvořit hash z jakéhokoliv řetězce textu a výstup z příkladu je vždy 32 znaků dlouhý, tak možných kombinací je limitovaný počet a zákonitě musí existovat stejný otisk pro různé řetězce. (40 str. 20)

Nejdůležitější pro útok na hash je zjistit, jakou hashovací funkcí byl vytvořen. Útoky na hashovací funkce probíhají zkoušením různých řetězců tzv. Brute Force attack (útok hrubou silou). Útočník zkouší jeden řetězec za druhým např. (a, b, ..., aa, ab), dokud nedostane, stejný otisk. (40 stránky 18, 20)

Druhou možností je použít tzv. slovníkový útok, kdy útočník má seznam například nejčastěji používaných hesel nebo všechna česká slova, seznam jmen a příjmení. Při slovníkovém útoku se tedy testují pouze otisky vytvořené z tohoto seznamu, takže útočník nezjistí všechna hesla, ale jen ta, která jsou ve slovníku. (40 str. 19)

Další možností je použít tzv. Rainbow tabulku (nebo si ji vytvořit), kdy se vytvoří tabulka se sloupcem, ve kterém budou uloženy řetězce textu a sloupec s jeho otisky. Tato tabulka je vytvořena před samotným zjišťováním textů a otisk se pak hledá v této tabulce a vrací se hodnota textu. (40 str. 19)

Zabezpečení

Pokud útočník útočí přes webový formulář, je nejjednodušší omezit množství chybně zadaných hesel například na 5 a pak na 5 minut zakázat opětovné přihlášení nebo vyžadovat změnu hesla, ale to je krajní řešení a až příliš restriktivní. Toto opatření odradí útočníky, protože nebudou moci využít skripty a zadávat ručně všechna možná hesla se jim nevyplatí.

Také je možnost odposlouchávat data během přenosu mezi uživatelem a serverem. K zamezení odposlouchávání je potřeba šifrovat data i během přenosu. Viz kapitola 5.9 Bezpečnost přenosu dat

Ve chvíli, kdy útočník získá databázi a chce získat informace, které byly předtím zahashovány, není jisté, zda je dokáže získat. Autor webové aplikace může útočníkovi jeho snahu velmi ztížit. Prvním přístupem, jak ztížit získání informací je kromě samotného vstupu od uživatele přidat ještě nějakou konstantu závislou na uživateli, aby v případě dvou stejných vstupů různých uživatelů byl hash rozdílný. Tomuto zabezpečení se říká solení.

Výběrem nepříliš používaného nebo nového hashovacího algoritmu můžeme také ztížit získání informace z hashe. Další možností je použít pomalé hashovací funkce jako je Bcrypt, PBKDF2 a Scrypt (40 str. 23) (41 str. 2)

Další možností jak ztížit útočnickovi získání informací je použít vícenásobné hashování. Nikde není řečeno, že se musí použít pouze jedna metoda hashe a pouze jednou, klidně mohou být vloženy 3 různé funkce do sebe, např. (hash("sha512",hash ("sha1",hash("md5",\$password)))) Na tuto kombinaci s největší pravděpodobností Rainbow tabulka nebude existovat.

5.6.Vernamova šifra

Tato šifra pracuje pomocí XOR funkce, pomocí které dojde nebo nedojde ke změně bitu. Vernamova šifra je speciální, protože její klíč je stejně dlouhý jako původní zpráva. Teoreticky může být z jakéhokoliv textu náhodně vytvořen jakýkoliv jiný. Podmínkou je úplná náhodnost vygenerování klíče. Bohužel některé funkce negenerují úplně náhodná čísla. (42 str. 266)

5.7.Délka klíče

V tabulce 4 je zobrazeno porovnání délky šifrovacích klíčů mezi různými systémy kryptografie. Podle Katholieke Universiteit Leuven je nejmenší použitelná délka šifrovacího klíče u blokových šifer 80 bitů a odpovídající délka u dalších typů, ale doporučuje se 128 bitová délka. V tabulce 5 je uvedena délka šifrovacího klíče a komentář této university. (43) Pro podrobnosti doporučuji projít celou zprávu university dostupnou na stránce: <http://www.ecrypt.eu.org/documents/D.SPA.17.pdf>

Blokové šifry	Eliptické křivky	RSA/DL (klasický diskretní logaritmus)
56	112	512
64	128	768
80	160	1024
112	224	2048
128	256	3072
192	384	7680
256	512	15360

Tabulka 4 - Srovnatelná bezpečnost různých kryptosystémů při různých délkách klíčů; Zdroj: (39 str. 3)

V dnešní době se už podařilo prolomit testy, které byly zašifrovány metodou RSA a měly klíč dlouhý 768 bitů nebo u šifrování eliptickou křivkou s klíčem o délce 109 bitů (19)

Bezpečnostní úroveň	Délka klíče (bit)	Ochrana	Komentář
1.	32	Prolomení "v reálném čase" jednotlivci	přijatelné pouze pro autentizaci.
2.	64	Velmi krátkodobá ochrana proti malým organizacím	Neměl by být použit pro důvěrná data v nových systémech
3.	72	Krátkodobá ochrana proti středním organizacím, střední ochrana proti malým organizacím	
4.	80	Velmi krátkodobá ochrana proti agenturám, dlouhodobá ochrana proti malým organizacím	Nejmenší univerzální úroveň ≤ 4 ochrany (např. 2-klíčový 3DES $< 2^{40}$ nešifrovaného/šifrovaného textu)
5.	96	Standardní úroveň	2-klíčový 3DES omezený na 10^6 nešifrovaného/šifrovaného textu 10 let ochrany
6.	112	Střednědobá ochrana	20 let ochrany (např. 3-klíčový 3DES)
7.	128	Dlouhodobá ochrana	Dobrá generická aplikační nezávislé doporučení, 30 let
8.	256	"Dohledná doba"	Dobrá ochrana proti kvantovým počítačům, pokud neuplatní Shorův algoritmus

Tabulka 5 - Síla klíče; Zdroj: (43 str. 32)

Cluster s 25 grafickými kartami od společnosti AMD vyzkouší všechna osmiznaková hesla za 5 a půl hodiny. (44) Vzhledem k tomu, že pokud je požadována minimální délka hesla, tak většinou bývá právě osmimístná a uživatelé si nepřidělávají práci s tím, aby si dávali delší hesla. Stačí se podívat na statistiky nepoužívanějších hesel, kterým vévodí hesla jako „123456“ nebo „password“. Nepoužívanější hesla jsou v tabulce 6.

Umístění	Heslo	Porovnání s rokem 2012
1	123456	1
2	password	-1
3	12345678	bez změny
4	qwerty	1
5	abc123	-1
6	123456789	novinka
7	111111	2
8	1234567	5
9	iloveyou	2
10	adobe123	novinka
11	123123	5
12	admin	novinka
13	1234567890	novinka
14	letmein	-7
15	photoshop	novinka
16	1234	novinka
17	monkey	-11
18	shadow	bez změny
19	sunshine	-5
20	12345	novinka
21	password1	4
22	princess	novinka
23	azerty	novinka
24	trustno1	-12
25	0	novinka

Tabulka 6 - Nejpoužívanější hesla; Zdroj: (45)

5.8.Zásady tvorby hesel

Každý uživatel by měl volit takové heslo, aby splňovalo následující kritéria: (46 stránky 4, 5)

- Délka alespoň 8 znaků (znaky z každé skupiny: velká a malá písmena, číslice, speciální znaky)
- Snadno zapamatovatelné a napsatelné
- Nelze heslo uhádnout
- Pro každý přístup jiné heslo
- Heslo měnit každých 6 měsíců

5.9. Bezpečnost přenosu dat

Šifrování dat na straně serveru je v pořádku, ale pokud nejsou data šifrována i po cestě k serveru, nikomu nebrání posílanou zprávu číst nebo i upravovat. K zabezpečení přenosu dat slouží protokol HTTPS s protokolem SSL nebo TLS.

Šifrování přenosu dat je prováděno na základě asymetrického šifrování popsaného v kapitole 5.4.

5.9.1. HTTPS

Běžná komunikace na internetu se provádí pomocí protokolu http, který ovšem neumožňuje žádné zabezpečení komunikace. Proto byl vyvinut protokol HTTPS, který umožňuje zabezpečenou komunikaci pomocí protokolu SSL nebo TLS.

5.9.2. SSL/TLS

Protokol SSL nebo TLS vyžaduje digitální certifikát, což je dokument s informacemi o identitě majitele stránek, doménovém jménu nebo šifrovacím algoritmu pro komunikaci a veřejný klíč, také je zde identifikace certifikační autority. Tento certifikát je vyžadován alespoň na straně serveru z důvodů jeho autentizace.

5.9.3. Certifikační autority

Certifikační autorita je tu proto, aby ověřila data, která majitel zadá do certifikátu. Přenáší se tím odpovědnost z uživatele na certifikační autoritu. Zde je nutná důvěra v certifikační autoritu a její dobrou pověst.

Certifikační autoritou může být i sám autor digitálního certifikátu. Tyto certifikáty vytvářejí v prohlížečích varovné hlášení, ale i když je toto lepší než nezabezpečené spojení, tak raději uživatelé ze stránek odejdou, protože je „nezabezpečená“.

Bez ověřených certifikátů se strany mohou stát obětí tzv. útoku Man in the middle, kdy je mezi serverem a uživatelem ještě útočník, který vzájemnou komunikaci přeposílá tak, aby si strany myslely, že komunikují s protistranou a zároveň komunikaci odposlouchává, případně ji může měnit. (47)

5.10. Bezpečnost webové aplikace

I když budeme mít zabezpečené spojení se serverem, stále můžou být chyby přímo v aplikaci na webových stránkách.

5.10.1. SQL injection

Dříve velmi častou a stále se opakující chybou je neošetření vstupu pro dotaz do databáze. Její ošetření je jednoduché a lze použít víc variant ošetření. Jednou z možností je použít funkci `mysql_real_escape_string()`, nebo se spolehnout na `magic_quotes_gpc`, která se zajistí přidáním jednoduchých uvozovek na začátek a konec proměnné. (48)

Na stránkách Jakuba Vrány je následující zdrojový kód:

```
„<?php
// ošetření vstupních dat
mysql_query("SELECT * FROM tabulka WHERE nazev = ' " .
mysql_real_escape_string($_GET["nazev"]) . "' OR id = ' " .
intval($_GET["id"]));

// spolehnutí se na magic_quotes_gpc
mysql_query("SELECT * FROM tabulka WHERE nazev = '$_GET[nazev]'
OR id = '$_GET[id]'");" (48)
```

5.10.2. Cross site Scripting (XSS)

Cross site scripting je metoda útoku, kdy se útočníkovi podaří pozměnit obsah webové stránky tak, že se provede jeho JavaScriptový kód. Proto je potřeba ošetřit data, která může útočník měnit, jako jsou metody GET a POST, data z cookie, session, a také je potřeba ošetřit data, která jsou zapisována do databáze. (49)

Ošetření probíhá již zmíněnými funkcemi `mysql_real_escape_string()`, `magic_quotes_gpc` nebo funkcí `htmlspecialchars()`

```
htmlspecialchars($_POST['subject'], ENT_QUOTES, "UTF-8");
```

5.10.3. Cross-Site Request Forgery (CSRF)

Útočník na základě odhadu nebo znalosti struktury webové aplikace vyšle požadavky, které mění nebo mažou záznamy v databázi. Obranou proti tomuto útoku je používat pro změny v databázi výhradně metodu POST. Tato obrana znesnadní, ale ještě nevyloučí tento útok. Pro obranu je potřeba použít ochranu například zjištěním hlavičky Referer. Další možností je autorizační token, kdy se vygeneruje náhodný řetězec, který se uloží do databáze a při každé akci se tento autorizační token kontroluje. (50)

6. Uživatelský komfort

Důležitou složkou každé webové stránky musí být její uživatelská přívětivost. Pokud by totiž web vyžadoval příliš mnoho akcí, uživatel by mohl používat jiný, méně bezpečný web, který by ho tolik neobtěžoval.

V každém případě je potřeba mít jedno heslo pro přístup do rozhraní pro přístup do zašifrované databáze, která může být zašifrována jedním hlavním šifrovacím klíčem, nebo každý uživatel bude mít své heslo, které bude šifrovat/dešifrovat svou část dat v databázi. Další možností je zadávat pro každou zprávu jiný šifrovací klíč.

Prvním důležitým parametrem při hodnocení je, k čemu web slouží. Pokud web bude provádět rozesílání zašifrovaných emailů, bude mít jiné požadavky než web, který vede šifrovanou databázi uživatelů.

Dalším parametrem je, kolik lidí bude mít možnost prohlížet informace v databázi. Jinak bude řešeno prohlížení ve stylu MLM struktury, kdy vedoucí chce mít přehled o aktivitách podřízených a jinak již zmíněné rozesílání emailů nebo aplikací typu Facebook.

Zásadní vliv na funkčnost a přívětivost webu má počet hesel. Pokud bude pouze jedno heslo společné s přístupem do databáze, tak v případě jeho ztráty jsou ztracena i veškerá data uživatele na webu. Osobně mi přijde, že mít společné heslo pro přístup i šifrování je bezpečnostní chyba a proto budu vždy uvažovat se dvěma hesly. Heslo však může mít i podobu jednoho hlavního hesla, které šifruje stejně pro všechny uživatele. Všechny typy řešení šifrovacích klíčů, popsanych v následujících kapitolách, jsem vložil do tabulky 7 na straně 27.

6.1.Šifrování společným klíčem

Databáze je šifrována jedním hlavním heslem, které vytvoří majitel stránek nebo administrátor. Toto řešení má své výhody, ale má i svá rizika.

Jednoznačnou **výhodou** je, že uživatel nikdy nezadává žádné heslo pro rozšifrování dat. Další výhodou je možnost rozšifrovat data v případě žádosti od soudu. V případě zapomenutí hesla pro přístup do rozhraní není problém vygenerovat heslo nové a umožnit znovu přistupovat k datům.

Nevýhody jsou jasné - uživatel nepozná, zda jsou data šifrována. Vzhledem k tomu, že šifrovací klíč je známý pro autora stránek, může data prohlížet i administrátor. Nelze také

ovlivnit sílu šifrovacího klíče. V případě zjištění šifrovacího klíče útočníkem, má útočník přístup do celé databáze.

6.1.1. Hodnocení

Toto řešení má více nevýhod než výhod a je založeno na komfortu pro uživatele, ne na vysokém zabezpečení dat. Proto ho považuji za vhodný pouze pro použití, kdy je potřeba sdílet informace s více lidmi. Jako vhodný příklad bych viděl data na sociálních sítích nebo diskuzních fórech.

6.2.Šifrování vlastním klíčem

V tomto řešení má každý uživatel svůj vlastní šifrovací klíč. Nejschůdnější řešení je zašifrovat tento klíč do databáze pomocí klíče pro přístup do rozhraní, protože v případě zapomenutí šifrovacího klíče dat je možné ho získat z databáze pomocí znalosti hesla do rozhraní. Změna šifrovacího klíče dat znamená dešifrovat všechna data uživatele pomocí starého klíče a poté zašifrovat novým. Tato operace v případě velkého množství dat je časově náročná a může být i neproveditelná. Záleží na nastavení časových limitů serveru.

Pokud uživatel zapomene heslo pro přístup do rozhraní je možné se znalostí šifrovacího klíče dat zaslat heslo nové bez ztráty dat.

V případě zapomenutí hesla i šifrovacího klíče není možno uživateli pomoci bez ztráty zašifrovaných dat.

Nakonec se podíváme na výhody a nevýhody tohoto řešení.

Nejzákladnější **výhodou** je bezpečnost, která závisí pouze na šifrovacím klíči, který zadá uživatel. Pokud je klíč dostatečně silný, tak se k datům dostane pouze vlastník dat.

Nevýhodou řešení pomocí vlastního klíče je, že uživatel může použít slabý šifrovací klíč, toto se dá ale omezit například minimální délkou klíče. V případě ztráty hesla a šifrovacího klíče není možné získat data. Co se týče uživatelského komfortu, je možné po uživateli požadovat šifrovací klíč při každém požadavku na zašifrovaná data nebo pokud nejde o nějaká tajná data, je asi vhodnější si pamatovat šifrovací klíč po určitou dobu a uživateli tak zpříjemnit práci na webu.

6.3.Šifrování zadávaným klíčem

Další variantou je použít pokaždé jiný klíč pro zašifrování dat. Tato varianta se hodí v případě rozesílání zpráv různým osobám, jako je například email.

S tímto systémem jsou spojeny stejné výhody a nevýhody jako se systémem vlastního klíče, ale navíc zde existují další výhody a nevýhody

Menší **výhodou** je, že v případě získání šifrovacího klíče, útočník získá data pouze z určité skupiny zašifrovaných zpráv, pokud se samozřejmě šifrovací klíče budou lišit.

Velkou **nevýhodou** tohoto řešení je vysoké riziko zapomenutí šifrovacích klíčů. Uživatelé pak mohou mít tendenci volit kratší klíče, které se lépe pamatují. Také může klíče opakovat pro všechny zprávy. Také zde vzniká problém s předáváním dešifrovacích klíčů.

6.4.Soukromý a veřejný klíč

Poslední variantou je využít asymetrické kryptografie.

Výhodou je, že soukromý i veřejný klíč je uložen v databázi a zprávy se proto šifrují a dešifrují bez zásahu uživatele a tím se zvyšuje uživatelův komfort, také uživatel nemůže šifrovací klíč zapomenout.

Menší **nevýhodou** tohoto řešení je nutnost mít v databázi uloženy klíče. Všechny klíče je potřeba dešifrovat z databáze, ale nepředpokládám, že by tato data měla významný vliv na velikost databáze. Také uživatel nepozná, zda jsou data šifrována.

6.4.1. Hodnocení

Šifrovací klíč	Výhody	Nevýhody
Společný klíč	• Uživatelský komfort • Přístup k datům pro policii a soudy	• Nelze rozpoznat šifrování • Zobrazení dat administrátorům (možnost zneužití dat) • Žádný vliv na šifrovací klíč
Vlastní klíč	• Nelze získat data bez šifrovacího klíče • Vlastní šifrovací klíč	• Zapomenutí šifrovacího klíče a možná ztráta dat • Uživatelský komfort
Zadávaný klíč	• Nelze získat data bez šifrovacího klíče • Vlastní šifrovací klíč • Při získání klíče má útočník přístup pouze k určité skupině zpráv	• Zapomenutí šifrovacího klíče a možná ztráta dat • Uživatelský komfort • Problém s předáním šifrovacího klíče • Riziko slabších klíčů
Veřejný klíč a soukromý klíč (Asymetrická kryptografie)	• Uživatelský komfort • Nelze získat data bez soukromého klíče	• Potřeba v databázi držet soukromý klíč a generovat veřejný klíč pro každé spojení • Nelze rozpoznat šifrování • Žádný vliv na šifrovací klíč

Tabulka 7 - Výhody a nevýhody typů klíčů; Zdroj: Autor

6.5. Osobní databáze

Osobní databáze je databáze s daty, ke kterým má přístup pouze uživatel. Pro takovouto databázi může být použito řešení jak se společným heslem, tak i s vlastním heslem, záleží na typu dat. Pokud je však data potřeba šifrovat, nemá smysl uvažovat o řešení, ke kterému má přístup i někdo jiný.

6.6. Databáze uživatelů

Databáze uživatelů se používá na všech webových stránkách, kde je možnost registrovat uživatele. Přístup do této databáze má pouze vlastník dat a majitel stránek. Zde jde především o registrace do diskuzních fór, nebo internetových obchodů.

Záleží na použití na webu, zda se hodí systém s vlastními hesly každého uživatele. Není vhodné asymetrické šifrování.

6.7. MLM systém prohlížení

V tomto případě je důležité prohlížet informace napříč strukturou zaměstnanců směrem od vedoucího k podřízeným. Vedoucí má přístup do databáze svojí a svých podřízených, podřízení by neměli mít přístup k datům vedoucího.

V této databázi je potřeba, aby v datech o podřízeném byla informace, kdo je nadřízený a šifrovací klíč podřízeného byl zašifrován šifrovacím klíčem nadřízeného.

Slabým místem systému je to, že v případě odcizení přístupu vedoucího do rozhraní, má útočník přístup i do struktury pod vedoucím. Také je zapotřebí další sloupec v databázi pro zašifrovaný šifrovací klíč.

Zde je potřeba, aby účet podřízeného potvrzoval jeho vedoucí, aby mohl zadat svůj šifrovací klíč pro zašifrování šifrovacího klíče podřízeného. Vedoucí, ale nemusí znát šifrovací klíč podřízeného. Změna šifrovacího klíče podřízeného bude muset být opět potvrzována nadřízeným. Při změně šifrovacího klíče nadřízeného probíhá změna bez potvrzení pouze v případě, pokud nemá žádného vedoucího. Výhodou potvrzování registrací nadřízeným je, že se do systému nedostane cizí osoba.

6.8.Sociální síť, diskuze

Pokud jde o příspěvky na sociálních sítích a diskuzích je zde situace složitá vzhledem k tomu, že příspěvky jsou viditelné veřejně nebo pro přátele. U sociálních sítí není nutno data šifrovat, protože pokud si informace může zobrazit v podstatě kdokoli, není potřeba je chránit šifrováním.

Samozřejmě, že možnost šifrovat data společným klíčem je možná, ale jak již bylo zmíněno výše, je to zbytečné

6.9.Email

Při odesílání emailů mohou nastat dvě situace. První je odeslání emailu do stejné emailové služby a druhá možnost je odeslání do cizí sítě.

6.9.1. Vlastní síť

Pokud jde o odeslání emailu uživateli stejné sítě, je situace jednodušší. Systém nejdříve zprávu zašifruje společným klíčem a až se přihlásí uživatel, kterému má být zpráva doručena, tak se dešifruje společným klíčem a zašifruje jeho vlastním klíčem.

Druhou možností je využití asymetrické kryptografie, kdy by v databázi příjemce byl uveden zašifrovaný soukromý klíč a také nezašifrovaný veřejný klíč, kterým by se zpráva zašifrovala. Ve chvíli, kdy by se uživatel přihlásil, tak by se dešifroval soukromý klíč z databáze. Záleží na implementaci v aplikaci, zda budou emaily stále zašifrovány pomocí asymetrické kryptografie nebo dojde při přihlášení k přešifrování na symetrickou šifru. Viz další odstavec.

U společného klíče a nutnosti přešifrovat emaily na šifrovací klíč uživatele může dojít, v případě dlouhodobější absence uživatele nebo velkého počtu přijímaných zpráv, k delšímu přihlášení v závislosti na počtu doručených zpráv.

Třetí variantou je použít úplně jiný šifrovací klíč a poté ho při doručení požadovat po příjemci. Zde je důležité předat šifrovací klíč příjemci jinou cestou, nejlépe osobně.

Dokud je zpráva zašifrována společným klíčem, jsou zde nevýhody zmíněné v tabulce 7. V případě zašifrování zadaným klíčem nevýhody společného klíče nejsou, ale jsou zde jiné problémy, které jsou také uvedeny v tabulce 7. Nejlepším řešením je proto využití asymetrické kryptografie s veřejným klíčem.

6.9.2. Cizí síť

V případě cizí sítě je situace jasná. Pokud chci zprávu zašifrovat uživateli jiné sítě, musím zadat šifrovací klíč ručně a oznámit ho osobě jinou cestou. Příjemci pak dojde zpráva v zašifrované podobě a je potřeba ji dešifrovat na základě předaného klíče. Zde by bylo vhodné, aby poskytovatel služby šifrovaných emailů měl i veřejnou stránku pro dešifrování zpráv posílaných do jiných emailových schránek, kam by se vyplnila zpráva a dešifrovací klíč.

V případě doručení emailu z cizí sítě se zpráva zašifruje pomocí společného klíče a po přihlášení se dešifruje společným klíčem a zašifruje vlastním klíčem.

Dokud je zpráva zašifrována veřejným klíčem, jsou zde nevýhody uvedené v tabulce 7.

Shrnutí požadavků podle funkčnosti stránek je v tabulce 8.

Funkce webové stránky	Šifrovací klíč	Požadavky
Osobní databáze	• Společný • Vlastní	
Databáze uživatelů	• Společný	
MLM systém prohlížení	• Vlastní	• Potvrzování změn nadřazeným • Sloupec s klíčem pro vedoucí
Sociální síť, diskuze	• Žádný • Společný	
Email	• Kombinace společného a vlastního klíče	
Email	• Soukromý a veřejný klíč	2 sloupce u uživatele se zašifrovaným soukromým a nezašifrovaným veřejným klíčem
Email	• Zadávaný	

Tabulka 8 - Souhrn výhod a nevýhod podle funkčnosti; Zdroj: Autor

7. Testovací systém Raspberry Pi

Raspberry Pi je velmi rozšířený mini PC, který má poměrně velkou komunitu, která s ním vytváří různé projekty a mezi nimi je právě i webový server. Jeho rozšířenost je z důvodu nízké pořizovací ceny (zhruba 1 000 Kč) a velké možnosti rozšíření díky různým vstupům.

7.1.Specifikace

Specifikace systému Raspberry Pi je převzata ze stránek e-shopu (51), který Raspberry Pi prodává a je doplněná o typ SD karty a vybraného systému. Celá specifikace je zobrazena v tabulce 9.

Procesor	Broadcom ARM1176JZF-S (armv6k) 700 MHz
Grafika	Broadcom VideoCore IV (Open GL ES 2.0,
Paměť	512MB
Video výstup:	HDMI, Kompozitní RCA,
Audio výstup:	3,5mm jack, HDMI, I2S audio
USB 2.0 2x	
SD/MMC/SDIO slot	
10/100mbs Ethernet Adapter	
8x GPIO, UART, I2Cm SPI, I2S audio	
SD karta:	Kingston SDHC 16GB UHS-I Class 10 Ultimate (52)
Systém:	Raspbian build ze dne 7. 1. 2014 (53)

Tabulka 9 - Specifikace Raspberry Pi (51); Zdroj: Autor

Operační systém nainstalovaný na SD kartě je Raspbian, založený na systému Debian, díky tomu se používají stejné příkazy v příkazové řádce jako v Debianu. Také nebylo nutno psát operační systém od začátku, ale pouze optimalizovat pro ARMv6 instrukce.

Operační systém je potřeba stáhnout, nejlépe ze stránek Raspeberrypi.org, rozbalit archiv a například pomocí programu Win32 Disk Imager (54) nakopírovat na SD kartu. Kopírovat data na SD kartu z archivu pomocí průzkumníka sice lze, ale Raspbian nebude fungovat

7.2.Nastavení

Při prvním spuštění naběhne systém do konfiguračního režimu, kde se dají nastavit parametry systému. V této části bude uvedeno, jaké změny byly provedeny v tomto režimu.

První změnou bylo rozšíření systému přes celou SD kartu, aby systém nebyl limitován přednastavenou velikostí oddílu SD karty. Dále bylo přenastaveno bootování místo do grafického rozhraní do konzole a login, protože grafické rozhraní není potřeba pro práci webového serveru a zbytečně by vytěžovalo poměrně pomalý procesor. Vždy se do grafického rozhraní dá přejít příkazem z konzole. Dále byla změněna časová zóna a lokalizace systému do češtiny - cs_CZ.UTF-8

Vzhledem k tomu, že je změněno rozhraní na příkazovou řádku, byla snížena také paměť pro GPU z 64MB na 16MB. Navíc výpočty šifrování probíhají na procesoru a ne na grafické kartě, takže procesor nebude tolik limitován velikostí operační paměti.

Nakonec proběhla aktualizace systému pomocí příkazů:

```
sudo apt-get update  
sudo apt-get upgrade
```

7.3.Instalace webového serveru

Pro instalaci webového serveru se do konzole zadá příkaz, který jsem převzal z knihy Raspberry Pi Uživatelská příručka:

```
„sudo apt-get install apache2 php5 php5-mysql mysql-server  
phpmyadmin“ (55 str. 123)
```

Během instalace byla zadána položka automatického nastavení webového serveru pro běh phpmyadmin na apache2

Všechny balíčky byly aktualizovány k 5. 5. 2014. Verze balíčků jsou zobrazeny v příloze 1.

7.4.Změna adresáře na plochu

Z důvodů bezpečnostní politiky operačních systémů založených linuxových systémů nelze manipulovat se soubory ve složkách systému pod běžným uživatelským účtem. Je potřeba být přihlášen jako administrátor. Změna souboru, kde je nastavena cesta k souborům webového serveru se provede příkazem:

```
sudo nano /etc/apache2/sites-enabled/000-default
```

Cesta do adresáře webového serveru byla změněna na „home/pi/Desktop/www“. Poté stačí soubor uložit a zavřít. Poslední reboot Raspberry Pi a systém je připraven pro testování.

7.5. Metodika testu

Testování probíhá pomocí PHP funkcí pro měření času, které zapíší čas šifrování nebo dešifrování do databáze a pomocí interního testeru Apache - utilitu AB pro simulaci výkonu a měření zátěže. (56)

Testování bude probíhat v těchto třech modelových situacích, kde bude sledována zátěž, doba zašifrování nebo dešifrování a následně vliv na velikost databáze.

7.5.1. Šifrovací metody

Funkce `mcrypt` v PHP zvládá tyto šifrovací metody seřazené podle abecedy zobrazené v tabulce 10, které jsem získal pomocí příkazu:

```
mcrypt_list_algorithms();
```

Šifrovací metoda
arcfour
blowfish
blowfish-compat
cast-128
cast-256
des
enigma
gost
loki97
rc2
rijndael-128
rijndael-192
rijndael-256
saferplus
serpent
tripledes
twofish
wake
xtea

Tabulka 10 - Šifrovací metody v PHP na Raspberry Pi; Zdroj: (57)

Šifrovací metoda
blowfish
cast-128
cast-256
rijndael-128
rijndael-192
rijndael-256
serpent
tripledes
twofish

Tabulka 11 - Vybrané šifrovací metody; Zdroj: Autor

Z šifrovacích metod z tabulky 10 jsem se rozhodl otestovat šifry Blowfish, cast-128, cast-256 rijndael-128, rijndael-192, rijndael-256, Serpent. TripleDES, Twofish. Všechny vybrané šifry jsou uvedené v tabulce 11.

7.5.1.1. Důvod výběru

Prvním omezením, které jsem musel respektovat, bylo omezení na straně PHP. Funkce mcrypt() může použít 19 různých algoritmů. Mě zajímal vliv rozdílné velikosti šifrovacího klíče u algoritmu Rijndael. Také jsem chtěl změřit rychlost algoritmů, které se dostaly do druhého kola při výběru algoritmu pro AES. Z nich je bohužel dostupný v PHP jen algoritmus Serpent a Twofish. Vzhledem k tomu, že AES je nástupce DES, zařadil jsem do testu bezpečnější variantu DES a to 3DES. Když už jsem u předchůdců, zařadil jsem algoritmus Blowfish, jehož nástupce je již zařazený Twofish. Jako poslední jsem zařadil CAST-128 a CAST-256.

Algoritmus Blowfish jsem zařadil, protože mě zajímal rozdíl proti jeho nástupci Twofish, ale Blowfish není vhodný z bezpečnostních důvodů používat.

7.5.1.2. Šifrovací třída v aplikaci

Třidu pro šifrování dat v aplikaci jsem našel již cca před 18 měsíci na webové stránce <http://4webmaster.cz/php-mysql/sifrovani-t374.html>. V době psaní bakalářské práce již odkaz není funkční, proto se odkazuji na archiv webových stránek (58). Tuto třídu jsem upravil, protože u některých vstupů nefungovala správně. Třidu classCrypt.php naleznete na přiloženém CD ve složce PHP/class/.

7.5.2. Vstupní data

Všechna vstupní data budou předem vygenerována pomocí funkce generuj (\$delka) a zapsána do databáze. Před začátkem šifrování budou data načtena z databáze.

```
public function generuj ($delka){
    $heslo = "";
    $pole = '0123456789_-
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';

    for ($i=0; $i < $delka; $i++) {
        $heslo .= substr($pole, mt_rand(0, strlen($pole) -1), 1);
    }
    return $heslo;
}
```

7.5.3. Šifrovací klíč/dešifrovací klíč

Šifrovací klíč bude pro všechna měření stejný. Bude použit klíč o délce 15 znaků, který bude obsahovat malá i velká písmena, číslice a zvláštní znaky. Klíč je: **v9Pc_7pH%2aCvnX**

7.5.4. Mód

Mód blokových šifer bude pro všechna měření CBC, proto je potřeba pro každou šifrovací metodu vytvořit Inicializační vektor.

7.5.5. Inicializační vektor

Z důvodu nutného stejného inicializačního vektoru pro zašifrování a dešifrování jsem se rozhodl vygenerovat jeden inicializační vektor, který bude použit pro všechna šifrování a dešifrování vybranou šifrovací metodou. Nepředpokládám vliv různých inicializačních vektorů na rychlost šifrování vzhledem k jeho stále stejné délce.

Šifrovací metoda	Inicializační vektor
3DES	2bd15252e40348bb
Blowfish	f8aa35759406bc78
CAST-128	64bb5e7bcb92d29a
CAST-256	fe9c8feffca12a0d35a0b327f1013467
Rijndael-128	3da562c11cd48e79175e8a8d75a85d28
Rijndael-192	8221a062a5bf9730b2c6adebf3a0ebbd3133d84cb1e77ebf
Rijndael-256	912ada5a708090aaee43f94c159cde276a3386a3f730baa94439c1a058786471
Serpent	0d26ded0512fa7786001706e840440d9
Twofish	ce439597b436b0994652ab49cbfe8dcd

Tabulka 12 - Inicializační vektory šifrovacích metod; Zdroj: Autor

7.5.6. Typy testů

7.5.6.1. Zátěžový test

V tomto testu se bude testovat text o takové délce, aby test probíhal alespoň 30 sekund na AES-256. Pro tento test budou vygenerovány tři bloky textu a všechny budou zašifrovány vybranými šifrovacími funkcemi.

Měření proběhne pomocí PHP funkcí pro měření času, kdy se časy zapíší do databáze. (Soubory: zatezSifrovani.php; zatezDesifrovani.php)

7.5.6.2.Blok textu (email)

Tento test bude simulovat vytížení serveru při použití, například v emailové komunikaci. Bude vytvořen text představující předmět emailu o délce 20 znaků a text o délce 1000 znaků. Pro toto měření bude vygenerováno 100 emailů a všechny budou zašifrovány vybranými šifrovacími funkcemi.

Měření proběhne pomocí PHP funkcí pro měření času, kdy se časy zapíší do databáze (Soubory: emailSifrovani.php; emailDesifrovani.php). Následně bude provedeno druhé měření pomocí AB a to jak při šifrování do databáze a dešifrování z databáze, tak při zápisu nešifrovaného textu do databáze a čtení nešifrovaného textu z databáze (Soubory: emailMSifrovani.php; emailMDesifrovani.php; bezZEmail.php; bezEmail.php). Při čtení bude simulovat zátěž 20 uživatelů při čtení jednoho emailu s celkovým počtem 500 požadavků a při zápisu 5 uživatelů s celkovým počtem 100 požadavků.

7.5.6.3.Více krátkých textů (uživatel)

Tato část testu je zaměřena na šifrování 9 textů o délce 8 znaků. Jedná se o simulaci zápisu například uživatelů do databáze. Měření proběhnou stokrát pro každou vybranou šifrovací funkci.

Měření proběhne pomocí PHP funkcí pro měření času, kdy se časy zapíší do databáze (Soubory: uzivatelSifrovani.php; uzivatelDesifrovani.php). Následně bude provedeno druhé měření pomocí AB a to jak při šifrování do databáze a dešifrování z databáze, tak při zápisu nešifrovaného textu do databáze a čtení nešifrovaného textu z databáze (Soubory: uzivatelSifrovani.php; uzivatelDesifrovani.php; bezZUzivatel.php; bezUzivatel.php). Při čtení bude simulovat zátěž 20 uživatelů při zobrazení jednoho uživatele s celkovým počtem 500 požadavků a při zápisu 5 uživatelů s celkovým počtem 100 požadavků.

7.6.Vlastní měření

Při měření jsem se zaměřil na 3 parametry:

- velikost databáze
- rychlost šifrování - jeden uživatel
- rychlost šifrování - více uživatelů

7.6.1. Velikost databáze

Po každé šifrovací metodě jsem zapsal do tabulky velikost jednotlivých tabulek v databázi. Nárůst velikosti databáze při šifrování dat je v případě tabulky s emaily o 78 %, u tabulky uživatelů je nárůst o 200 % a u zátěžového testu o 103 %. Změna šifry nemá vliv na velikost databáze.

Tabulka	Velikost
email	144 KiB
semail	256 KiB
suzivatel	48 KiB
szatez	29,5 MiB
uzivatel	16 KiB
zatez	14,5 MiB

Tabulka 13 - Velikost tabulek v databázi; Zdroj: Autor

7.6.2. Rychlost šifrování - jeden uživatel

Tento test probíhal v prohlížeči Midori. Spustil jsem soubor testování.php, jehož zdrojový kód je vypsán níže. Poté jsem spustil soubor kontrola.php, zda jsou dešifrované soubory totožné s původními texty a následně jsem z databáze vypsál časy šifrování a dešifrování pomocí souboru cas.php. Pak jsem vymazal data z databáze pomocí příkazu TRUNCATE, změnil šifrovací metodu v souboru classCrypt a opakoval měření.

7.6.2.1.Zdrojový kód - testování.php

```
require_once './class/classCrypt.php';
require_once './class/classSQL.php';

set_time_limit(3600);

require_once './uzivatelSifrovani.php';
require_once './uzivatelDesifrovani.php';
echo "konec 1";

require_once './emailSifrovani.php';
require_once './emailDesifrovani.php';
echo "konec 2";

require_once './zatezSifrovani.php';
require_once './zatezDesifrovani.php';
echo "konec 3";
```

7.6.2.2.Zdrojový kód - kontrola.php

```
require_once './class/classCrypt.php';
require_once './class/classSQL.php';

set_time_limit(3600);

require_once './kontrolaEmail.php';
echo "konec Emailů";

require_once './kontrolaUzivatel.php';
echo "konec Uživatelů";

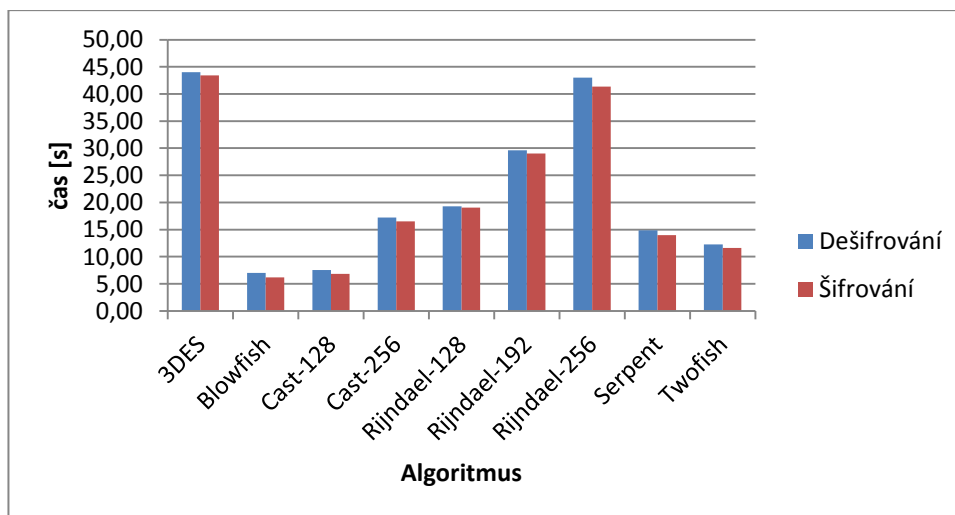
require_once './kontrolaZatez.php';
echo "konec Zátěže";
```

7.6.2.3.Zdrojový kód - cas.php

```
require_once './class/classCrypt.php';  
require_once './class/classSQL.php';  
  
set_time_limit(3600);  
  
require_once './casEmail.php';  
echo "konec Emailů";  
  
require_once './casUzivatel.php';  
echo "konec Uživatelů";  
  
require_once './casZatez.php';  
echo "konec Zátěže";
```

7.6.2.4.Zátěž

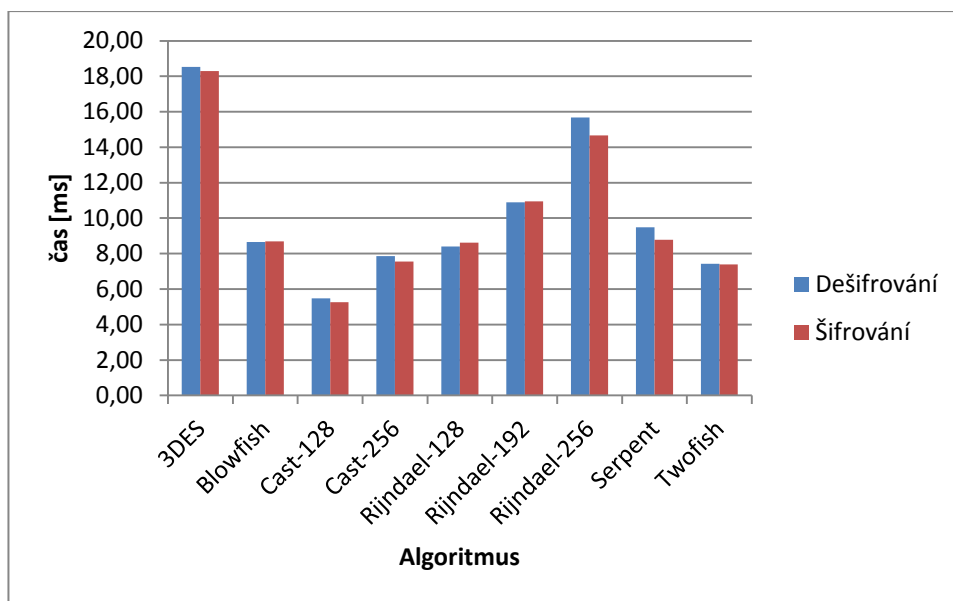
Test zátěže eliminuje rychlost přípravy klíčů algoritmu a ukazuje jen rychlost operací při šifrování/dešifrování. V grafu 1 jsou zobrazena data z měření zátěže. Nejrychlejší operace má Blowfish a CAST-128, zatímco nejpomaleji pracuje 3DES a Rijndael-256.



Graf 1 - Rychlost šifrování při zátěži; Zdroj: Autor

7.6.2.5.Email

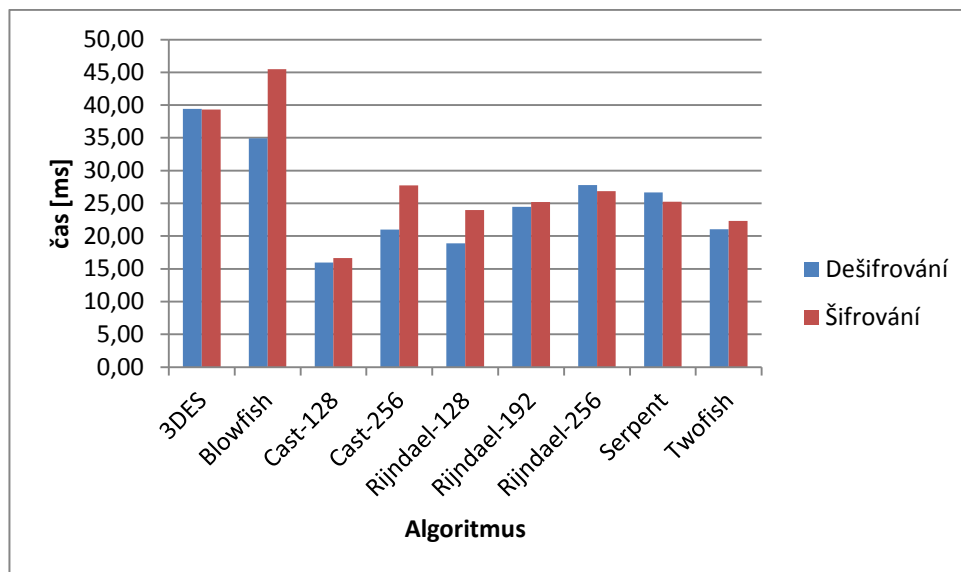
Tento test by měl ukázat rychlost algoritmu při reálném použití. V grafu 2 jsou zobrazena data z měření emailu. Čas šifrování i dešifrování se pohybuje přibližně mezi 6 až 18 ms, což je sice zanedbatelný čas, ale ukazuje rozdíl mezi jednotlivými algoritmy až 300 %. Nejpomalejším algoritmem je 3DES a nejrychlejším CAST-128.



Graf 2 - Rychlost šifrování emailů; Zdroj: Autor

7.6.2.6.Uživatel

Zajímavá je skutečnost, že čas potřebný k šifrování a dešifrování uživatelů je delší než v případě 1000 znakového emailu. V grafu 3 jsou zobrazena data z měření simulace uživatele. Řekl bych, že je to kvůli počáteční přípravě klíčů, která je delší než samotné šifrování/dešifrování. Také se ukázal významný rozdíl mezi šifrováním a dešifrováním u Blowfish, CAST-256 a Rijndaelu-128. U Rijndaelu jsou rozdíly v grafu mezi jednotlivými délkami klíčů stejné.



Graf 3 - Rychlost šifrování uživatelů; Zdroj: Autor

7.6.3. Rychlost šifrování - více uživatelů

Tento test probíhal z příkazové řádky, kde jsem spustil dávkový soubor prikazy.sh s příkazy funkce AB a výstupy do souborů. Pak jsem vymazal data z databáze pomocí příkazu TRUNCATE, změnil šifrovací metodu v souboru classCrypt.php a opět spustil soubor prikazy.sh.

7.6.3.1. Spuštění dávkového souboru

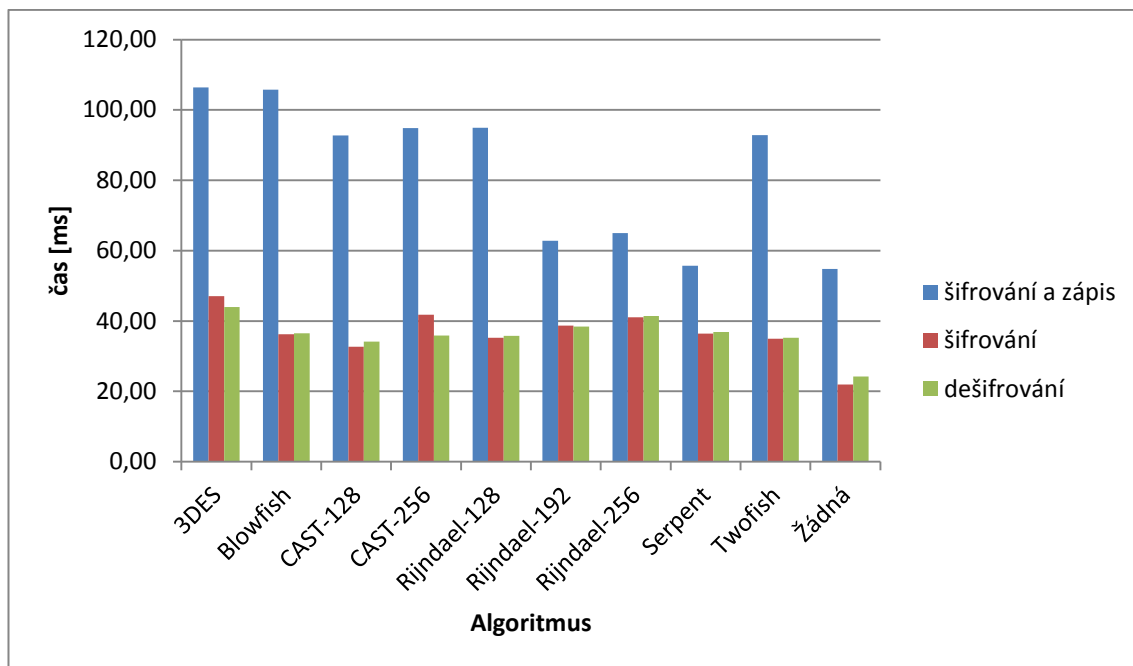
```
bash prikazy.sh
```

7.6.3.2. Zdrojový kód - prikazy.sh

```
ab -n 100 -c 5 http://localhost/bezUZivatel.php >>uBZ.txt
ab -n 100 -c 5 http://localhost/uzivatelMSifrovani.php >>uMS.txt
ab -n 500 -c 20 http://localhost/bezUZivatel.php >>uB.txt
ab -n 500 -c 20 http://localhost/uzivatelMDesifrovani.php >>uMD.txt
ab -n 100 -c 5 http://localhost/bezZEmail.php >>eBZ.txt
ab -n 100 -c 5 http://localhost/emailMSifrovani.php >>eMS.txt
ab -n 500 -c 20 http://localhost/bezEmail.php >>eB.txt
ab -n 500 -c 20 http://localhost/emailMDesifrovani.php >>eMD.txt
```

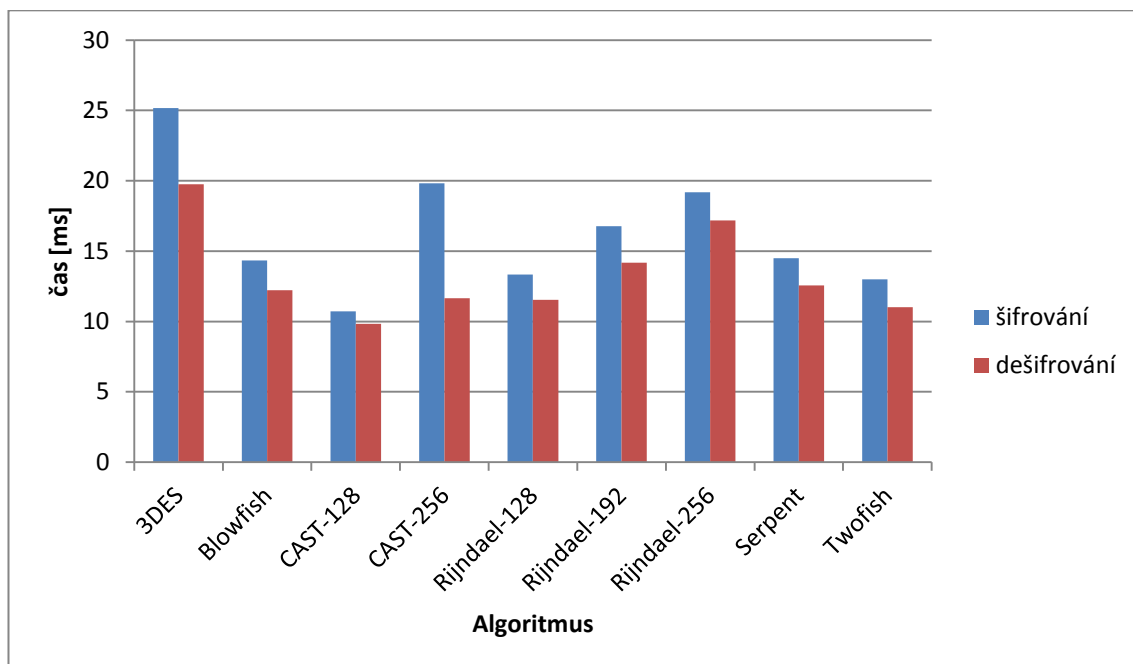
7.6.3.3. Email

V grafu 4 jsou zobrazena data z měření emailu. Při šifrování a zápisu do databáze došlo k nepředpokládanému výsledku, kdy složitější algoritmus byl zpracován rychleji, jak je vidět na grafu 4. Proto byl přidán test šifrování bez zápisu do databáze, který byl proveden až po provedení všech měření dávkovým souborem Prikazy.sh.



Graf 4 - Průměrný čas zpracování skriptu emailů; Zdroj: Autor

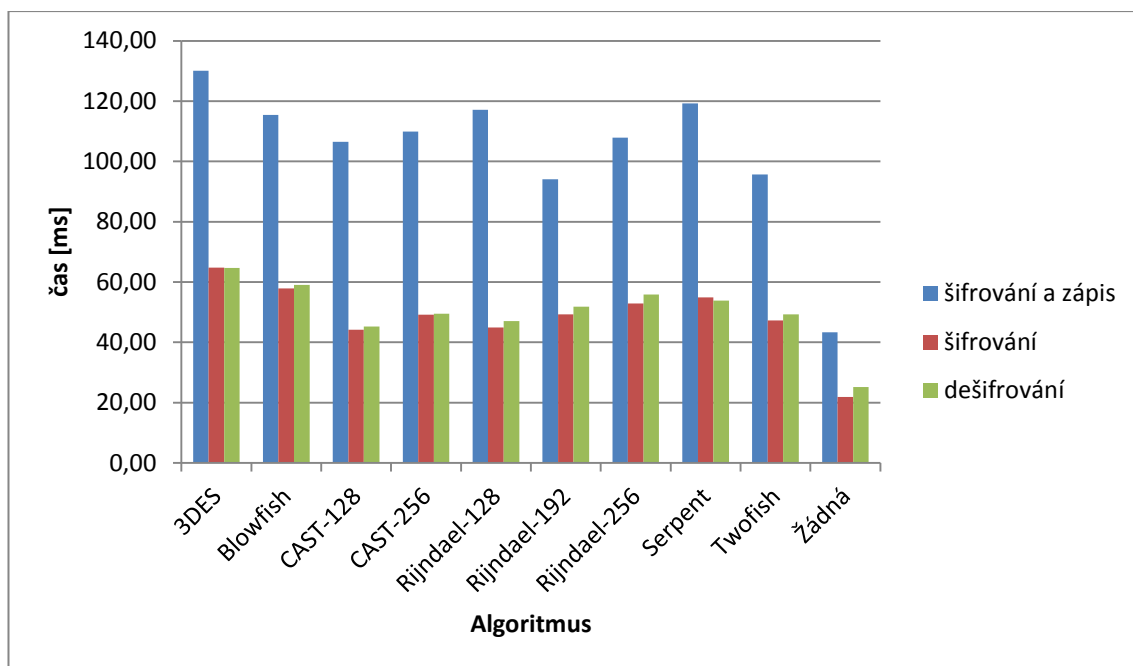
V grafu 5 je odečtena průměrná rychlost skriptu bez šifrovacího algoritmu a odstraněn sloupec s šifrováním a zápisem. V testu z příkazového řádku rychlosti celého skriptu je vidět, že tento test je zhruba o 5-7 ms pomalejší než v případě použití testu z prohlížeče, ale v příkazové řádce také probíhalo paralelně 5 požadavků. Nejrychlejším algoritmem byl opět CAST-128.



Graf 5 - Průměrný čas zpracování skriptu emailů po odečtení doby běhu skriptu bez šifrování; Zdroj: Autor

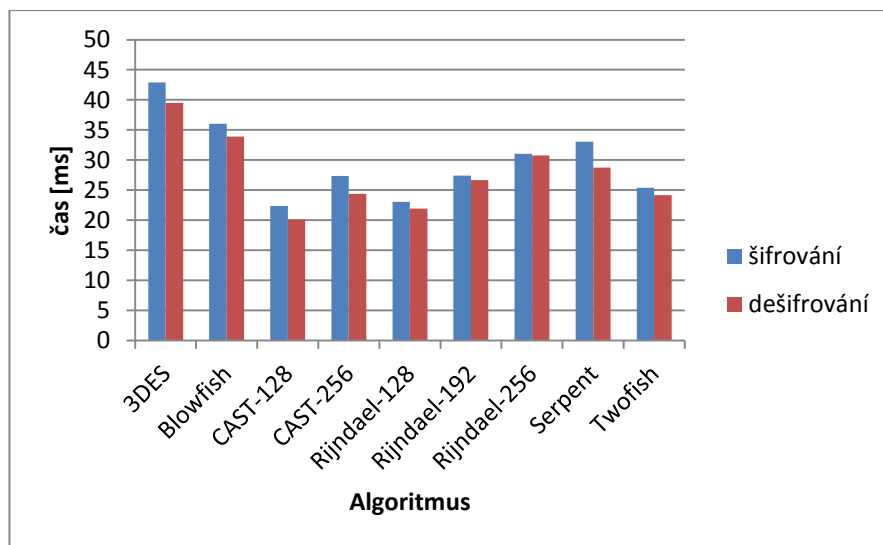
7.6.3.4. Uživatel

V grafu 6 jsou zobrazena data z měření simulace uživatele. I v případě testu s tabulkou uživatelů došlo k vyrovnaní rozdílů mezi algoritmy, bohužel spíše došlo ke zpomalení těch rychlejších než obráceně. I u uživatelů rychlost při šifrování významně kolísala, jak je vidět v grafu 6 a proto byl proveden dodatečný test šifrování bez zápisu do databáze.



Graf 6 - Průměrný čas zpracování skriptu uživatelů; Zdroj: Autor

V grafu 7 je odečtena průměrná rychlost skriptu bez šifrovacího algoritmu a odstraněn sloupec s šifrováním a zápisem. Reakce je snížena až o 10 ms. Na pořadí algoritmů se nic nemění.



Graf 7 - Průměrný čas zpracování skriptu uživatelů po odečtení doby běhu skriptu bez šifrování; Zdroj: Autor

Čas trvání skriptu jednotlivých algoritmů je zobrazeno v tabulce 14 a 15, které jsou doplněny o % výpočet doby skriptu mezi skriptem bez šifrovacího algoritmu a skriptem s algoritmem.

Šifrovací metoda	Šifrování	% navýšení při šifrování	Dešifrování	% navýšení při dešifrování
3DES	47,09	114,83%	44,01	81,41%
Blowfish	36,25	65,36%	36,48	50,38%
CAST-128	32,64	48,92%	34,09	40,54%
CAST-256	41,74	90,43%	35,90	48,01%
Rijndael-128	35,27	60,89%	35,79	47,52%
Rijndael-192	38,68	76,47%	38,43	58,41%
Rijndael-256	41,09	87,47%	41,43	70,80%
Serpent	36,42	66,13%	36,83	51,82%
Twofish	34,92	59,32%	35,27	45,38%
Žádná	21,92	0,00%	24,26	0,00%

Tabulka 14 Doba trvání skriptu u emailu; Zdroj: Autor

Šifrovací metoda	Šifrování	% navýšení při šifrování	Dešifrování	% navýšení při dešifrování
3DES	64,73	196,32%	64,62	157,04%
Blowfish	57,86	164,88%	59,02	134,76%
CAST-128	44,17	102,18%	45,25	80,00%
CAST-256	49,17	125,09%	49,48	96,81%
Rijndael-128	44,89	105,47%	47,05	87,15%
Rijndael-192	49,24	125,40%	51,81	106,06%
Rijndael-256	52,85	141,95%	55,89	122,30%
Serpent	54,91	151,34%	53,87	114,26%
Twofish	47,24	116,23%	49,28	96,02%
Žádná	21,85	0,00%	25,14	0,00%

Tabulka 15 - Doba trvání skriptu u uživatele; Zdroj: Autor

7.6.4. Vyhodnocení testů

Všechny testy vytížily procesor Raspberry Pi na 100 %. Mezi šifrovacími algoritmy jsou rozdíly v šifrování, které bych rozdělil na dvě části: příprava šifry a samotné šifrování.

Velmi rychlou přípravu klíče má šifra Rijndael, která se v testech paralelních požadavků na uživatele nebo email, blížila výsledku nejrychlejšímu algoritmu CAST-128. V zátěžovém testu se zvýšila rychlost proti testům uživatelů a emailů, zejména u algoritmů Blowfish a CAST-128, drobné zlepšení zaznamenaly i šifry Serpent a Twofish.

Navýšení času proti nešifrovaným datům podle typu testu je mezi 10 - 25 ms nebo 20 - 45ms.
Cože je v procentech navýšení o 40 % až 115 % u emailů nebo u uživatelů o 80 % až 200 %.
Viz tabulka 14 a 15.

8. Závěr

Cílem teoretické části bylo seznámit čtenáře s možnostmi šifrování na webových stránkách a v praktické části ověřit vliv šifrovacích algoritmů na rychlost serveru a na velikost databáze.

V teoretické části jsem popsal pojmy XOR a runda, které jsou nezbytná pro pochopení základních principů šifrovacích technik. Dále jsem popsal jednotlivé cíle kryptografie, což je důvěrnost a integrita dat, autentizace entit a dat, nepopíratelnost a také řízení přístupu.

Dále jsem popsal typy šifrování, které se dělí na asymetrické, symetrické a také hashovací funkce, které jsou pouze jednosměrné, proto nelze získat ze zašifrovaného textu zpět původní zprávu. Symetrické šifry se dále dělí na blokové a proudové. Většina šifer je blokových, ale díky různým módům zpracování šifer z nich lze vytvořit proudovou šifru.

U symetrické kryptografie jsem vypsal nejpoužívanější šifry a popsal funkci módů šifer. U asymetrické kryptografie jsem popsal postup předávání klíčů, metodu RSA a eliptické kryptosystémy.

Dvě kapitoly jsem věnoval doporučení pro délku klíčů jednotlivých šifrovacích technik a doporučení pro uživatele, jak správně vytvořit hesla, s tabulkou nejčastěji používaných hesel na internetu. Na závěr teoretické části jsem vypsal nejdůležitější zabezpečení webových aplikací a jak se proti těmto útokům bránit.

Na pomezí teoretické a praktické části jsem se věnoval tématu, v jakých případech a jak šifrovat ve webové aplikaci se zaměřením na uživatelský komfort při ovládání aplikace. Vypsal jsem výhody a nevýhody jednotlivých implementací šifrování v aplikaci.

Na začátku praktické části jsem se věnoval přípravě systému Raspberry Pi, jak systém nastavit a jak nainstalovat webový server. V další části jsem popsal metodiku testu, definoval jsem, jaký mód a šifrovací algoritmy budu testovat, jaký vstup a při jakých modelových situacích budu testovat.

Šifrovací mód jsem vybral CBC a vygeneroval inicializační vektory pro algoritmy: 3DES, Blowfish, CAST-128, CAST-256, Rijndael-128, Rijndael-192, Rijndael-256, Serpent a Twofish. Testovací situace jsem zvolil takové, u kterých si myslím, že je potřeba, aby bylo šifrování použito. Tím je šifrování dat o uživateli, tím myslím hodně buněk s krátkými texty. Já zvolil osmimístný text. Jako druhou situaci jsem zvolil simulaci emailu, tedy první text o délce dvaceti a druhý o délce tisíce znaků. A poslední test byl zátěžový test, kdy jsem chtěl, aby šifrovací

algoritmus pracoval alespoň 30 sekund. Nakonec jsem měřil dobu šifrování textu o délce 5 milionu znaků. Zajímavostí je, že když jsem zkoušel na notebooku s Windows 7 a instalovaným XAMPPem verze 1.8.1 funkčnost aplikace, tak při délce 1 milionu znaků ukončil skript předčasně chybou o nedostatku paměti.

Následovalo měření. Nejdříve jsem začal testováním šifrování a dešifrování v prohlížeči Midori, který je předinstalovaný v Raspbianu. V tomto testu jsem provedl zašifrování/dešifrování a tuto dobu jsem zapsal do databáze. Měřila se přímo rychlost algoritmu. Všechny testy v prohlížeči byly nejrychlejší s šifrovacím algoritmem CAST-128, který bych ovšem nedoporučoval použít z důvodu možnosti vytvoření šifry s krátkým šifrovacím klíčem.

Při druhém testu v testovací utilitě AB, která probíhá v příkazové řádce, se měřila doba celého skriptu, takže koncepce testu zašifrování do databáze a dešifrování z databáze bez následného opětovného zapsání do ní poukázal na problém s významnými výkyvy měření při zápisu šifrovaného textu do databáze. Z výstupu vyšly podezřelé časy šifrování s rozdílem až 60ms, které se mi potvrdily při zpracování dat z kontrolního měření bez jakéhokoli šifrování. V tomto měření bylo 5 měření s průměrnou dobou požadavku kolem 40 ms, jeden s dobou 60 ms a dva s délkou 90 ms. Proto jsem se rozhodl zařadit test nad rámec metodiky testu pro šifrování bez zápisu do databáze, aby se odstranil vliv reakce na databázi, který měření se zápisem zkreslil.

Pro mě překvapivě byl rychlejší test simulace emailů než simulace uživatele. Zdá se, že příprava algoritmu a klíčů trvá delší dobu než samotné šifrování. Nejvíce byla vidět změna mezi testem zátěže a uživatelem, kde se ukázalo, že Rijndael má velmi rychlou přípravu šifry, ale pomalé šifrování. Zato Blowfish má velmi rychlé šifrování, ale pomalou přípravu.

Šifrování je poměrně rychlá záležitost, ale pokud by se jednalo o větší webové aplikace, zvýšení času pro zpracování požadavku o 40 % až 200% podle nasazení u nejrychlejší šifry CAST-128 je významné. Samozřejmě by bylo možno tuto testovací aplikaci nějakým způsobem optimalizovat, ale je otázka, do jaké míry a kolik času by na to bylo potřeba vynaložit.

Dobrou zprávou je, že 5 paralelních požadavků od uživatelů sníží výkon jen o 10%, což je z mého pohledu pro reálné nasazení nízké zvýšení. Je dobře, že se čas neprodloužil násobně.

Vliv na velikost databáze byl rozdílný u každého typu testu. U uživatelů se jednalo o navýšení na 300 % původní velikosti tabulky, u testu emailů na 178 % původní velikosti tabulky a u zátěžového testu na 203 % původní velikosti tabulky.

8.1. Zhodnocení šifrování dat ve webové aplikaci

Šifrování dat není levná záležitost, vezmeme-li v úvahu průměrně dvojnásobné zvýšení kapacity paměti, kterou šifrovaný text zabírá a nutnost vlastnit průměrně o 120 % více serverů. To je významné navýšení pořizovací ceny a ještě vyšší provozní náklady. Šifrování se poskytovatelům služeb nevyplatí z několika důvodů:

- a) poskytovatelé by přišli o cenná data, která mohou zpeněžit, ať už přímo prodejem citlivých dat, nebo jejich analýzou.
- b) Vyšší pořizovací a provozní náklady
- c) Žádný přínos pro poskytovatele, uživatelé by přínos možná ocenili, ale podle mě na tom většině uživatelů nezáleží, vždyť ani nečtou licenční podmínky.

8.2. Přínosy práce

Největším přínosem práce je zjištění, jak šifrovací funkce zatěžují server, konkrétně Raspberry Pi a jaký vliv mají na velikost databáze. Zajímavým výstupem také je, jakým způsobem implementovat šifrování do webové aplikace pro různé typy aplikací.

Závěrem bych chtěl říci, že by bylo dobré zesílit povědomí o této tématice, aby si lidé více chránili své soukromé nebo firemní informace.

9. Seznam obrázků

Obrázek 1 - Schéma šifrování zprávy; Zdroj: (8)	7
Obrázek 2 - ECB schéma zašifrování Zdroj: (62)	13
Obrázek 3 - ECB Schéma dešifrování Zdroj: (61).....	13
Obrázek 4 - CBC schéma zašifrování Zdroj: (60)	14
Obrázek 5 - CBC schéma dešifrování Zdroj: (59).....	14
Obrázek 6 - CFB schéma zašifrování; Zdroj: (63).....	15
Obrázek 7 - CFB schéma dešifrování; Zdroj: (64).....	15
Obrázek 8 - OFB schéma šifrování; Zdroj: (65).....	15
Obrázek 9 - OFB schéma dešifrování; Zdroj: (66)	15
Obrázek 10 - Schéma asymetrického šifrování; Zdroj: (8); Úprava: Autor	16

10. Seznam tabulek

Tabulka 1- XOR; Zdroj: (9)	7
Tabulka 2 - Symetrické šifry; Zdroj: (13)	9
Tabulka 3 - Módy blokových šifer; Zdroj: (37)	16
Tabulka 4 - Srovnatelná bezpečnost různých kryptosystémů při různých délkách klíčů; Zdroj: (39 str. 3)	19
Tabulka 5 - Síla klíče; Zdroj: (43 str. 32)	20
Tabulka 6 - Nejpoužívanější hesla; Zdroj: (45)	21
Tabulka 7 - Výhody a nevýhody typů klíčů; Zdroj: Autor	27
Tabulka 8 - Souhrn výhod a nevýhod podle funkčnosti; Zdroj: Autor	30
Tabulka 9 - Specifikace Raspberry Pi (51); Zdroj: Autor	31
Tabulka 10 - Šifrovací metody v PHP na Raspberry Pi; Zdroj: (57)	33
Tabulka 11 - Vybrané šifrovací metody; Zdroj: Autor	33
Tabulka 12 - Inicializační vektory šifrovacích metod; Zdroj: Autor	35
Tabulka 13 - Velikost tabulek v databázi; Zdroj: Autor	37
Tabulka 14 Doba trvání skriptu u emailu; Zdroj: Autor	44
Tabulka 15 - Doba trvání skriptu u uživatele; Zdroj: Autor	44

11. Seznam grafů

Graf 1 - Rychlost šifrování při zátěži; Zdroj: Autor	39
Graf 2 - Rychlost šifrování emailů; Zdroj: Autor.....	40
Graf 3 - Rychlost šifrování uživatelů; Zdroj: Autor	41
Graf 4 - Průměrný čas zpracování skriptu emailů; Zdroj: Autor	42
Graf 5 - Průměrný čas zpracování skriptu emailů po odečtení doby běhu skriptu bez šifrování; Zdroj: Autor	42
Graf 6 - Průměrný čas zpracování skriptu uživatelů; Zdroj: Autor	43
Graf 7 - Průměrný čas zpracování skriptu uživatelů po odečtení doby běhu skriptu bez šifrování; Zdroj: Autor	43

12. Bibliografie

1. Česká republika. Listina základních práv a svobod. In: *Ústava České republiky* [Online] [Citace: 21. 4 2014.] Dostupné z: <http://business.center.cz/business/pravo/zakony/listina-zakladnich-prav-a-svobod/hlava2.aspx>.
2. MOOS, Jiří. CDR Security Update - Google aktualizoval podmínky a veřejně přiznal šmírování | CDR.cz. *CDR.cz - Vybráno z IT*. [Online] 22. 4 2014. [Citace: 3. 5 2014.] Dostupné z: <http://cdr.cz/clanek/cdr-security-update-google-aktualizoval-podminky-verejne-priznal-smirovani>. ISSN 1213-2225.
3. Smluvní podmínky společnosti Google – Zásady a pravidla – Google. *Google*. [Online] 30. 4 2014. [Citace: 3. 5 2014.] Dostupné z: <https://www.google.cz/intl/cs/policies/terms/regional.html>.
4. ELMINAAM, D. S. Abdul., KADER, H. M. Abdul a HADHOUD, M. M. Communications of the IBIMA IBIMA Publishing. [Online] roč. 2009, č. 8 [Citace: 11. 5 2014.] ISSN 1943-7765. Dostupné z: <http://www.ibimapublishing.com/journals/CIBIMA/volume8/v8n8.pdf>.
5. COFFEY, Neil. Comparison of encryption ciphers in Java. *Javamex: Java tutorials and performance information*. [Online] [Citace: 11. 5 2014.] Dostupné z: <http://www.javamex.com/tutorials/cryptography/ciphers.shtml>.
6. TOTZAUER, Tomáš. Závěrečné práce. *ISIS*. [Online] 5 2012. [Citace: 12. 5 2014.] Dostupné z: https://isis.vse.cz/zp/portal_zp.pl?prehled=vyhledavani;podrobnosti=103674;download_prace=1.
7. GANIAN, Robert. Microsoft Word - Dokument1 - Testovani_rychlosti_proudovych_sifer.pdf. *Veřejné služby Informačního systému*. [Online] 2006. [Citace: 10. 5 2014.] Dostupné z: http://is.muni.cz/th/99352/fi_b/Testovani_rychlosti_proudovych_sifer.pdf.
8. HENZLER, Jiří a PELIKÁN, Jan. *Matematické Základy informatiky*. Praha: Oeconomica, 2009. ISBN 978-80-245-1537-3.
9. Hradlo XOR – Wikipedie. *Wikipedie, otevřená encyklopedie*. [Online] 14. 3 2013. [Citace: 11. 5 2014.] Dostupné z: http://cs.wikipedia.org/wiki/Hradlo_XOR.
10. PALOVSKÝ, Radomír. *Bezpečnost informací v internetu: Přednáška 9*. [Online] 21. 5 2012. [Citace: 28. 4 2014.] Dostupné z: https://isis.vse.cz/auth/dok_server/slozka.pl?id=110576;download=82047. Veřejně nedostupné
11. KLÍMA, Vlastimil. Symetrická kryptografie I. - Symetricka_kryptografie_I_2004.pdf. *Crypto-World*. [Online] [Citace: 6. 5 2014.] Dostupné z: http://crypto-world.info/klima/mffuk/Symetricka_kryptografie_I_2004.pdf.
12. KLÍMA, Vlastimil. Symetrická kryptografie II. - Symetricka_kryptografie_II_2004.pdf. *Crypto-World*. [Online] [Citace: 5. 5 2014.] Dostupné z: http://crypto-world.info/klima/mffuk/Symetricka_kryptografie_II_2004.pdf.
13. Symmetric-key algorithm - Wikipedia, the free encyclopedia. *Wikipedia, the free encyclopedia*. [Online] 25. 2 2014. [Citace: 5. 5 2014.] Dostupné z: http://en.wikipedia.org/wiki/Symmetric-key_algorithm.

14. Known-plaintext attack - Wikipedia, the free encyclopedia. *Wikipedia, the free encyclopedia*. [Online] 18. 1 2014. [Citace: 11. 5 2014.] Dostupné z: http://en.wikipedia.org/wiki/Known-plaintext_attack.
15. Chosen-plaintext attack - Wikipedia, the free encyclopedia. *Wikipedia, the free encyclopedia*. [Online] 1. 3 2014. [Citace: 11. 5 2014.] Dostupné z: http://en.wikipedia.org/wiki/Chosen-plaintext_attack.
16. Differential cryptanalysis - Wikipedia, the free encyclopedia. *Wikipedia, the free encyclopedia*. [Online] 5. 2 2014. [Citace: 11. 5 2014.] Dostupné z: http://en.wikipedia.org/wiki/Differential_cryptanalysis.
17. Linear cryptanalysis - Wikipedia, the free encyclopedia. *Wikipedia, the free encyclopedia*. [Online] 12. 3 2014. [Citace: 11. 5 2014.] Dostupné z: http://en.wikipedia.org/wiki/Linear_cryptanalysis.
18. Boomerang attack - Wikipedia, the free encyclopedia. *Wikipedia, the free encyclopedia*. [Online] 21. 12 2013. [Citace: 11. 5 2014.] Dostupné z: http://en.wikipedia.org/wiki/Boomerang_attack.
19. WOOD, Lamont. Nastal čas na změnu principů šifrování? (1) | Computerworld.cz. *Computerworld.cz*. [Online] 30. 4 2012. [Citace: 11. 5 2014.] Dostupné z: <http://computerworld.cz/securityworld/nastal-cas-na-zmenu-principu-sifrovani-1-48441>.
20. VONDRUŠKA, Pavel. crypto10_00.pdf. *Crypto-World*. [Online] 15. 10 2000, roč. 2, č. 10/2000 [Citace: 3. 5 2014.] Dostupné z: http://crypto-world.info/casop2/crypto10_00.pdf. ISSN 1801-2140.
21. Data Encryption Standard - Wikipedia, the free encyclopedia. *Wikipedia, the free encyclopedia*. [Online] 5. 4 2014. [Citace: 4. 5 2014.] Dostupné z: http://en.wikipedia.org/wiki/Data_Encryption_Standard.
22. RSA Laboratories - 2.4.4 What is the RSA Secret Key Challenge? *EMC - Leading Cloud Computing, Big Data, and Trusted IT Solutions*. [Online] [Citace: 4. 5 2014.] Dostupné z: <http://www.emc.com/emc-plus/rsa-labs/standards-initiatives/what-is-the-rsa-secret-key-challenge.htm>.
23. FIPS 46-3, Data Encryption Standard (DES) (withdrawn May 19, 2005) - fips46-3.pdf. *NIST Computer Security Division - Computer Security Resource Center*. [Online] 25. 10 1999. [Citace: 4. 5 2014.] Dostupné z: <http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf>.
24. VONDRUŠKA, Pavel. Crypto-World - crypto09_99.pdf. *Crypto-World*. [Online] 7. 9 1999, roč. 1, č. 9/1999 [Citace: 3. 5 2014.] Dostupné z: http://crypto-world.info/casop1/crypto09_99.pdf. ISSN 1801-2140.
25. DOLEŽAL, Petr. AES - šifra pro třetí tisíciletí. *AES - šifra pro třetí tisíciletí*. [Online] 6. 7 2001. [Citace: 3. 5 2014.] Dostupné z: <http://www.jikos.cz/~gumysh/docs/AES/Pages/Page02.html>.
26. DOLEŽAL, Petr. AES - šifra pro třetí tisíciletí. *AES - šifra pro třetí tisíciletí*. [Online] 6. 7 2001. [Citace: 3. 5 2014.] Dostupné z: <http://www.jikos.cz/~gumysh/docs/AES/Pages/Page08.html>.
27. DOLEŽAL, Petr. AES - šifra pro třetí tisíciletí. *AES - šifra pro třetí tisíciletí*. [Online] 6. 7 2001. [Citace: 3. 5 2014.] Dostupné z: <http://www.jikos.cz/~gumysh/docs/AES/Pages/Page03.html>.

28. Advanced Encryption Standard – Wikipedie. *Wikipedie, otevřená encyklopedie*. [Online] 24. 4 2014. [Citace: 11. 5 2014.] Dostupné z: http://cs.wikipedia.org/wiki/Advanced_Encryption_Standard.
29. SCHNEIER, Bruce. Schneier on Security: Blowfish. *Schneier on Security*. [Online] [Citace: 11. 5 2014.] Dostupné z: <https://www.schneier.com/blowfish.html>.
30. Blowfish (cipher) - Wikipedia, the free encyclopedia. *Wikipedia, the free encyclopedia*. [Online] 3. 5 2014. [Citace: 11. 5 2014.] Dostupné z: [http://en.wikipedia.org/wiki/Blowfish_\(cipher\)](http://en.wikipedia.org/wiki/Blowfish_(cipher)).
31. CAST-128 - Wikipedia, the free encyclopedia. *Wikipedia, the free encyclopedia*. [Online] 26. 2 2014. [Citace: 11. 5 2014.] Dostupné z: <http://en.wikipedia.org/wiki/CAST-128>.
32. CAST-256 - Wikipedia, the free encyclopedia. *Wikipedia, the free encyclopedia*. [Online] 23. 11 2013. [Citace: 11. 5 2014.] Dostupné z: <http://en.wikipedia.org/wiki/CAST-256>.
33. Serpent home page. *The Computer Laboratory*. [Online] [Citace: 11. 5 2014.] Dostupné z: <http://www.cl.cam.ac.uk/~rja14/serpent.html>.
34. Serpent (cipher) - Wikipedia, the free encyclopedia. *Wikipedia, the free encyclopedia*. [Online] 26. 4 2014. [Citace: 11. 5 2014.] Dostupné z: [http://en.wikipedia.org/wiki/Serpent_\(cipher\)](http://en.wikipedia.org/wiki/Serpent_(cipher)).
35. SCHNEIER, Bruce. Schneier on Security: Twofish. *Schneier on Security*. [Online] [Citace: 11. 5 2014.] Dostupné z: <https://www.schneier.com/twofish.html>.
36. Twofish - Wikipedia, the free encyclopedia. *Wikipedia, the free encyclopedia*. [Online] 30. 1 2014. [Citace: 11. 5 2014.] Twofish - Wikipedia, the free encyclopedia.
37. Block cipher mode of operation - Wikipedia, the free encyclopedia. *Wikipedia, the free encyclopedia*. [Online] 2. 5 2014. [Citace: 4. 5 2014.] Dostupné z: http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation.
38. KLÍMA, Vlastimil. Symetrická kryptografie III. - Symetrická_kryptografie_III_2004.pdf. *Crypto-World*. [Online] [Citace: 11. 5 2014.] Dostupné z: http://crypto-world.info/klima/mffuk/Symetricka_kryptografie_III_2004.pdf.
39. PINKAVA, Jaroslav. B - crypto11_99.pdf. *Crypto-World*. [Online] 1. 11 1999, roč. 1, č. 11/1999 [Citace: 3. 5 2014.] Dostupné z: http://crypto-world.info/casop1/crypto11_99.pdf. ISSN 1801-2140.
40. KÜMMEL, Roman. crypto0910_13.pdf. *Crypto-World*. [Online] 28. 10 2013, roč. 15, č. 9-10/2013 [Citace: 3. 5 2014.] Dostupné z: http://crypto-world.info/casop15/crypto0910_13.pdf. ISSN 1801-2140.
41. VRÁNA, Jakub. Crypto-World - crypto1112_13.pdf. *Crypto-World*. [Online] 15. 12 2013, roč. 15, č. 11-12/2013 [Citace: 3. 5 2014.] Dostupné z: http://crypto-world.info/casop15/crypto1112_13.pdf. ISSN 1801-2140.
42. HOWARD, Michael a LEBLANC, David. Bezpečný kód. Brno: 2008, Computer Press. ISBN 978-80-251-2050-7.
43. D.SPA.17.pdf. *ECRYPT NoE*. [Online] 30. 6 2011. [Citace: 4. 5 2014.] Dostupné z: <http://www.ecrypt.eu.org/documents/D.SPA.17.pdf>. ICT-2007-216676.

44. GOODIN, Dan. 25-GPU cluster cracks every standard Windows password in <6 hours | Ars Technica. *Ars Technica*. [Online] 10. 12 2012. [Citace: 4. 5 2014.] Dostupné z: <http://arstechnica.com/security/2012/12/25-gpu-cluster-cracks-every-standard-windows-password-in-6-hours/>.
45. SKÝPALA, Martin. Žebříček nejpoužívanějších hesel 2013. *Eset*. [Online] 22. 1 2014. [Citace: 6. 5 2014.] Dostupné z: <http://www.eset.com/cz/o-nas/blog/article/zebricek-nejpouzivanejsich-hesel-2013/>.
46. VONDRUŠKA, Pavel. crypto03_06 - crypto03_06.pdf. *Crypto-World*. [Online] 15. 3 2006, roč. 8, č. 3/2006 [Citace: 9. 5 2014.] Dostupné z: http://crypto-world.info/casop8/crypto03_06.pdf.
47. HTTPS – Wikipedie. *Wikipedie, otevřená encyklopedie*. [Online] 19. 3 2014. [Citace 11. 5 2014.] Dostupné z: <http://cs.wikipedia.org/wiki/Https>.
48. VRÁNA, Jakub. PHP triky - Obrana proti SQL Injection. *PHP triky - Weblog o elegantním programování v PHP pro mírně pokročilé*. [Online] 2. 3 2005. [Citace: 9. 5 2014.] Dostupné z: <http://php.vrana.cz/obrana-proti-sql-injection.php>.
49. VRÁNA, Jakub. PHP triky - Cross Site Scripting. *PHP triky - Weblog o elegantním programování v PHP pro mírně pokročilé*. [Online] 23. 11 2005. [Citace: 9. 5 2014.] Dostupné z: <http://php.vrana.cz/cross-site-scripting.php>.
50. VRÁNA, Jakub. PHP triky - Cross-Site Request Forgery. *PHP triky - Weblog o elegantním programování v PHP pro mírně pokročilé*. [Online] 24. 4 2006. [Citace: 9. 5 2014.] Dostupné z: <http://php.vrana.cz/cross-site-request-forgery.php>.
51. Raspberry Pi - model B eshop.raspishop.cz. *Eshop - prodej Raspberry Pi a doplňky k Raspberry Pi*. [Online] [Citace: 19. 4 2014.] Dostupné z: <http://eshop.raspishop.cz/raspberry-pi-model-b-p-1.html>.
52. Kingston SDHC 16GB UHS-I Class 10 Ultimate | Alza.cz. *Alza*. [Online] [Citace: 8. 5 2014.] Dostupné z: <http://www.alza.cz/kingston-sdhc-16gb-uhs-i-class-10-ultimate-d466743.htm>.
53. Downloads | Raspberry Pi. *Raspberry Pi*. [Online] 7. 1 2014. [Citace: 24. 3 2014.] Dostupné z: <http://www.raspberrypi.org/downloads>.
54. Win32 Disk Imager - Browse /Archive at SourceForge.net. *SourceForge - Download, Develop and Publish Free Open Source Software*. [Online] 7. 8 2013. [Citace: 24. 3 2014.] Dostupné z: <http://sourceforge.net/projects/win32diskimager/files/Archive/>.
55. UPTON, Eben a HALFACREE, Gareth. *Raspberry Pi Uživatelská příručka*. Brno : Computer Press, 2013. 978-80-251-4116-8.
56. ab - Apache HTTP server benchmarking tool - Apache HTTP Server. *Apache Software Foundation*. [Online] [Citace: 24. 3 2014.] Dostupné z: <http://httpd.apache.org/docs/2.2/programs/ab.html>.
57. PHP: mcrypt_list_algorithms - Manual. *PHP: Hypertext Preprocessor*. [Online] [Citace: 8. 5 2014.] Dostupné z: <http://cz2.php.net/manual/en/function.mcrypt-list-algorithms.php>.

58. BESANEK. [PHP] Šifrování - 4WebMaster.cz. *4WebMaster.cz*. [Online] 17. 10 2011. [Citace: 10. 5 2014.] Dostupné z: <https://web.archive.org/web/20120313042959/http://4webmaster.cz/php-mysql/sifrovani-t374.html>.
59. WHITETIMBERWOLF. File:CBC decryption.svg - Wikipedia, the free encyclopedia. *Wikipedia, the free encyclopedia*. [Online] 1. 6 2013. [Citace: 28. 4 2014.] Dostupné z: http://en.wikipedia.org/wiki/File:CBC_decryption.svg.
60. WHITETIMBERWOLF. File:CBC encryption.svg - Wikipedia, the free encyclopedia. *Wikipedia, the free encyclopedia*. [Online] 1. 6 2013. [Citace: 28. 4 2014.] Dostupné z: http://en.wikipedia.org/wiki/File:CBC_encryption.svg.
61. WHITETIMBERWOLF. File:ECB decryption.svg - Wikipedia, the free encyclopedia. *Wikipedia, the free encyclopedia*. [Online] 1. 6 2013. [Citace: 28. 4 2014.] Dostupné z: http://en.wikipedia.org/wiki/File:ECB_decryption.svg.
62. WHITETIMBERWOLF. File:ECB encryption.svg - Wikipedia, the free encyclopedia. *Wikipedia, the free encyclopedia*. [Online] 1. 6 2013. [Citace: 28. 4 2014.] Dostupné z: http://en.wikipedia.org/wiki/File:ECB_encryption.svg.
63. WHITETIMBERWOLF. File:CFB encryption.svg - Wikipedia, the free encyclopedia. *Wikipedia, the free encyclopedia*. [Online] 1. 6 2013. [Citace: 28. 4 2014.] Dostupné z: http://en.wikipedia.org/wiki/File:CFB_encryption.svg.
64. WHITETIMBERWOLF. File:CFB decryption.svg - Wikipedia, the free encyclopedia. *Wikipedia, the free encyclopedia*. [Online] 1. 6 2013. [Citace: 28. 4 2014.] Dostupné z: http://en.wikipedia.org/wiki/File:CFB_decryption.svg.
65. WHITETIMBERWOLF. File:OFB encryption.svg - Wikipedia, the free encyclopedia. *Wikipedia, the free encyclopedia*. [Online] 1. 6 2013. [Citace: 28. 4 2014.] Dostupné z: http://en.wikipedia.org/wiki/File:OFB_encryption.svg.
66. WHITETIMBERWOLF. File:OFB decryption.svg - Wikipedia, the free encyclopedia. *Wikipedia, the free encyclopedia*. [Online] 1. 6 2013. [Citace: 28. 4 2014.] Dostupné z: http://en.wikipedia.org/wiki/File:OFB_decryption.svg.

13. Příloha 1 - Verze balíčků PHP, MySQL a Apache

```
php5
Versions:
5.4.4-14+deb7u9
Dependencies:
5.4.4-14+deb7u9 - libapache2-mod-php5 (18 5.4.4-14+deb7u9)
libapache2-mod-php5filter (18 5.4.4-14+deb7u9) php5-cgi (18
5.4.4-14+deb7u9) php5-fpm (2 5.4.4-14+deb7u9) php5-common (2
5.4.4-14+deb7u9)

Package: php5-mysql
Versions:
5.4.4-14+deb7u9
Dependencies:
5.4.4-14+deb7u9 - libc6 (2 2.13-28) libmysqlclient18 (2
5.5.24+dfsg-1) zlib1g (2 1:1.1.4) phpapi-20100525+lfs (0 (null))
php5-common (5 5.4.4-14+deb7u9) ucf (0 (null)) dpkg (2
1.15.7.2~) php5-mysqli (0 (null)) php5-mysqldb (0 (null)) php5-
mysqli (0 (null)) php5-mysqldb (0 (null))

Package: apache2
Versions:
2.2.22-13+deb7u1
Dependencies:
2.2.22-13+deb7u1 - apache2-mpm-worker (21 2.2.22-13+deb7u1)
apache2-mpm-prefork (21 2.2.22-13+deb7u1) apache2-mpm-event (21
2.2.22-13+deb7u1) apache2-mpm-itk (5 2.2.22-13+deb7u1)
apache2.2-common (5 2.2.22-13+deb7u1)
Provides:
2.2.22-13+deb7u1 -
Reverse Provides:
apache2-mpm-worker 2.2.22-13+deb7u1
apache2-mpm-prefork 2.2.22-13+deb7u1
apache2-mpm-itk 2.2.22-13+deb7u1
apache2-mpm-event 2.2.22-13+deb7u1

Package: mysql-server
Versions:
5.5.37-0+wheezy1
Dependencies:
```

```
5.5.37-0+wheezy1 - mysql-server-5.5 (0 (null))
```

```
Provides:
```

```
5.5.37-0+wheezy1 -
```

```
Reverse Provides:
```

```
mysql-server-5.1 5.1.62-1
```

14. Příloha 2 - SQL kódy

SQL kód struktury databáze a SQL kód struktury databáze včetně vygenerovaných dat naleznete na přiloženém CD ve složce SQL.

15. Příloha 3 - Zdrojový kód

Zdrojový kód všech souborů aplikace naleznete na přiloženém CD ve složce PHP.

16. Příloha 4 - Konfigurace PHP

Výpis konfigurace PHP je umístěn na přiloženém CD ve složce Systém (soubor php.html).