VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

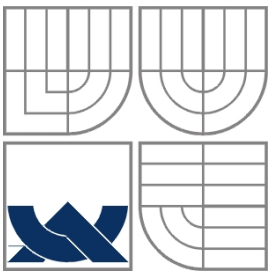
BEZPEČNÉ KRYPTOGRAFICKÉ ALGORITMY

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

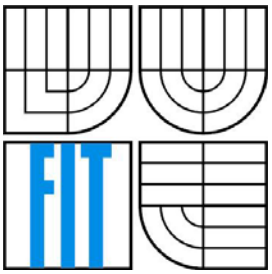
AUTOR PRÁCE
AUTHOR

Bc. JAKUB MAHDAL

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

BEZPEČNÉ KRYPTOGRAFICKÉ ALGORITMY

SAFE CRYPTOGRAPHY ALGORITHMS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JAKUB MAHDAL

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PETR CHMELAŘ

BRNO 2008

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2007/2008

Zadání diplomové práce

Řešitel: **Mahdal Jakub, Bc.**
Obor: Informační systémy
Téma: **Bezpečné kryptografické algoritmy**
Kategorie: Bezpečnost

Pokyny:

1. Seznamte se s problematikou kryptografických algoritmů a jejich bezpečnosti.
2. Zaměřte se na současně používané algoritmy pro symetrickou, asymetrickou kryptografii, tvorbu hashů a generování náhodných čísel. Porovnejte je s předchozími verzemi z pohledu slabých míst a nalezených zranitelností.
3. Vytvořte přehled algoritmů, které dosud nebyly prolomeny a je možné je považovat za bezpečné a těch, které nesplňují tyto nároky.
4. Proveďte zhodnocení vlastností bezpečných a případné vylepšení nevhodných algoritmů.

Literatura:

- Cílem je najít relevantní literaturu a zhodnotit její obsah z:
  www.crypto-world.info
  www.scmagazine.com
  www.gartner.com
  scholar.google.com
- Konzultace s externím odborníkem.

Při obhajobě semestrální části diplomového projektu je požadováno:

- Body 1 až 2.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVR-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Chmelař Petr, Ing.**, UIFS FIT VUT
Konzultant: Drozd Michal, Ing., FIT VUT
Datum zadání: 24. září 2007
Datum odevzdání: 19. května 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Ing. Jaroslav Zendulka, CSc.
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. Pan

Jméno a příjmení: **Bc. Jakub Mahdal**
Id studenta: 89738
Bytem: Mosty u Jablunkova 512, 739 98 Mosty u Jablunkova
Narozen: 24. 07. 1984, Třinec
(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
diplomová práce

Název VŠKP: Bezpečné kryptografické algoritmy
Vedoucí/školitel VŠKP: Chmelař Petr, Ing.
Ústav: Ústav informačních systémů
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:
tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☐ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

....

Nabyvatel Autor

Abstrakt

Práce podává přehled historického i moderního světa kryptografických metod a postupů, zhodnocuje aktuální stav vývoje kryptografie a kryptografických algoritmů, které jsou používány v dnešních aplikacích. Popisuje a vysvětluje aktuálně nejčastěji používané symetrické a asymetrické šifrovací algoritmy, hašovací funkce, funkce pro generování pseudonáhodných čísel, autentizační protokoly a protokoly pro tvorbu VPN. Práce dále popisuje základní úspěšné metody kryptoanalýzy a ukazuje, které algoritmy jsou z hlediska dostupných prostředků zranitelné, a které jsou náchylné k útokům. Dokument dále podává přehled a doporučení k jednotlivým metodám, které doposud útoky vydržely a u kterých se dá i nadále předpokládat bezpečné využití i do budoucna.

Klíčová slova

Šifrování, symetrické šifry, asymetrické šifry, kryptoanalýza, kryptografie, bezpečnost, hashovací funkce, generátor pseudonáhodných čísel, PRNG, VPN, SSL, TLS, protokoly, autentizace

Abstract

This thesis brings a reader an overview about historical and modern world of cryptographic methods, as well evaluates actual state of cryptographic algorithm progressions, which are used in applications nowadays. The aim of the work describes common symmetric, asymmetric encryption methods, cryptographic hash functions and as well pseudorandom number generators, authentication protocols and protocols for building VPNs. This document also shows the basics of the successful modern cryptanalysis and reveals algorithms that shouldn’t be used and which algorithms are vulnerable. The reader will be also recommended an overview of cryptographic algorithms that are expected to stay safe in the future.

Keywords

Encryption, symmetric encryption, asymmetric encryption, cryptanalysis, cryptography, security, hash function, random number generator, PRNG, VPN, SSL, TLS, protocols, authentication

Citace

Mahdal Jakub: Bezpečné kryptografické algoritmy. Brno, 2008, diplomová práce, FIT VUT v Brně.

Bezpečné kryptografické algoritmy

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Petra Chmelaře. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jakub Mahdal
28.4.2008

Poděkování

Rád bych poděkoval vedoucímu práce Ing. Petru Chmelařovi a Ing. Michalu Drozdovi za cenné rady a zkušenosti, které mi po dobu práce ochotně předávali a své přítelkyni za duševní podporu během psaní této práce.

Obsah

1 Úvod

Je již dlouho známo, že bezpečnost není stav, ale trvalý proces. Proces, který obsahuje celou řadu požadavků a cílů, kterých chce lidstvo ve většině odvětvích dosáhnout, a ke kterým se snaží přiblížit. V široké oblasti informačních technologií mezi ty základní a nejdůležitější aspekty bezpečnosti řadíme: důvěrnost, integrita, autenticita a nepopíratelnost. V této oblasti pojem bezpečnosti ve smyslu výše uvedených cílů prostupuje téměř veškerým působením a nelze jej snadno vymezit. Co je však zřejmé, že mezi odvětví, která se naplněním aspektů bezpečnosti zabývají, neodmyslitelně patří i oblast kryptografie.

Moderní éra kryptografie prodělala velmi rychlý vývoj a tento vývoj je patrný i v posunu paradigmat během posledních historických etap jejího vývoje.

Prostředky v kryptografii k podpoře zajištění bezpečnostních cílů se nazývají kryptografické algoritmy. Nejsou však strůjcem dokonalého bezpečí. Mezi návrhem, implementací a praktickym použitím je ještě dlouhá cesta. Nicméně je to podmínka nutná, nikoli dostačující, a tedy pokud není kvalitní návrh, je zbytečné na něm dále stavět, protože systém je tak silný, jak je silná jeho nejslabší část. A není od věci zmínit, že téměř ve všech systémech je z bezpečnostního hlediska nejslabší částí lidský faktor.

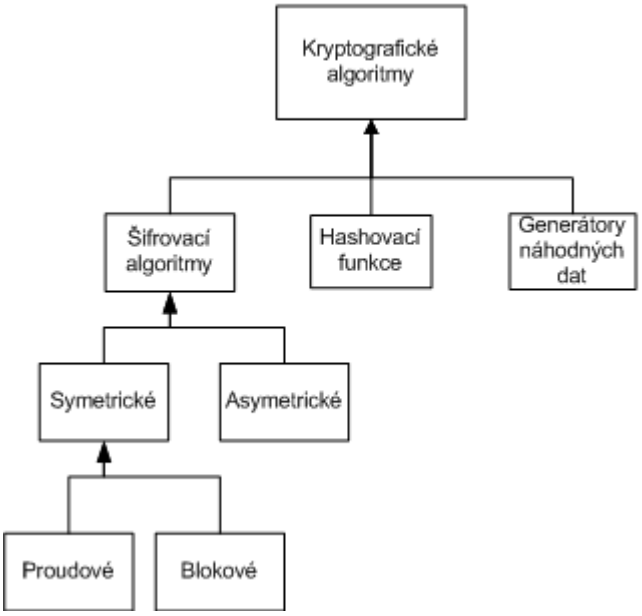
Útoky na kryptografické algoritmy se podle slov předního světového kryptoanalytika Bruce Schneiera [1] časem pouze zrychlují, nikdy ne naopak. Pokusím se tedy v rámci této práce vytvořit komplexní a aktuální přehled doposud dosaženého v této oblasti s vlastním zhodnocením a návrhy pro další směry vývoje.

1.1 Co je to kryptografie? Základní pojmy

Kryptografie [2] (z řeckého *kryptos* – skrytý a *gráfo* psát – skryté psaní) je pro většinu lidí termín zcela ztotožňován s pojmem šifrování. Šifrování jako transformace, která převádí zdrojová data do nečitelné podoby s využitím utajované informace. Je pravda, že původní vymezení pojmu kryptografie dřívější potřeby pokrývalo. Ovšem toto vymezení stačilo až do objevení nových přístupů moderní kryptografie, kdy se pro zajištění bezpečnostních cílů objevují i další oblasti, které původní definici nezbytně rozšiřují. Moderní kryptografie pracuje s termíny šifrování, hashovací (jednocestné) funkce, symetrické a asymetrické šifrování, které tvoří novodobý kryptografický základ, jak si ukážeme dále pro zajištění dalších bezpečnostních cílů. Jsou to právě ty algoritmy, na které bude v této práci kladen důraz, a které se budu snažit podrobněji analyzovat.

Kryptoanalýza je věda zabývající se analýzou matematických mechanismů ochrany dat [3] – tedy prolomením kryptografických metod. Kryptoanalytik, tj. člověk provádějící kryptoanalýzu, se snaží odhalovat slabiny kryptografických algoritmů – např. pro oblast šifrování je tedy příjemce

šifrované zprávy, který za pomocí nových metod, bez znalostí klíče a se znalostí informací z návrhu šifry odhaluje její slabiny. Pro hashovací funkce provádí kryptoanalytik útok na některé ze základních vlastností těchto funkcí - např. bezkoliznost (– tj. snaží se najít tzv. kolize – stejné výstupy pro jiný vstupní text (více v kapitole 4.4).



Obrázek 1-1 Schéma moderní kryptografie

Rád bych osvětlil také pojem kódování, který je s pojmem šifrování často mylně zaměňován. Stejně jako šifrování je kódování proces transformace dat, nicméně kódování není přímo spojeno s cílem informaci utajit – ale v nahrazení původní informace informací jinou. Hlavní rozlišující kritérium mezi kódy a šiframi je fakt, že k převodu kódu do čitelné podoby je nutná pouze znalost mechanismu kódování [4], k převodu šifry ale nikoli. K převodu šifry do čitelné podoby je ještě nezbytná nutnost znalosti další informace – šifrovacího klíče, tj. informace, která vstupuje jako další dodatečná znalost. Vzpomněl bych pro ilustraci nejznámější kódy Morseova abeceda, z počítačové oblasti pak kódy Unicode, ASCII.

1.1.1 Symetrické a asymetrické šifrovací algoritmy

Pojem šifrování bychom mohli rozdělit strukturovaně do několika kategorií. Obecný princip šifrování je uveden na obrázku Obrázek 1-2.

Obrázek 1-2 Šifrování a dešifrování obecně

Na obrázku Obrázek 1-2 lze demonstrovat rozdíl mezi symetrickou a asymetrickou kryptografií. Symetrická kryptografie [4] používá k šifrování i dešifrování stejný klíč, zatímco asymetrická nikoli. Hlavním problémem v komunikaci symetrických spojení je zajištění bezpečného přenosu klíče všem stranám komunikace.

Symetrické algoritmy se dále dělí podle typu šifry na blokové a proudové (viz obrázek 1-1). Proudové šifrovací algoritmy šifrují vstupní zprávu po bitech, zatímco blokové šifry rozdělí vstupní zprávu na bloky pevné délky a šifrují je (více v podkap. 4.2).

Asymetrická kryptografie používá k šifrování a dešifrování rozdílné klíče, které jsou v určitém matematickém vztahu. Tyto klíče se nazývají klíčový pár a nejčastěji jsou koncipovány tak, že jeden je uschován u majitele klíčového páru a druhý je volně distribuován na veřejnost – hovoříme o klíčích – soukromý (dále také jako privátní) a veřejný. Princip komunikace pak funguje tak, že se zpráva šifruje buďto veřejným nebo soukromým klíčem a příjemce jej dešifruje klíčem druhým, ovšem ze stejného klíčového páru. Postup funguje oběma směry, má však jiné praktické vlasnosti. Pokud tedy chce odesílatel zprávu utajit a zaslat příjemci, zašifruje veřejným klíčem patřícím příjemci. Vzhledem k tomu, že druhým klíčem je klíč soukromý, a nikdo jiný, než příjemce jeho znalostí nedisponuje. Rozšifrování zprávy je čistě v moci příjemce se správným soukromým klíčem (více v podkap. 4.3). Problémem v asymetrickém způsobu šifrování je samotná existence klíčových párů, nebo spíš efektivní distribuce veřejných klíčů druhým stranám a v případě ztráty/krádeže privátního klíče nebo jeho expirace, pak efektivní *revokace*, což je veřejné zneplatnění klíčového páru a zajištění, aby měl odesílatel v co nejkratším čase informaci, že klíčový pár příjemce nepoužívá (viz také podkap. 7.2.1). Obecným problémem asymetrické kryptografie je také rychlost celého šifrovacího procesu, která je neúměrně pomalejší, než je tomu v případě symetrických algoritmů [4]. Zatímco u symetrické kryptografie opakovaně využíváme základních logických a aritmetických operací, asymetrická kryptografie stojí na výpočtu mocnin, eliptických křivek nebo diskrétního logaritmu, což výpočet velmi zpomaluje (viz podkap. 4.3).

Další zajímavou vlastností asymetrických šifrovacích algoritmů je fakt, že návrh asymetrického algoritmu je v porovnání se symetrickými velmi složitý – prakticky se jedná o objev

nového matematického vztahu. Tím lze vysvětlit také omezený počet algoritmů, které se pro asymetrickou kryptografii úspěšně používají [4].

1.1.2 Hashovací funkce

Hashovací funkce, také nazývána jednosměrná či jednocestná funkce, produkuje z obecně dlouhého konečného vstupu unikátní deterministický výstup [5]. Hashovací funkce tedy získává z libovolně dlouhého konečného datového vstupu výstup konstantní délky (prakticky ve stovkách bitů), který je v praxi nejčastěji nazýván tzv. *hash* nebo také *otisk zprávy*.
Cíle ideální hashovací funkce [4]:

1. **Jednocestnost** (často také **jednosměrnost**)
   Zajištění, že pro všechny výstupy je výpočetně velmi obtížné nalézt vstup, ze kterého je funkce zpočítána – tedy pro danou hodnotu *y* z výpočtu *hash(x) = y*, nalézt jakékoliv *x*.
2. **Odolnost proti kolizím druhého řádu** (slabá odolnost proti kolizím)
   Je výpočetně velmi obtížné nalézt druhý vstup takový, který má stejný otisk zprávy jako první vstup. Tedy řečeno formálněji: nalézt *x'* takové, že *hash(x) = hash(x') = y*. Zajištění, že pro určitý vstup *x* nelze jednoduše najít druhé *x'* takové, že získají stejný výstup *y*.
3. **Odolnost proti kolizím** (silná odolnost proti kolizím)
   Podobný záměr jako v předchozím případě, ale u odolnosti proti kolizím je cílem nalézt jakékoli dva rozdílné vstupy takové, že produkují stejný výstupní otisk. Formálně - najít oba vstupy *x* a *x'* takové, že *hash(x) = hash(x') = y* (volný výběr obou vstupů).

V praxi se nejčastěji setkáváme se selháváním hashovacích funkcí u 2. a 3. podmínky v odolnosti proti kolizím.

1.1.3 Pseudonáhodné generátory

Náhodný generátor je zařízení nebo algoritmus, který produkuje sekvenci statisticky nezávislých dat [4]. Pseudonáhodný generátor (dále také PRNG) resp. pseudonáhodný algoritmus je předpis, který nám produkuje výstup, který se statisticky přibližuje vlastnostem výstupu generátoru náhodných čísel. Je důležité poznamenat, že pseudonáhodný generátor pracuje deterministicky, tedy tento algoritmus sám o sobě nefunguje náhodně - na základě stejných inicializačních hodnot produkuje jednoznačné výstupy.

Kvalitní náhodná data jsou v kryptografii velmi užitečná [6], poskytují plné využití např. pro praxi generování klíčů. Nabízí se otázka – V čem jak vytvořit kvalitní pseudonáhodný algoritmus? A jak objektivně zjistit jeho kvalitu? (viz dále podkap. 4.6).

Bohužel, samotná vytvoření a existence opravdu dokonale náhodného generátoru je nemožná. Lze se k výsledkům náhodných generátorů pouze více či méně přiblížit na základě určitých postupů a ověřením statistických výsledků výstupních dat generátorů. Cílem je nejčastěji produkce výstupu s rovnoměrným rozložením.

Kvalitu pseudonáhodných generátorů lze zjistit sérií statistických testů. Mezi základní testy patří: test dobré shody χ^2, test rovnoměrnosti rozložení, poker test nebo autokorelační test. Je vhodné podotknout, že ani návrh, ani testování kvality pseudonáhodných algoritmů není triviální proces.

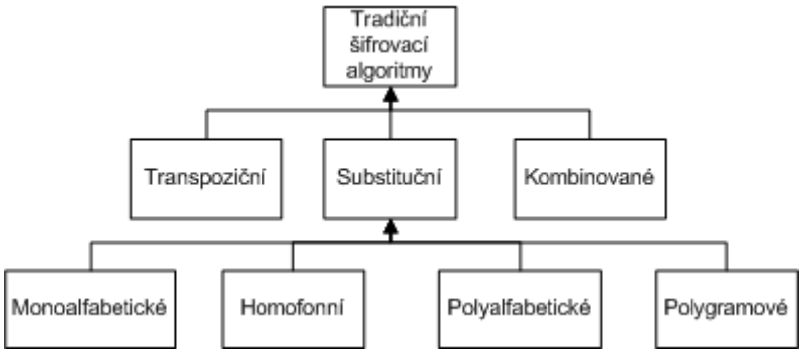
2 Historie a současnost

Člověk odjakživa hledal způsoby a metody, jak skrýt důvěrné údaje před nepřáteli, aby se nedostaly do nepovolaných rukou. Šifrované zprávy mezi spojenci byly vždy doménou vládních a vojenských zájmů, kdy v případě vyzrazení strategických informací mělo za následek často osud celé války a přeneseně i celých národů [7].

Čas prověřil, že informace je obecně jedno z nejdůležitějších aktiv, a že člověk vymyslel mnoho principů, které se později z hlediska bezpečnosti ukázaly být nevhodné. Mnohé šifry odporovaly tzv. Kerckhoffovu a později Shannonovu principu, které nám říkají, že by měl být kryptografický algoritmus a nejlépe celý bezpečnostní systém navržen tak, aby jeho bezpečnost nestála na utajení samotného algoritmu, ale aby se přistupovalo k algoritmu jako k veřejně známému a zabezpečení záviselo pouze na znalosti klíče [8]. Snahy znepříjemnit útočníkovi přístup k metodě byly sice přirozené, z historického pohledu však ne dlouhodobě přínosné a díky metodám reverzního inženýrství bylo často pouze otázkou času, kdy se šifry podařilo analyzovat a najít jejich slabá místa. Co je však zarážející, tyto dlouho známé principy jsou často porušovány i v éře moderní kryptografie – případ symetrické blokové šifry Skipjack (viz také podkap. 4.2.11), která byla utajována a krátce po zveřejnění byla shledána jako nedostatečně bezpečná [9].

2.1 Historická kryptografie

Samotné dochované počátky kryptografie sahají zhruba do období 2000 př. n. l, kdy se ve starověkém Egyptě používaly hieroglyfy záměrně užívající určitou formu substituce [7]. Později, ve starověkém Řecku byla používaná transpoziční šifra, známá jako *Skytalé* [1]. Římský císař Caesar užíval jednu z nejznámějších a nejjednodušších šifer, založenou na jednoduché *monoalfabetické substituci* písmen o 3 znaky v abecedě dopředu (viz také obr. 2-1). Slovo „CAESAR“ se tedy po aplikaci šifry převede na text „FDHVDU“, tedy v zobecněném pohledu *i*-tý znak abecedy je nahrazen *(i+j)*-tým znakem abecedy, analogicky při dešifrování je zašifrovaný znak *i*-tý převeden na *(i-j)*-tý znak abecedy [10].



Obrázek 2-1 Schéma tradiční kryptografie

Nejstarší a nejjednodušší substituční šifry byly náchylné k útoku využívající tzv. *frekvenční analýzu*. Analýza provedla vytvoření tabulky znaků daného jazyka a vychází z předpokladu, že dostatečně dlouhý zašifrovaný text splňuje určité statistické vlastnosti, které jsou shodné s textem rozšifrovaným. Především se jednalo o četnost výskytu jednotlivých symbolů v daném jazyce. Procentuálním ohodnocením jednotlivých symbolů útočník porovná písmena s tabulkovými hodnotami, které jsou pro jazyk charakteristické, a za původní zašifrovaná písmena dosadí písmena, která odpovídají patřičným hodnotám v tabulce. Zjištění neznámého jazyka může útočník provést pomocí tzv. *indexu koincidence*, což je pravděpodobnost, že náhodně vybrané dva znaky šifrovaného textu budou stejné. Vzhledem k tomu, že operace substituce nemá na tuto vlastnost vliv, výsledky testu útočník opět porovná s tabulkou známých indexů koincidence a v případě velmi blízké hodnoty s vysokou pravděpodobností útočník uhodne, že je šifra založena na monoalfabetické substituci a nejblížší tabulková hodnota vypočteného indexu je nejvíce pravděpodobný jazyk. Tento test je znám také jako *Friedmanův test*.

Částečným řešením problému uvedeného typu šifer bylo zavedení *polyalfabetických substitučních šifer*, které řešily nedostatky monoalfabetických substitučních šifer zavedením šifrování n-tic znaků, které omezily možnosti frekvenční analýzy (obr. 2-1). Asi nejznámější středověkou šifrou spadající mezi polyalfabetické substituční šifry byla *Vigenérova šifra*, původně popsaná Giovanem Bellasou v polovině 16. století [10]. Šifra používá tabulku 26 x 26 znaků, kde každý řádek obsahuje abecední posloupnost, posunutý vždy o jeden znak Caesarovou šifrou oproti řádku výše. Tabulka byla nadepsaná písmenem, o který se původní abeceda posouvala, a který zároveň tvořil index do Vigenérovy tabulky (obr. 2-2).

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Obrázek 2-2 Vigénerova tabulka (zdroj: [11])

Klíč zopakujeme až do velikosti otevřeného textu. Šifrování probíhalo jednoduše tak, že písmeno bylo nahrazeno Caesarovou šifrou podle řádku indexovaného písmenem aktuálního symbolu klíče. Příklad viz tabulka:

Vstup	**TAJNYOTEVRENYTEXT**
Klíč	**SIFRASIFRASIFRASI**
Výstup	**LIOEYGBJMRWVDKEPB**

Tabulka 2-1 Vigénerova šifra

Dešifrování proběhne obdobně, hledáme název sloupce v řádku klíče, ve kterém se vyskytuje znak zašifrovaného textu. Matematicky lze šifrování popsat jako:

$$C_i \equiv T_i + K_i \ (mod\ 26) \qquad (2\text{-}1)$$

a dešifrování jako:

$$T_i \equiv C_i - K_i \ (mod\ 26) \qquad (2\text{-}2)$$

kde C_i je zašifrovaný znak, T_i původní znak, K_i znak klíče a konečně *i* je index posunutí od začátku textu.

Vigenerova šifra měla značný nedostatek, text šifry se neustále opakoval. Útok na ní spočíval ve zjištění délky klíče, čímž otevřel možnost klasickému útoku pomocí frekvenční analýzy [10]. Délku klíče lze zjistit pomocí analýzy zašifrovaného textu a zjištění vzdálenosti mezi opakujícími se segmenty zašifrovaného textu. Je vhodné se soustředit na určitá běžná a často používaná krátká slova a najít tak více výskytů a zaznamenat jejich vzájemné vzdálenosti. Pokud najdeme nejmenší společný dělitel vzdáleností všech výskytů, je to pravděpodobná velikost klíče. Máme-li už k dispozici informaci o velikosti klíče, můžeme se soustředit na znaky, které jsou od sebe v zašifrovaném textu vzdáleny právě o tuto velikost klíče, protože jsou šifrovány společně jedním písmenem, a tudíž se prakticky jedná o Caesarovu šifru, na kterou lze již aplikovat známou frekvenční analýzu a Friedmanův test. Tento útok publikoval jako první v roce 1863 Kasiski.

Kromě substitučních šifer je jistě vhodné zmínit také *transpoziční šifry*. Ty fungují na vzájemné výměně znaků v rámci textu. Šifrování lze tedy provést pomocí operace permutace a při dešifrování se na zašifrovanou zprávu aplikuje inverzní operace permutace. Obecně se jedná o použití bijektivní funkce, při které je šifrování textu *x* resp. dešifrování *x'* definováno jako $F(x)$ resp. $F^{-1}(x')$ s určitou periodou. U transpozičních šifer je kryptoanalýza založena na pokusu o původní uspořádání znaků pomocí znalostí n-gramů (bigramy, trigramy,…), Friedmanovým testem ke zjištění jazyka zprávy a frekvenční analýzou zašifrovaného textu k získání správných písmen v daném jazyce.

Počátkem 20. let vytvořil inženýr AT&T Gilbert Vernam teoretický model tzv.Vernammovy šifry (také známé jako *one-time pad*). Šifra je založena na existenci nekonečného zdroje dokonale náhodných dat jako klíče, vstupních dat a operace logického vylučovacího součtu (XOR). Pokud je vstupní informace logicky sečtena operací XOR bit po bitu s klíčem tvořeným náhodnou informací, výstupem je zašifrovaná infromace. Pokud je zašifrovaný text opět sečten operací XOR podruhé, výstupem je opět původní vstupní informace. Jedná se tedy o symetrickou proudovou šifru. Vernamova šifra se stala první a zároveň jedinou, u které byla formálně prokázána neprolomitelnost. Jedná se sice pouze o model, který nelze v praxi sestrojit, lze se však k němu velmi dobře přiblížit kvalitou a dostatečnou délkou generovaných náhodných dat. Hlavní nevýhodou praktického rozšíření je problém bezpečného transportu klíče druhé straně. Neprolomitelnost Vernamovy šifry byla dokázána v roce 1949 již dříve zmiňovaným kryptologem Claudem Shannonem [8].

Posledním algoritmem, o kterém bych se rád okrajově zmínil, a který svou konstrukcí patří mezi jedny z posledních strojů klasické kryptografie, je stroj Enigma, který sehrál velkou roli v průběhu první poloviny 20. století. Enigma byla původně zkonstruovaná ve 20. letech 20. stol a během 2. sv. války vylepšena a používaná německým Wehrmachtem. Z kryptografického hlediska se jedná o algoritmus, založený na substituční šifře (rotory) a transpozici podle časově proměnného algoritmu.

Celý mechanismus byl díky získání stroje spojenci odhalen a jeho složitost řešení tak výrazně klesla, díky zjištění nastavení vnitřního propojení mechanických částí a úspěšně odchycených zprávách [10].

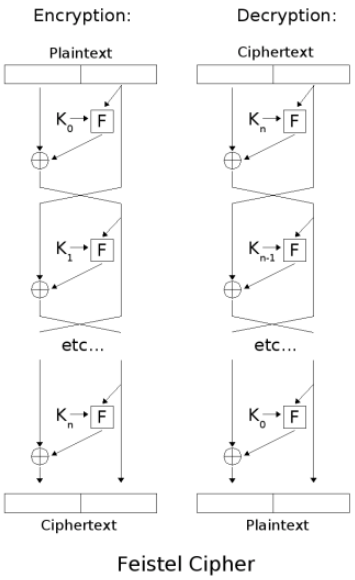
2.2 Moderní kryptografie

Klasická kryptografie ustupovala do pozadí a s rozšiřováním výpočetní síly se objevovaly stále nové a nové návrhy kryptografických algoritmů. Počátky moderní kryptografie sahají do počátků 70. let 20. století, ve kterém kromě rozšíření počítačů vznikla i první asymetrická šifra RSA, která je používaná dodnes a tzv. Feistelovy šifry, na jejíchž základu dnes stojí velký počet používaných symetrických blokových šifer. Velmi důležitým milníkem byla definice šifrovacího standartu DES, založeného na Feistelově šifře, pro veřejnou ochranu důvěrných dat. Co bylo na tehdejší dobu velmi prozíravé, bylo zveřejnění celého šifrovacího principu. Lze říci, že moderní kryptografie většinou důsledně respektuje Kerckhoffův princip.

2.2.1 Feistelova šifra

Horst Feistel navrhl v roce 1973 stejnojmennou blokovou šifru využívající tzv. Feistelovu síť [5]. Je to typ tzv. produkční šifry, jejíchž hlavním principem je opakování některých základních transformací (např. substituce, permutace) nebo operací aritmetiky modulo. Proces šifrování a dešifrování je pak

velmi obdobný a je založen na opakování operace produkční šifry, operace XOR s jejím výstupem a druhé části textu. Každé takové sekvenci se říká *runda*, (anglicky *round*). Rund je v celé šifře vždy několik a každá sestává z opakování sekvence a prohození výsledků jejího výstupu do vstupu rundy další, jak je uvedeno na obr. 2-3.



Obrázek 2-3 Schéma Feistelovy šifry [12]

Každá runda má svůj vlastní dílčí klíč (tzv. *subkey*) K_i, který je odvozen z klíče šifry K. Feistelovu rundu můžeme popsat takto [13]:

$$L_i = R_{i-1} \qquad R_i = (L_{i-1} \oplus F(R_{i-1}; K_i)) \qquad (2\text{-}3)$$

Rovnice 1 - Feistelova šifra

Pro i platí: $1 \leq i \leq r$, což je počet rund. L_0 a R_0 jsou části vstupního bloku, $\oplus$ je operace XOR a budeme ji pro zjednodušení používat také dále. Důležitou informací je fakt, že je rundová funkce F(x,y) bijektivní, dešifrování je tedy provedeno stejným způsobem jako šifrování s tím rozdílem, že jsou dílčí klíče použity v opačném pořadí od K_r po K_1. Rundová funkce kromě výše zmíněného splňuje vlastnosti kryptograficky bezpečného pseudonáhodného generátoru (viz konfůze a difůze v další kapitole).

2.2.2 Data Encryption Standard

Data Encryption Standard (DES) patří mezi nejstarší a bezesporu také nejznámější šifrovací algoritmy z třídy Feistelových šifer. Další šifry podle Feistelova schématu, kterými jsou např. Skipjack, 3DES nebo rodina šifer RC, rozeberu v kapitole 4.2.

DES pracuje s šifrou dlouhou 56 bitů + 8 bitů parity a šifruje 64-bitové bloky a provádí 16 rund. Schéma DESu nově zavádí tzv. S-boxy, což je důmyslně navržená substituční tabulka 4-bitových hodnot, které slouží pro nahrazení vstupních hodnot konstantami [4]. Zavedení substituční tabulky bylo předmětem mnoha diskuzí. Nicméně jejím hlavním úkolem je zavedení do algoritmu nelineárního prvku, jejíchž důvod byl popsán Shannonem v r. 1949 (viz také 3.2).

$$L_i = R_{i-1} \qquad R_i = (L_{i-1} \oplus f(R_{i-1};K_i)) \tag{2-4}$$

$$f(R_{i-1}; K_i) = P(S(E(R_{i-1}) \oplus K_i)) \tag{2-5}$$

Rovnice 2 – Schéma šifry DES

Zpátky k samotnému schématu činnosti (viz také Rovnice 2). Pro každou rundu vygeneruje z klíče K 48-bitový subklíč. Šifra provede nad blokem úvodní permutaci $IP(x)$, kterou rozdělí 64-bitový vstup na dva 32-bitové vstupy L_0 a R_0. Funkce $E(x)$ provádí expanzi 32-bitového z vstupu x na 48-bitový výstup. Funkce $S(x)$ provádí pro každý 6-bitový vstup mapování, ze kterého jsou použity 2 vnější bity resp. 4 vnitřní bity jako indexy do řádků resp. sloupců v tabulce S-boxů. Výstupem této mapovací funkce je 4-bitový údaj z tabulky.

Konečně funkce $P(x)$ nám provádí permutaci nad 32-bitovým vstupem. Algoritmus DES nakonec provede inverzní permutaci $IP^{-1}(x)$ nad výsledkem z rund $R_{16}L_{16}$. Takto jsme zašifrovali 64-bitový blok vstupních dat šifrou DES. Přehled současného stavu kryptoanalýzy šifry DES bude popsán v podkapitole 4.2.2.

2.2.3 RSA, Knapsack

V roce 1977 byl publikován mechanismus šifrování pomocí veřejného klíče trojicí vědců Rivest, Shamir a Adleman o novém přístupu v kryptografii, který otevírá zcela nové možnosti bezpečnosti, které nebyly do tehdejší doby řešitelné.

Matematický princip spočívá nejprve v nalezení dostatečně veliké dvojice prvočísel p a q. Provede jejich součin $p \cdot q = n$. Z čísla n vypočte hodnotu Eulerovy funkce $\varphi(n) = (p - 1)(q - 1)$. Zvolíme pak číslo e takové, že $\varphi(n)$ je s e nesoudělné. Pokud nalezneme d takové, že $d \cdot e \equiv 1 \pmod{\varphi(n)}$, získáme soukromý klíč. Klíčový pár je pak charakterizován dvojicí údajů pro soukromý klíč (n,d) a pro veřejný klíč (n,e) [13].

Zprávu c zašifrujeme takto: $c = m^e \bmod n$ dešifrování provedeme $m = c^d \bmod n$. Problémem, na kterém šifra RSA stojí, souvisí s neschopností řešit faktorizaci čísla n v rozumném čase. RSA není jediným problémem, na kterém je postavena asymetrická kryptografie. Známým problémem je také např. problém Knapsack – problém závaží a váhy batohu [14]. U něj se pokoušíme zjistit, která závaží jsou uvnitř batohu, známe-li jeho celkovou váhu (zanedbáme-li samotnou váhu batohu) a váhy jednotlivých závaží. Tento problém se však ukázal jako nedostatečný pro kryptografii, protože se našly metody, které tento problém částečně řeší.

Objevem a využitím asymetrické kryptografie, kdy se pro šifrování a dešifrování používají odlišné klíče, se světu otevírají nové možnosti a bezpečnosti, které symetrická kryptografie obsáhnout nedokázala. Asymetrická kryptografie umožňuje s použitím šifrování veřejným i soukromým klíčem kromě důvěrnosti naplnit také další bezpečnostní cile, jako jsou: nepopíratelnost, integrita a autentizace.

Ukázal jsem základní přehled principů i algoritmů z oblasti klasické i moderní kryptografie. Nyní bude potřeba definovat teoretické kryptografické základy abych se mohl následně hlouběji věnovat samotnému rozboru kryptografických algoritmů, nejčastěji užívaných v praxi.

3 Teoretické základy kryptografie

Moderní kryptografie staví na pevných formálních matematických základech. Kryptografie je z teoretického hlediska souhrn určitých matematikých modelů, jejichž návrh by měl být dostatečně odolný proti metodám kryptoanalýzy – hledání metod, které jsou schopny řešit určitý kryptografický problém. Ústředním problémem, na kterém kryptografie stojí, je především *složitost*. Právě výpočetní složitost je klíčovým pojmem, který dnes z praktického hlediska brání v prolomení algoritmů, a na němž kryptografie stojí.

3.1 Složitost

Pojem složitost vychází z formálního výpočetního stroje, nazvaného *Turingův stroj*, který je formálním modelem programovatelného stroje. Historicky bylo navrženo více formálních modelů strojů, nicméně bylo dokázáno, že jsou s Turingovým strojem výpočetně ekvivalentní, nebo slabší. Omezím zde formální popis a definici Turingova stroje - Turingův stroj obsahuje nekonečnou pásku, skládající se z políček, a čtecí/zapisovací hlavy, která se nachází právě na jednom políčku pásky. Stroj pak může provést načtení nebo zapsání symbolu na políčko, pod kterým je hlava. Dále pak může posunout hlavu nad páskou buďto doleva, nebo doprava. Políčka mohou být buďto prázdná, anebo mohou nést symboly z konečné množiny prvků. Páska pak v počátečním stavu určuje program, který definuje akce, které má stroj provést. Algoritmus je pak skrytý v programu Turingova stroje, který jej takto matematicky definuje.

Z definice *Church-Turingovy teze* platí: *Každý výpočetní postup (algoritmus) je možné realizovat na Turingově stroji*. Výpočetní postup – algoritmus - je obvykle pak řešení algoritmického problému. Příklady: Zjisti nejkratší cestu v grafu, seřaď konečnou posloupnost čísel nebo také: *Pro odpovídající definované vstupy a výstupy najdi klíč šifry XYZ*. Turingův stroj je pak matematicky formální model určitého programu, který sám vykonává. Pro naše potřeby se budu soustředit pouze na řešitelné problémy s hlavním důrazem na složitost takového řešení. K tomuto účelu zevrubně definuji složitost a ukážu složitostní třídy. Pro potřeby této sekce popíšu rozdíly mezi tzv. nedeterministickými resp. deterministickými Turingovy stroji. Deterministický Turingův stroj (dále také DTS) obsahuje v přechodové funkci právě jeden přechod (změna stavu, zápis/čtení symbolu, posunutí hlavy), zatímco v nedeterministickém TS (dále také NTS) je možné definovat více možných přechodů, z nichž si stroj vybere jeden na základě náhody. Nedeterminismus je z hlediska složitosti klíčovým pojmem.

Složitost je nejčastěji definována jako časová a paměťová (prostorová) a v souvislosti s Turingovými stroji ji definujeme takto (přesná definice [15]).

Časová složitost popisuje počet kroků TS provedený od počátku do konce výpočtu.

Paměťová složitost pak definuje počet buněk pásky Turingova stroje, který je požadován pro výpočet daného algoritmu.

3.1.1 Asymptotická omezení složitosti.

Mějme F množinu funkcí f : N → N. Pro danou funkci $f \in F$ definujeme množiny funkci $O(f(n))$, $\Omega(f(n))$ a $\theta(f(n))$ jako [15]:

Asymptotická horní omezení funkce f(n) je množina

$$O(f(n)) = \{g(n) \in F \mid \exists c \in R^+, n_0 \in N\ \forall n \in N : n \geq n_0 \Rightarrow 0 \leq g(n) \leq c \cdot f(n)\} \quad (3\text{-}1)$$

Asymptotická dolní omezení funkce f(n) je množina

$$\Omega(f(n)) = \{g(n) \in F \mid \exists c \in R^+, n_0 \in N\ \forall n \in N : n \geq n_0 \Rightarrow 0 \leq c \cdot f(n) \leq g(n)\} \quad (3\text{-}2)$$

Asymptotická oboustranné omezení funkce f(n) je množina

$$\theta(f(n)) = \{g(n) \in F \mid \exists c_1, c_2 \in R^+, \exists n_0\ \forall n \in N : n \geq n_0 \Rightarrow 0 \leq c_1 \cdot f(n) \leq g(n) \leq c_2 \cdot f(n)\} \quad (3\text{-}3)$$

Asymptotická omezení složitosti nám umožňují algoritmy klasifikovat do určitých složitostních tříd, kde algoritmy zobecníme na úroveň složitosti problémů. V souvislosti s kryptografickými algoritmy a kryptoanalýzou uvažujeme nejčastěji složitost s asymptotickým horním omezením $O(f(n))$. K definicím složitostních tříd nyní potřebujeme ještě definovat paměťovou a časovou složitost.

Mějme funkce t,s: N → N. Funkce T_M a S_M značí časovou resp. paměťovou složitost Turingova stroje M, *DTS* a *NTS* značí deterministické resp. nedeterministické Turingovy stroje [15]:

$$DTime(t(n)) = \{L \mid \exists\, DTS\ M : L = L(M)\ a\ T_M \in O(t(n))\}. \quad (\text{Def. } 3\text{-}4)$$

$$NTime(t(n)) = \{L \mid \exists\, NTS\ M : L = L(M)\ a\ T_M \in O(t(n))\}. \quad (\text{Def. } 3\text{-}5)$$

$$DSpace(s(n)) = \{L \mid \exists\, DTS\ M : L = L(M)\ a\ S_M \in O(s(n))\}. \quad (\text{Def. } 3\text{-}6)$$

$$NSpace(s(n)) = \{L \mid \exists\, NTS\ M : L = L(M)\ a\ S_M \in O(s(n))\} \quad (\text{Def. } 3\text{-}7)$$

kde L je jazyk přijímaný Turingovým strojem M (3-4 až 3-7, v praxi také hovoříme o schopnosti Turingova stroje řešit určitý problém) a zároveň jeho časová resp. paměťová složitost odpovídá danému omezení. Definice zakončeme vymezením pojmů složitostních tříd.

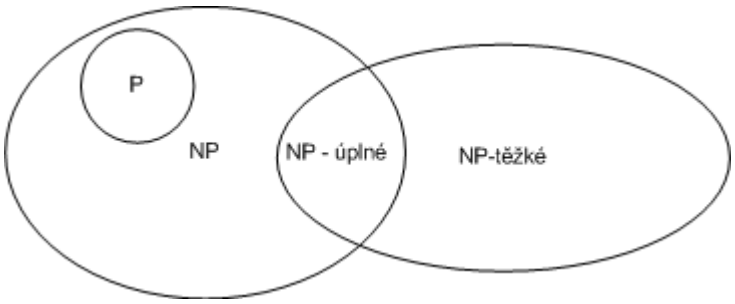
3.1.2 Složitostní třídy

Definice zakončím výčtem nejznámějších složitostních tříd [15], které se v souvislosti s kryptografickými algoritmy používají.

$\boldsymbol{P} = \bigcup_{k=0}^{\infty} DTime(n^k)$	$\boldsymbol{NP} = \bigcup_{k=0}^{\infty} NTime(n^k)$
$\boldsymbol{EXP} = \bigcup_{k=0}^{\infty} DTime(2^{n^k})$	$\boldsymbol{NEXP} = \bigcup_{k=0}^{\infty} NTime(2^{n^k})$
$\boldsymbol{LOGSPACE} = \bigcup_{k=0}^{\infty} DSpace(k \cdot \lg(n))$	$\boldsymbol{NLOGSPACE} = \bigcup_{k=0}^{\infty} NSpace(k \cdot \lg(n))$
$\boldsymbol{PSPACE} = \bigcup_{k=0}^{\infty} DSpace(n^k)$	$\boldsymbol{NPSPACE} = \bigcup_{k=0}^{\infty} NSpace(n^k)$
$\boldsymbol{EXPSPACE} = \bigcup_{k=0}^{\infty} DSpace(2^{n^k})$	$\boldsymbol{NEXPSPACE} = \bigcup_{k=0}^{\infty} NSpace(2^{n^k})$

Tabulka 3-1 Vybrané třídy složitosti [15]

Tabulka Tabulka 3-1 nám vymezuje základní složitostní třídy. Pro vymezení základních vztahů mezi kryptografií nám postačí základní třídy *P* a *NP*, které ještě rozšíříme o definice několika dalších tříd složitosti, které budeme v této kapitole používat. Jedná se o třídu *NP-těžkých* problémů a *NP-úplných* problémů (viz také def. 3-1).



Obrázek 3-1 Předpokládaný vztah vybraných složitostních tříd

1. *NP-těžké* jsou takové problémy, na které lze převést libovolné problémy z třídy *NP*.
2. *NP-úplné* problémy jsou takové problémy, které jsou zároveň *NP-těžké* a zároveň patří do třídy *NP*.

Definice 3-1 - NP-úplné a NP-těžké problémy

Z definice 3-1 plyne, že NP-úplné problémy mají tu vlastnost, že jsou vzájemně převoditelné. Pro lepší ilustraci jsem znázornil vztahy vybraných tříd na obrázku 3-1 (není přesně jasné o vztahu P a NP viz 3.1.3).

3.1.3 Souvislost složitosti a šifrování

Pro praktické algoritmy/problémy (nepočítám-li složitostní třídy jako např. třídy logaritmické) nám postačí třída *P*. Třída *P* vymezuje praktické výpočetní problémy. Platí vztah $P \subseteq NP$, tedy že *P* je podmnožinou třídy *NP*. O vztahu (ne)rovnosti tříd *P* a *NP* není doposud známo. Určitou útěchou pro kryptografii se všeobecně předpokládá, že rovnost tříd neplatí [16]. V oblasti kryptografie by rovnost obou tříd implikovala existenci algoritmů kryptoanalýz, které by všechny šifry luštily v polynomiálním čase, což by po objevení odpovídajících algoritmů stávající šifry přivedlo k zániku.

Z definice 3-1 také dále plyne, že pokud bychom znali polynomiální algoritmus pro kterýkoli NP-úplný problém, mohli bychom pak vzájemně transformovat kterýkoli NP problém a najít polynomiální algoritmus pro jeho řešení - a platí to tedy i o šifrovacích algoritmech (diskrétní logaritmus, faktorizace), jejichž složitost patří do *NP*.

Dalším možným rizikem z pohledu praktické složitosti by mohlo být sestrojení stroje, který by nesl některé vlastnosti NTS, příklad kvantového počítače (viz také kapitola 4.3.2). Takový stroj by pak dokázal v polynomiálním čase řešit kryptografické problémy spadající do třídy *NP*, bez ohledu na aktuální stav výzkumu vztahů *NP* a třídy *P*.

3.2 Konfúze, difúze a lavinový efekt

Matematik Claude Shannon popsal v roce 1949 teoretické vlastnosti systémů, které se mj. dotýkají veškerých symetrických šifer i hašovacích funkcí [17]. Konkrétně Shannon popisuje [1] a definuje vlastnosti tzv. konfúze resp. difúze (DES i některé jiné šifry je ustáleně zajišťují jako funkce *S-BOX(x)* resp. *P-BOX(x)*, které se poprvé objevují s nástupem blokových šifer (viz také DES v podkap. 2.2.2).

Konfúze nám definuje složitost závislosti, jak jednotlivé vstupní bity klíče pronikají do jednotlivých výstupních bitů šifrovaného textu. Difúze pak hovoří konkrétně o závislosti vstupního na výstupním šifrovaném textu.

Tímto je zajištěno, že má výstup funkcí permutačních i substitučních funkcí při několikanásobné aplikaci charakter výstupu pseudonáhodné funkce – díky tzv. lavinovitém efektu konfúze a difúze je dosáhnuto stavu, kdy změna jediného vstupního bitu klíče nebo vstupního textu změní polovinu bitů výstupu.

4 Detailní popis a útoky

Existuje celá řada kryptoanalytických metod, využívaných k útokům na známé algoritmy. Rád bych nejprve popsal principy a techniky některých základních kryptoanalýz, které se objevují častěji a ze kterých vychází další možné útoky. Je pravděpodobné, že určité organizace, které by mohly mít zájem na narušení bezpečnostního cíle - důvěrnost, mohou disponovat světu neznámými a utajovanými technologiemi a prostředky, které nechávají známé metody kryptoanalýzy daleko za sebou. Do této oblasti se ale pouštět nebudu - hodlám především vycházet z obecně a veřejně známých kryptoanalytických metod a znalostí. Každý algoritmus bude kromě nalezených analýz také z bezpečnostního hlediska na konci zkonfrontován pohledem podkapitoly 4.5.

4.1 Moderní kryptoanalýza

Moderní kryptoanalytické metody se opírají o skupinu možných technik útoků, které se snaží napadnout sílu kryptografického mechanismu a najít jeho slabiny, kterými se zjednoduší útoky podle charakteru kryptografického algoritmu. V praxi se takové činnosti říká kryptoanalýza (hovorově také lámání). Útoky moderní doby jsou zajímavé a komplexnější především v tom, že lze provádět kromě pasivního poslechu komunikace také aktivní zapojení útočníka do komunikace. Zejména v komunikačních protokolech lze provést např. útok Man-in-the-middle, při kterém útočník přesměruje komunikaci každé legitimní strany na sebe, které se mylně domnívají, že v pořádku komunikují se stranou druhou, zatímco útočník provádí oboustranný odposlech.

4.1.1 Diferenciální kryptoanalýza

Tato kryptoanalýza byla poprvé představena kryptology Bihamem a Shamirem v roce 1990. Diferenciální kryptoanalýza [18] byla navržena pro analýzu algoritmů podobné DES – tedy iterativní blokové šifry, založené na Feistelově síti a lze ji použít taky proti některým hashovacím funkcím. Vychází z předpokladu, že je útočník schopen vybrat nezašifrovaný text a zároveň získá i jeho zašifrovaný výsledek (chosen plaintext attack – CPA, v souvislosti se složitostí bude v této práci zkratka CPA vyjadřovat také počet vybraných nezašifrovaných zpráv). Cíle útočníka závisí na zjištění diferencí mezi dvěma souvisejícími vstupními texty, které jsou zašifrovány stejným klíčem. Při dostatečně velkém počtu dostupných dat získáme klíč, který bude identifikován jako nejpravděpodobnější. Např. u algoritmu DES lze diferenciální kryptoanalýzou získat klíč ze 2^{47} zvolených zpráv, který je pro DES z praktického hlediska neefektivní.

4.1.2 Lineární kryptoanalýza

Byla představena v roce 1992 japonským kryptologem Mitsuru Matsuim [19] a od té doby byly zaznamenány úspěšné útoky na některé blokové a proudové šifry. Je založena na snaze vyjádřit šifrovací algoritmus jako rovnici, ze které lze získat určitým způsobem klíč. Metoda hledá lineární aproximaci pro nelineární funkci – šifrovacího algoritmu. Jedná se o útok s předpokladem známého textu (known plaintext attack – KPA, v souvislosti se složitostí bude dále v této práci zkratka KPA vyjadřovat také počet známých zpráv pro dosažení útoku). V útoku na šifru DES je účinnější než diferenciální kryptoanalýza, k úspěchu potřebuje pouze 2^{43} dvojic plaintextů a k němu patřičný počet šifrovaných zpráv.

4.1.3 Útok hrubou silou

Pod tímto typem útoku si nepředstavujme nic jiného než vyzkoušení všech možných variant klíčů. Jedná se o návrhově nejjednodušší, ale zato v praxi nejdéle trvající útok, v závislosti na počtu vyzkoušených možností a na metodě, jakým se klíč ověřuje. Pro každý pokus potřebuje útočník ověřit, zda-li byl klíč opravdu nalezen. V ideálním případě tento typ útoku vyžaduje získání KPA. Kryptoanalýza hledá přednostně možnosti, jak prolomit kryptografickou metodu rychleji, než je právě útok hrubou silou.

4.1.4 Narozeninový útok

Výše uvedené útoky byly doménou především šifrovacích algoritmů. Narozeninový útok [4], spočívající v narozeninovém paradoxu, se soustřeďuje na hledání kolizí v hashovacích funkcích a je prakticky použitelný proti každé hashovací funkci. Narozeninový paradox spočívá v otázce, kolik lidí je potřeba shromáždit, aby se dalo aspoň s 50% pravděpodobností odhadnout, že jsou v sále aspoň 2 lidé se stejným datem narození. Převedeno do řeči kryptografie – mějme hashovací funkci s pevným 128 bitovým otiskem. Kolik vstupních textů bychom měli vyzkoušet, abychom získali s pravděpodobností větší než 50% alespoň 2 stejné otisky - kolizi?

Narozeninový paradox lze definovat jako $x(N) \approx (0.5 \prod \cdot N)^{1/2}$, kde $x(N)$ je očekávaný počet hodnot, než dojde k pravděpodobné kolizi a N je počet možných výstupů hashovací funkce. Pro 2^{128} teoretických otisků hashovací funkce je to tedy pouze cca 2^{64} pokusů. V případě, že nalezneme způsob, jak nalézt kolize, jednodušeji, jedná se o prolomení hashovací funkce.

4.2 Symetrické šifrovací algoritmy

4.2.1 Režimy činnosti blokových šifer

Termínem *režim činnosti* (v jiné literatuře také *operační módy*) blokových šifer se rozumí způsob aplikace jednotlivých operací šifrování na vstupní text (resp. operace dešifrování na zašifrovaný text) o délce zpravidla více než jeden blok [1]. Režimy činnosti jsou ze své podstaty aplikovatelné prakticky na jakoukoli blokovou šifru – samotná operace šifrování/dešifrování bloku textu je z pohledu režimu činnosti blokových šifer atomická operace v rámci celého procesu šifrování. Důvodem pro zavedení různých režimů bylo především zajištění bezpečnosti jako celku – požadované náhodné statistické vlastnosti výsledného zašifrovaného textu jako celku by jednoduché zašifrování bloků po sobě (ECB) neumožňovalo. Rád bych zmínil nejznámější a nejpoužívanější řežimy.

ECB (electronic code book)

Nejzákladnějším režimem činnosti blokových šifer je ECB (tzv. kódová kniha), který šifruje vstupní text rozdělený na bloky stejným klíčem nezávisle na dalších blocích. Metoda se dá podle názvu přirovnat k tzv. elektronické kódové knize, ve které mají vždy stejně zašifrované bloky stejný obraz a naopak. Výhodou tohoto režimu je okamžitý přístup k jednotlivým blokům.

Tento režim šifrování je jediným režimem, který nelze doporučit právě díky nedostatečné náhodnosti výsledného ciphertextu a možné manipulaci s bloky – útočník může do zašifrované zprávy vložit nebo odejmout bloky textu, pokud není zajištěna dodatečná kontrola integrity textu jinak [1]. I bez znalosti klíče dokáže útočník poměrně snadno napáchat velké škody. Situace je pak tím horší, čím kratší je délka šifrovaného bloku.

CBC (cipher block chaining)

Režim CBC zavádí některé nové pojmy. Především je to tzv. *inicializační vektor* (IV), který je nejčastěji náhodná konstanta velikosti šifrovaného bloku. Princip CBC pak spočívá v provedení operace XOR vstupního textu se zašifrovaným blokem výsledku předchozího kroku. IV je pak první vstupní hodnota pro operací XOR s prvním blokem vstupního textu. Zavedením tohoto schématu docílíme, že se se změnou inicializačního vektoru změní také výsledný zašifrovaný text, byť byl vstupní plaintext stejný. Dešifrování se provede analogicky – dešifruje se zašifrovaný blok a provede se operace XOR s předchozím blokem. K de/šifrování je tedy třeba narozdíl od ECB provést ze 2 bloků zprávy (viz algoritmy 4-2 a 4-3).

$$Z_0 = IV \qquad \text{(alg. 4-1)}$$

pro šifrování

$$Z_x = Zašifrovat(T_x \oplus Z_{x-1}) \qquad \text{(alg. 4-2)}$$

pro dešifrování

$$T_x = (Dešifrovat(Z_x) \oplus Z_{x-1}) \qquad \text{(alg. 4-3)}$$

kde Z_x je zašifrovaný text, T_x zdrojový text, operace *Zašifrovat(x)* a *Dešifrovat(x)* provádí šifrování resp. dešifrování bloku *x* blokovou šifrou.

Právě závislost CBC na předchozích blocích spolu s operací XOR, odvíjející se od inicializačního vektoru poskytuje náhodnost celého šifrovaného textu textu i při zcela nenáhodné charakteristice vstupní zprávy. Z hlediska důvěrnosti je metoda CBC považována jako bezpečná [20]. Bohužel díky závislosti bloků na svém předchůdci se v případě chyby v jednom bloku rozšíří chyba také do následujícího bloku a velikost zprávy je potřeba zarovnávat na násobek velikosti bloku šifry. Přesto je tato metoda velmi populární – je využívána např. v některých verzích protokolu SSL a vychází z ní také autentizační metoda MAC, kterou lze použít pro zajištění integrity a také k autentizaci původu dat (víz dále).

CFB (cipher feedback)

Na rozdíl od CBC, režim CFB šifruje místo bloku otevřeného textu předchozí (zašifrovaný) blok, na který je až poté aplikována operace XOR s blokem vstupního textu. [21]. Prvním blokem k zašifrování je *IV,* známá už z režimu CBC.

$$Z_0 = IV \qquad \text{(alg. 4-4)}$$

pro šifrování

$$Z_x = Zašifrovat(Z_{x-1}) \oplus T_x \qquad \text{(alg. 4-5)}$$

pro dešifrování

$$T_x = Zašifrovat(Z_{x-1}) \oplus Z_x \qquad \text{(alg. 4-6)}$$

kde Z_x je zašifrovaný text, T_x zdrojový text, operace *Zašifrovat(x)* a *Dešifrovat(x)* provádí šifrování resp. dešifrování bloku *x* blokovou šifrou.

OFB

Dalším režimem z dlouhé řady operačních módů bych zmínil tzv. synchronní proudovou šifru OFB. Mechanismus pracuje v prvním kroku se zašifrováním *IV*, který jako mezivýsledek předává jednak jako vstup pro další blok zprávy a jednak pro operaci XOR s blokem textu zprávy, čímž je získán výsledný zašifrovaný blok [22]. Označení pro synchronní proudovou šifru si tento algoritmus získal kvůli nápadně podobnému mechanismu u proudových šifer, které stejně jako OFB a podobně u CFB vytvoří klíčový proud, kterým posléze XORují bloky vstupní zprávy.

$$V_0 = IV \qquad \text{(alg. 4-7)}$$

$$V_x = Zašifrovat(V_{x-1}) \qquad \text{(alg. 4-8)}$$

pro šifrování

$$Z_x = T_x \oplus V_x \qquad \text{(alg. 4-9)}$$

pro dešifrování

$$T_x = Z_x \oplus V_x \qquad \text{(alg. 4-10)}$$

Režimy OFB a CFB jsou náchylné k útokům, které směřují ke změně textu [23]. Prostým nahrazením zašifrovaného bloku zprávy pomocí $Z_x \oplus$ podvržený text dokáže útočník efektivně změnit výsledný text. Dodatečné zajištění integrity je tedy v případě použití těchto režimů opět nutností. Oproti CBC však zmíněné režimy nepotřebují být doplněny na velikost násobku zašifrovaného bloku.

GCM (Galois/Counter Mode)

Posledním režimem, který bych zde rád zmínil je jeden z nejnovějších – z nové doby (z roku 2005 jako standart definován v RFC 4106) – GCM (viz také [24]). Jeho zajímavostí je fakt, že kromě šifrování dokáže také provést autentizaci. Samotný algoritmus obsahuje 2 základní operace – *autentizační šifrování* a *autentizační dešifrování*. Vstup pro šifrování GCM sestává z:

1) Klíč patřičné šifry (dále K)
2) Vstupní text (dále P)
3) Inicializační vektor (dále IV)
4) Dodatečná autentizační data (dále A) – data, která jsou autentizovaná, ale nejsou šifrovaná.

Výstupem funkce je pak jednak šifrovaný text C, a také tzv. autentizační tag T s variabilní délkou (0 až 128 bitů). K autentizačnímu dešifrování vstupují pak hodnoty K,C, IV, A a T. Výstupem pro operaci autentizačního dešifrování je pak text P, nebo informace, že vstup není autentický. Kromě GCM existují také jiné autentizační režimy jako OCM (Offset codebook mode) nebo CCM (Counter with CBC-MAC), důvodem pro zmínění je fakt, že je tento mechanismus ve hře o začlenění do nového standartu pro protokol TLS v1.2 (viz také kap. 8.6).

Režimy činnosti blokových šifer přímo ovlivňují míru bezpečnosti celé zprávy. Proto je v implementovaných protokolech a bezpečnostních systémech důležitý nejenom výběr bezpečných algoritmů, ale také výběr vhodného řežimu, kterým je zabezpečena celá zpráva.

Nyní se již mohu věnovat provedeným kryptoanalýzám a postupně rozboru jednotlivých blokových šifer.

4.2.2 DES

Charakter šifry DES byl popsán v dřívějších kapitolách. Útoky na něj po stránce diferenciální (dále také DK) a lineární kryptoanalýzy (dále také LK) však nelze veřejně známými prostředky jednoduše provést. Z praktického hlediska byl však DES prolomen pouze útoky hrubou silou a díky malé velikosti klíče zůstává velikost klíče 56-bitů nejslabší část celého systému DES. Jako nejnovější a nejzajímavější lze považovat [25] velmi obecný útok pomocí algebraického řešení šifry DES jako systémů rovnic o velkém počtu neznámých [26], převedeného na problém řešení SAT-problému (který je NP-úplný - viz také 3.1.2). Na rozdíl od DK a LK nepotřebuje tento typ útoků velké množství zašifrovaného textu – výsledky dokázaly velmi rychle vyřešit 6 rund DESu pouze s 1 blokem plaintextu. Autoři se domnívají, že vzhledem k obecnému charakteru půjde tento typ útoku dobudoucna využít také na některé další šifry.

Z méně praktických útoků bych zmínil zatím nejúspěšnější Biryukovův útok [27], který vylepšil Matsuiho lineární kryptoanalýzu a vyžaduje 2^{41} KPA.

4.2.3 3DES

Zmíněná krátká velikost klíče algoritmu DES přiměla už v roce 1978 IBM vytvořit jeho novou variantu, založenou na jednoduchém principu rozšíření klíče o 2 další a použít stejný algoritmus. Algoritmus 3DES (Triple DES) je nešťastné souhrnné označení pro dvě jeho varianty: 2TDES a 3TDES, užívající 2 resp. 3 různé klíče DES (112 bitů a 168 bitů dlouhých) . Šifra má 48 rund (3 x 16 původního DES) a pracuje s 64-bit blokem. Rozdíly mezi variantou 3TDES a 2TDES tkví v použití stejného klíče pro 2 šifrování. Funkce 3DES lze ilustrovat takto:

$$3DES(k_0k_1k_2) \sim DES(k_2,DES(k_1,DES(k_0,B)))$$

Schéma 4-1 Princip šifry 3DES

kde k_0, k_1 a k_2 jsou klíče velikosti 56-bitů, B vstupní blok dat a DES šifrování bloku operace šifrou DES.

Útok na alternativu 2DES byl popsán už v roce 1977 kryptologem Diffiem [4], který ukázal, že zdvojnásobením délky klíče se prakticky složitost odpovídajícím způsobem nezvyšuje, protože lze provést tzv. meet-in-the-middle útok. Útok spočívá v pokusu zašifrovat text pro všechny klíče původní šifry DES tak, aby bylo dohledatelné, který zašifrovaný text byl vytvořen kterým klíčem (počet záznamů je tedy 2^{56}) a tyto hodnoty porovnat s dešifrovanými hodnotami zašifrovaného textu všech možných klíčů tohoto textu celou šifrou. Útok tedy hledá správný mezivýsledek, ve kterém se takto strany „setkají“ a pokud se shodují, zjistíme výsledný klíč. Tímto útokem omezíme výpočetní složitost z 2^{112} na $2 \cdot 2^{56}$. Nevýhoda útoku spočívá především v nutnosti uložení velkého množství 2^{56} mezivýsledků, který je i v dnešních podmínkách nereálný. Tento útok lze provést na všechny typy podobně koncipovaných šifer.

Nejlepší útok na 3DES [28] v současné době potřebuje 2^{90} DES operací a 2^{32} KPA, tudíž algoritmus 3DES je pro nynější účely považován stále jako dostatečně bezpečný, nicméně pro symetrickou kryptografii je kvůli slabšímu výkonu nahrazován rychlejšími alternativami.

4.2.4 International Data Encryption Algorithm

International Data Encryption Algorithm (dále také známá pod zkratkou IDEA) byla vytvořena v roce 1990 kryptology Xuejia Lai a Jamesem Massey a díky jejímu rozšíření prostřednictvím známých protokolů jako PGP, či SSH je také jednou z nejvíce podporovaných blokových šifer vůbec. IDEA byla původně známá pod názvem Proposed Encryption Standard [1], ale po roce kdy kryptologové Shamir a Biham provedli útok pomocí diferenciální a lineární kryptoanalýzy, autoři zesílili algoritmus a přejmenovali šifru na dnes již známou - IDEA. IDEA používá 128-bitové klíče a 64-bitovou délku bloku.

Schneier svého času hodnotil tuto šifru jako nejbezpečnější blokovou šifru [1]. Nejnovější a nejpokročilejší kryptoanalýza [29] i po více než 15 letech její existence potvrzuje, že toto oprávněný názor byl. Pominu-li třídu 2^{64} slabých klíčů, pak složitost útoku na 7 z celkových 8.5 rund, které šifra používá, je z nich 2^{43} KPA a 2^{115} šifrovacích operací.

Šifru IDEA bych hodnotil jako další bezpečnou blokovou šifru, které byly prověřené časem.

4.2.5 RIJNDAEL (AES)

Rijndael je známou blokovou šifrou, navrženou belgickými kryptology Joanem Daemenem a Vincentem Rijmenem v roce 1998. Šifra se stala vítězem soutěže o nový šifrovací standart AES (Advanced Encryption Standard) ve výběrovém řízení pořádaném institucí NIST v letech 1997 až 2000 a nahradila tak původní algoritmus DES [30]. Rijndael používá 128-bitový blok a klíče o velikosti 128, 192 nebo 256 bitů. Podle délky klíče je zvolen také počet rund – 10, 12 nebo 14 rund [31]. Schéma šifry Rijndael vychází ze šifry Square a lze jej rozdělit na tyto hlavní části:

1. Vygenerování rundových klíčů, inicializace pomocí operace XOR sloupců bloku s rundovými klíči.
2. Runda:
    a) SubBytes – V prvním kroku se provede nelineární substituce, známá již z prvních šifer, používající S-box. S-box je adresovatelný 8 bitovou hodnotou.
    b) ShiftRows – S každým řádkem provede cyklický posun o 1 bajt doleva více, než na řádku předchozím. Postupně tedy posune řádky o 0, 1,2 a 3 bajty doleva.
    c) MixColumns – Postup užitý k dosažení difůze. Tento krok se aplikuje na sloupce, kdy se sloupec vynásobí polynomem $3x^3 + x^2 + x + 2$ a provede operace modulo (x^4+1)
    d) AddRoundKey – Operace XOR sloupců s rundovými klíči.
3. Nakonec se provede finální runda, která neobsahuje operaci MixColumns.

V souvislosti s útoky [31] na šifru Rijndael se hovoří především o útocích využívajících tzv. postranních kanálů, tj. útoky, které nesouvisí přímo s návrhem šifry, ale dalšími vrstvami - jeho implementací a použitím šifry. Dále je zmíněn problém nově objeveného útoku z r. 2002 – tzv. XSL (eXtended Sparse Linearization), který může snížit složitost šifry na 2^{100} operací. V tuto chvíli není přesně známá dosažitelná složitost v případě použití tohoto typu útoku, nejoptimističtější odhad hovoří o složitosti 2^{87}. Postihnuty mohou být šifry Rijndael a Serpent a další. Útok spočívá v převedení šifry na soustavu kvadratických rovnic o více neznámých, nicméně útok nebyl plně prokázán. Nejnovější článek o bezpečnosti Rijndael z roku 2007 [31] neprokazuje reálnou možnost útoků.

Doposud nejsou veřejně známé žádné úspěšné útoky na šifru AES, využívající kryptoanalytické metody. Vzhledem k uvedeným skutečnostem lze AES doporučit jako vhodnou blokovou šifru už od délky 128 bitů.

4.2.6 RC2

RC2, také známá jako zkratka Ron's Code 2, navržena kryptologem Ronem Rivestem v roce 1987. Využívá klíč proměnné délky 1-128 bytů, šifruje 64-bitový blok. Sestává z 18 rund. Jedná se opět o druh šifry využívající Feistelovy sítě. Šifra je zajímavá střídáním tzv. mashing a mixing rund, při kterých provádí substituci a transpozici bloku s přičítáním klíče. Šifrování probíhá sekvencí kroků - inicializace, pak po sobě jdoucí rundy: 5 mixing 1 mashing, 6 mixing, 1 mashing a nakonec 5 mixing rund [32]. Další zajímavostí je využití desetinného rozvoje čísla π do tabulky k získání substitučních hodnot pro získání nelinearity. Zdrojový kód RC2 byl utajován jako obchodní tajemství a pronikl na veřejnost v roce 1996.

Útoky: Diferenciální kryptoanalýza objevila útok možný s pomocí tzv. related-key attack, který průchozí bity, které projdou přes většinu rund s vysokou pravděpodobností [33]. Zlomení klíče vyžaduje 2^{32} CPA. Algoritmus RC2 z hlediska bezpečnosti není perspektivní, nedoporučuji tedy šifru RC2 používat.

4.2.7 RC4

Šifra RC4 patří narozdíl od šifer RC2, 5 a 6 k nejznámějším a nejrozšířenějším proudovým šifrám. RC4 byla navržena Rivestem v roce 1987 ve společnosti RSA a byla obdobně jako šifra RC2 utajovaná. O několik let později byl RC4 kód anonymně zveřejněn na internetu [34]. Šifra našla široké využití v různých protokolech a zabezpečujících standartech, např. ochrana bezdrátových sítí WEP i WPA vychází ze šifry RC4 (WPA využívá narozdíl od nedostatečně bezpečného WEP dynamicky měnící se klíč).

Kód šifry vychází ze zjednodušené verze Vernamovy šifry, k šifrování se používá 256-bitový S-box [35], který se inicializuje klíčem šifry (proces Key-scheduling), pro který se používá 40 nebo 128-bitový klíč. S-box (dále také keystream) je inicializován pseudokódem takto (Klíma v této souvislosti hovoří o *míchání karet* [36]):

```
for i = 0 to 255  {
    S(i) = i;
}
j = 0;
for i = 0 to 255  {
    j = (j + S(i) + K(i)) mod 256;
    swap S(i) and S(j);
}
```

Schéma 4-2 – Míchání karet RC4

Výsledné šifrování textu spočívá v provedení operace XOR otevřeného textu a keystreamu s tím, že po dosažení konce keystreamu RC4 probíhá cyklicky zase od jeho začátku [35].

RC4 se časem stala zcela nepoužitelnou. Nejprve kryptolog Golic ukázal, že sekvence výstupních dat RC4 není náhodná (rozpoznali nenáhodnost ve vzorku 2^{40} bajtů). Fluhrer a McGrew zkoumali náhodnost dvou po sobě jdoucích bajtů a nenáhodnost zjistili už s 2^{30} bajty. V roce 2001 publikovali vědci Shamir, Mantin a Fluhrer útok na šifru RC4 spočívající ve slabých inicializačních vektorech. Našli vztah, podle kterého lze s určitou pravděpodobností odvodit z prvních bajtů další bajty keystreamu (analýza, která vedla k úspěšnému odvození klíče vyžadovala cca 2^{22} odchycených paketů). Andreas Klein objevil v roce 2006 útok, na jehož základě byl později představen nástroj, který běžně dostupnými prostředky zlomí 104-bitový klíč šifry RC4 v protokolu WEP během několika minut. V roce 2007 ukázali Eric Tews a kol. úpravu Kleinova útoku [37], ve které demonstrovali, že k odvození klíče s pravděpodobností 0.5 stačí pouze cca 40 000 zachycených paketů, s počtem zachycených 85000 paketů je pravděpodobnost už 0.95.

Šifru RC4 tedy nelze k praktickému použití vůbec doporučit.

4.2.8 RC5

Jedná se o rychlejší a kvalitnější šifru, než původní bloková RC2, kterou Ron Rivest představil v roce 1994. Velikost klíče je variabilní od 0 - 2 Kbit, velikost bloku pro 32, 64, 128 bitů. Proměnný je i počet rund, který může být od 1 do 255. Originální návrh šifry pracuje s 12 rundami a 128-bitovým klíčem. Úspěšné útoky byly doposud zveřejněny pouze na odlehčené varianty originálního návrhu. Nejlepší diferenciální kryptoanalýza šifry RC5 s 32 bitovým klíčem, která byla na šifru doposud publikovaná, vyžaduje 2^{28} CPA při 12 rundách. Zlomení 64-bitového klíče při 12 rundách vyžaduje 2^{44} CPA.

Kryptoanalytik Byriukov se domnívá, že u 16 rundové varianty s 64-bitovým klíčem je možné snížit časovou složitost až 2^{20} krát, metoda však nalezena nebyla. Postup lineární kryptoanalýzy [38] na 32-bitovou šifru s 5 rundami se pohybuje mezi 2^{32} až 2^{34} KPA, více KPA má za následek vyšší úspěšnost uhodnutí klíče. Stejní vědci neuvažují, že by zmíněné útoky měly mít dopad na šifru RC6. Z praktického hlediska byly podniknuty úspěšně útoky hrubou silou na 64 bitový klíč RC5 pomocí sítě počítačů.

RC5 lze v implementaci s 128-bitovým klíčem a zvýšeným počtem rund (18-20) jako bezpečnou šifru rozhodně doporučit.

4.2.9 RC6

Šifra RC6 je vylepšená verze původní RC5, zpracovávaná týmem v čele s původním návrhářem Ronem Rivestem. Oproti RC5 obsahuje druhou paralelní větev obdobného zpracování vstupního bloku dat. Šifra podporuje 128, 192 a 256 bitové klíče a obsahuje 20 rund, ale lze ji užít podobně obecně jako předchůdce. RC6 se dostala do finálového výběru soutěže AES. V tuto chvíli jsou k dispozici útoky na RC6 typu Key-recovery s 128-bitovým klíčem vyžadující paměťovou složitost 2^{74} a 2^{118} textů.

Návrh šifry RC6 lze doporučit stejně jako RC5 (šifru RC6 ale narozdíl od RC5 ve všech oficiálně podporovaných velikostech).

4.2.10 SERPENT

32-rundová šifra, která se ve finále soutěže AES umístila na 2. místě autorů Eli Bihama, Rosse Andersona a Larse Knudsena, byla navržena v roce 1998. Serpent má kořeny v šifře Square a stejně jako ostatní AES soutěžící šifruje 128-bitové bloky.

Pokud bych nepočítal dříve zmíněný XSL útok, Serpent nedisponuje žádnou známou vážnou chybou, která by měla za následek ohrožení její bezpečnosti. Útoky v teoretické rovině byly prozatím vedeny na omezené verze šifry – mezi nejlepší útoky na 11-rundovou verzi této šifry patří modifikovaná verze lineární kryptoanalýzy [39], vyžadující 2^{118} známých textů a 2^{214} přístupů do paměti.

4.2.11 SKIPJACK

Jedná se o americkou armádní 80-bitovou šifru původně určenou pro všechny úrovně utajení a pro implementaci ve speciálních šifrovacích kartách. Šifra byla odtajněna agenturou NSA v červnu roku 1998. Skipjack disponuje 32 rundovým mechanismem, je založen částečně na Feistelově síti, neobsahuje inicializační permutace a z klíče se negenerují subklíče, používají se tak, jak jsou uloženy. Hned v červnu 1998 objevil Biham mechanismus, který nalézá v šifře možná rizika [19]. Následně na konferenci CRYPTO'98 Shamir publikoval útok , který oslabuje 31 z 32 rund, využívající tzv. nemožnou diferenciální kryptoanalýzu (impossible differential cryptanalysis – dále také IDC). IDC [40] vychází z původní diferenciální kryptoanalýzy – na rozdíl od klasické diferenciální kryptoanalýzy se IDC snaží hledat a eliminovat nemožné (nebo velmi nepravděpodobné) události v procesu šifrování. Metoda má tak velkou pravděpodobnost, že by eliminované klíče nemohly zapříčinit, aby nemožnou událost způsobil správný klíč.

V roce 2002 byla publikována úspěšná kryptoanalýza všech 32 rund [9], při které ke zjištění klíče Skipjack postačí $2^{32.5}$ KPA a 2^{44} šifrovacích operací. Vzhledem k povaze útoků a velikosti klíče nedoporučuji šifru používat vůbec.

4.2.12 TWOFISH, BLOWFISH

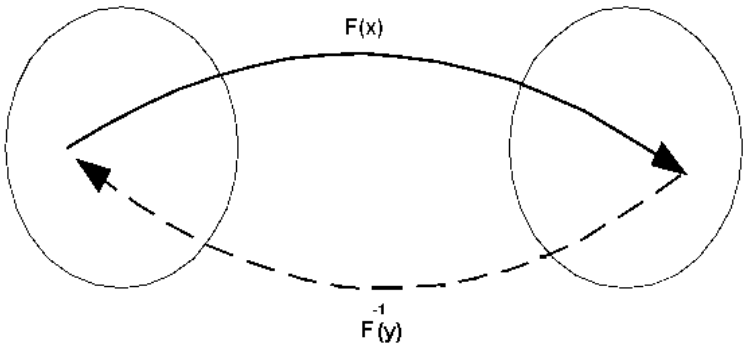
Blowfish je známá a populární 16-rundová šifra navržena v roce 1993 Brucem Schneierem. Konstrukce Blowfish vychází z klasické Feistelovy sítě [41] a substitučních a permutačních fází. Délka klíče je variabilní od 32 do 448 bitů. Výsledky kryptoanalýz odhalily dílčí úspěchy v rozlišení několika rund od pseudonáhodného textu a existenci slabých klíčů, které lze ověřit po získání 2^{34} známých textů [42].

Šifru Blowfish v roce 1998 autor spolu s jeho týmem doplnil o následníka - Twofish, který má stejný počet rund jako předchůdce a jehož velikosti klíče se ustálily na klasických 128, 192 a 256 bitů. V roce 2000 publikovali Moriai a Yin doposud nejúspěšnější kryptoanalýzu [43], postavenou na zobecnění diferenciální kryptoanalýzy, která na zlomení plných 16-rund šifry Twofish vyžaduje za určitých podmínek cca 2^{51} zvolených textů.

Obě šifry lze považovat vzhledem k teoretické úrovni útoků za dostatečně bezpečné.

4.3 Asymetrické šifrovací algoritmy

Asymetrická kryptografie je veřejnosti známá už 30 let. Jedná se o stabilní oblast kryptografických algoritmů, ve které se nové metody objevují velmi zřídka, na rozdíl od ostatních kategorií kryptografie. Objevení nového asymetrického algoritmu rovná se objevení složité jednocestné funkce s tzv. *padacími vrátky*. Padací vrátka zajišťují znalci tajemství (podle obr. 4-1 znalost *y*) - přístup k jednodušší metodě, otevírající rychlou zkratku, kterou se lze k zašifrovaným informacím dostat [3].



4-1 Jednocestná funkce s padacími vrátky

Stabilita této kategorie je zapříčiněna čistě matematickým charakterem problému, na němž daný asymetrický algoritmus stojí a jehož řešení by znamenal historický průlom nejenom

v kryptografii, ale také i v matematice. Principy, na kterých asymetrické algoritmy stojí, lze bez znalosti tajného klíče v současné době nejrychleji řešit pouze s exponenciální složitostí. V rámci diplomové práce bych rád rozepsal další známé asymetrické metody, které se běžně využívají v praxi.

4.3.1 Elgamal

Šifrovací algoritmus pro asymetrickou kryptografii Elgamal byl poprvé popsán v roce 1984 stejnojmenným kryptologem [4]. Princip spočívá v řešení úlohy tzv. diskrétního logaritmu a vychází z Diffie-Hellmanova (dále také D-H) schématu pro výměnu klíčů. Stejně jako RSA dokáže také Elgamal kromě digitálního podpisu provádět asymetrické šifrování. Co se týče praktických vlastností, je ale RSA více efektivní, protože zašifrovaný text šifrou Elgamal je 2x delší než text originální.

Blíže k samotnému schématu. Tvorba klíče spočívá v nalezení vhodných celých čísel y, g, x, p tak, že platí vztah $y = g^x \pmod{p}$ s tím, že čísla y, g, p tvoří veřejný klíč, $g<p$, p je prvočíslo a exponent x klíč tajný. Proces šifrování spočívá ve vytvoření 2 zpráv podle:

$$a = g^k \pmod{p} \qquad \text{(alg. 4-11)}$$

$$b = y^k M \pmod{p} \qquad \text{(alg. 4-12)}$$

kde k je náhodné tajné číslo zvolené odesílatelem

Pak jsou zprávy a,b zaslány příjemci, který původní zprávu M dešifruje s pomocí privátního klíče x takto:

$$M = a/b^x \pmod{p} \qquad \text{(alg. 4-13)}$$

Nejznámější kryptoanalýza Elgamal dokáže za určitých okolností falšovat originální podpis [44]. Metoda ale nepojednává o útoku, řešícím problém diskrétního logaritmu, který Elgamal využívá, ale za jistých okolností dokáže s určitou pravděpodobností provést útok na falšování podpisových zpráv. Bezpečnost asymetrické kryptografie jako takové v moderní oblasti shrneme spolu s RSA.

4.3.2 RSA

Princip asymetrické šifry RSA byl zevrubně popsán výše. Z bezpečnostního hlediska algoritmus RSA stojí na problému faktorizace, což je rozklad jednoho čísla na součin dvou prvočísel a na problém RSA, který spočívá v nalezení kořenu m v operaci m^e. Doposud nebyl nalezen algoritmus, nebo není veřejně znám, k efektivnímu řešení problémů, na kterých šifra stojí. Popíšeme si některé pokusy o útok, na kterých stojí hledání soukromého klíče.

Pokud by mohl útočník najít z n součinitele p a q, zjistil by Eulerovu funkci $\varphi(n) = (p - 1)(q - 1)$. Z té by dokázal spočíst soukromý klíč d, na základě vztahu $d = e^{-1} \pmod{\varphi(n)}$. Schopnost faktorizace a tedy hledání čísel p a q ze známého n, je ale omezená [4]. Nejznámější a nejefektivnější faktorizační metodou na klasických počítačích je GNFS (General number field sieve), která má ale stále exponenciální časovou složitost (viz kapitola 3.1) [45]. Za efektivní metodu je zde považován mechanismus, který dokáže najít p a n v polynomiálním čase.

Jedinou známou faktorizační metodou, která dokáže faktorizovat čísla v polynomiálním čase [46] je tzv. *Shorův algoritmus*, ovšem ten je implementovatelný pouze na kvantovém počítači, jehož praktickou realizaci k faktorizaci dnešních šifer svět ještě nezná. Jeho princip spočívá v převedení problému faktorizace na problém hledání periody v určité periodické funkci $F(a) = y^a \pmod{n}$, kde n je číslo, které chceme faktorizovat. Teoretické řešení úlohy pro výpočet diskrétního logaritmu a možnost využití u analýzy kryptografických algoritmů založené na bázi eliptických křivek uvádí Shor již spolu s problémem faktorizace [47]. Lze se tedy oprávněně domnívat, že s příchodem kvantových počítačů bude potřeba od moderní asymetrické kryptografie upustit.

Nicméně pro dnešní i budoucí účely asymetrické kryptografie lze RSA stejně jako schéma Elgamal s patřičnými délkami klíčů (4.5) bez problémů doporučit.

4.3.3 RSA a digitální podpis

Zmíněný algoritmus RSA lze využít také pro potřeby digitálního podpisu. Podle postupu z kapitoly 4.3.2 je vypočten podpis takto:

Odesílatel vypočte pomocí svých klíčů $s \equiv z^d \bmod n$. Následně odešle dvojici (s,z) příjemci, který verifikuje zprávu: $z \equiv s^e \bmod n$. Parametry jsou definovány jako: d je odesílatelův soukromý klíč (platí: $e*d=1 \bmod (p-1)(q-1)$), n a e jeho veřejný klíč a z zpráva - otisk.

Bezpečnost

Některé starší implementace podpisů RSA mohou být náchylné k Bleichenbacherovu útoku. Bleichenbacher na konferenci ukázal, že v případě použití schématu PKCS#1 verze 1.5 nebo ANS X9.31 a s exponentem o čísle 3 lze díky rozboru dojít k formálně platným, ale prakticky zfalšovaným podpisům.

4.3.4 Digital Signature Algorithm

Digital Signature Algorithm (dále také DSA), také známý jako Digital Signature Standart [48] – americký národní standart pro tvorbu digitálních podpisů. Nejnovější verze standartu FIPS-186, nyní už ve své třetí revizi, ve svém návrhu zohledňuje i nové hashovací funkce třídy SHA-2. Délky klíčů

podporuje formát DSS v první verzi 512 až 1024, resp. konstantních 1024 bitů ve druhé verzi standartu a 2048, 3072 bitů u navrhované 3. verze. DSA je zajímavé svou konstrukcí v podobnosti Elgamalově algoritmu, využívající pouze podgrupu prvočíselného řádu.

Rozšířením algoritmu DSA na podgrupu eliptických křivek, resp. záměnou podgrupy Z_p* za podgrupu eliptické křivky (nejčastěji náhodně vygenerované) získáme nové podpisové schéma ECDSA.

Z praktických výhod je vhodné poznamenat především nutnost menšího počtu bitů pro používané klíče, který je dostatečně bezpečný již ve stovkách bajtů (4.5) (pro srovnání klíče DSA i RSA jsou dnes používány nejčastěji řádově v tisících bajtů).

Algoritmus stojí na rovnosti malé Fermatovy věty a problému diskrétního logaritmu (viz alg. 4-14).

$$g = h^{(p-1)/q} \bmod p \Rightarrow g^q \equiv h^{p-1} \equiv 1 \pmod p \qquad \text{(alg. 4-14)}$$

U diskrétního logaritmu $a = b^c \bmod d$, pro a,b,c,d z oboru přirozených čísel pak diskrétním logaritmem nazýváme proměnnou c.

Generování klíče

První fází je volba délky klíče, dále výběr hashovací funkce a výběr takového čísla q, které má stejnou délku jako otisk hashovací funkce. Dále vygenerujeme prvočíslo p takové, aby platilo: $q \mid (p-1)$. Nalezneme generátor g grupy Z_p*, generované prvočíslem p řádu q. Nyní vygenerujeme náhodné číslo x, které bude privátním exponentem a tedy soukromým klíčem, kde $0 < x < q$. Posledním krokem je výpočet $y = g^x \bmod p$, pro vygenerování posledního veřejného parametru y. Veřejným klíčem je pak čtveřice (p, q, g, y), soukromý klíč se uvádí nejčastěji jako čtveřice (p, q, g, x).

Podpis zpráv

Provedeme výpočet r,s tak, aby platila rovnost:

$$r = (g^k \bmod p) \bmod q$$
$$s = (k^{-1}(Hash(m) + x*r)) \bmod q$$

Rovnice 3 Podpis zpráv DSA

kde m je zpráva, k je tajné číslo, pro které platí $0 < k < q$, a platí $kk^{-1} \equiv 1 \pmod q$. Podpisem zprávy je pak dvojice (r,s).

Verifikace podpisu

Pomocí veřejného klíče (p, q, g, y) ověříme platnost neznámého podpisu (r,s) takto:

$w = s^{-1} \bmod q$

$u_1 = ((Hash(M')w) \bmod q$

$u_2 = (r*w) \bmod q$

$v = ((g^{u_1} y^{u_2}) \bmod p) \bmod q.$

Rovnice 4 Verifikace podpisu DSA

kde M' je podepisovaná zpráva. Pokud platí, že $v = r$, podpis je považován za platný.

Bezpečnost

Podpisové algoritmy, opírající se o problém diskrétního logaritmu (schémata DSA, ECDSA) jsou z bezpečnostního hlediska považovány za bezpečné a jako podpisový standart jsou hojně využívány. Nejlepším řešením hledaných parametrů eliptických křivek lze dosáhnout za pomocí Pollardových metod [49], jejichž složitost se pohybuje v řádu $\sqrt{n}$, kde n je počet bodů křivky. V dnešních měřítkách je doporučováno použít velikost klíče aspoň 160 bitů, což odpovídá na základě tabulky doporučených velikostí klíčů cca 512 bitů u šifry RSA.

U ECDSA lze hovořit ještě také o Pohlig-Hellmanově redukci, která sice samotná neřeší problém diskrétního logaritmu, ale redukuje ho na několik problémů v grupě prvočísel. Takže se v praxi nejčastěji provádí P-H redukce a následně Pollardova ró-metoda. V současné době však není známa žádná metoda, která by byla schopna řešit úlohu diskrétního logaritmu v polynomiálním čase.

Z hardwarově-implementačního hlediska se v době psaní této kapitoly objevil i nový útok na ECDSA pomocí tzv. šablon [50], který využívá informace o zjištění rozdílů spotřeby zařízení, během výpočtu operací ECDSA, a jako obranu vyžadující zvýšení odolnosti proti útokům Differential Power Analysis.

4.3.5 Rabin

Matematický problém schématu Rabin byl obdobně jako RSA založen na problému celočíselné faktorizace [4] a tedy neschopnosti v praxi v rozumném čase rozložit číslo na součin 2 jeho prvočísel. Algoritmus Rabin je určitým speciálním případem RSA, jehož veřejný exponent e je roven 2.

Generování

1) Nejprve vygenerujeme prvočísla p a q, která jsou dostatečně veliká.

2) Výpočteme $n = p * q$.

3) Proměnnou n uvažujeme dále jako veřejný klíč a dvojici (p,q) jako soukromý klíč.

Podpis a verifikace

1) Podpis autor vypočte jako: $s = m^{1/2} \bmod n$.

2) Verifikace se provede výpočtem takto: $m = s^2 \bmod n$.

Bezpečnost

Tento velmi jednoduchý mechanismus má výhodu i slabinu v prokazatelnosti jeho bezpečnosti. Rabin byl prvním algoritmem, u kterého se podařilo prokázat, že jeho prolomení je stejně obtížné jako je obtížná úloha faktorizace. Svým způsobem je tedy teoreticky schéma Rabin více bezpečné než RSA, u kterého se toto předpokládá také, ale prokázat se to doposud nepodařilo. Rabin je také odolný vůči útoku pasivním útočníkem. Jeho slabina spočívá v možném útoku na vybrané zprávy (CPA). Útočník si vybere náhodné číslo m, patřící do Z_n* a vypočte $c = m^2 \bmod n$. Následně se pokusí dešifrovat zprávu c a získat tak y a s určitou pravděpodobností pak dokáže zjistit jedno z privátních prvočísel [4].

4.3.6 Výměna klíčů

Tato oblast aplikované kryptografie se snaží odpovědět na otázku: Jak dokážou všechny komunikující strany přenést všechny bezpečné údaje po nezabezpečeném kanále? Zmíněné metody řeší výměnu klíčů především algoritmy na bázi asymetrické kryptografie. Výsledek pak může být použit k účelu sdílených klíčů u symetrické kryptografie. Asymetrická kryptografie umožňuje bezpečnou výměnu klíčů.

4.3.7 Diffie-Hellman Key Agreement Method

První protokol pro výměnu klíčů byl navržen stejnojmennou dvojicí autorů již v roce 1976. Protokol Diffie-Hellman (dále také D–H) je založen na výše zmíněném problému řešení úlohy diskrétního logaritmu. D-H byl standardizován a patentován v roce 1980, jeho platnost vypršela v roce 1997.

Komunikační schéma:

Mějme komunikující strany A a B. Strany A i B se shodnou na parametrech u a p. Strana A vypočte zprávu $z = u^x \bmod p$, B analogicky $z' = u^y \bmod p$, kde x i y jsou privátní klíče daných stran. Obě strany si vymění zprávy z a z', A vypočítá klíč K jako $K = (u^x \bmod p)^y \bmod p$, strana B pak klíč $K' = (u^y \bmod p)^x \bmod p$. Z vlastností použité grupy pak plyne, že $K = (u^x)^y \bmod p$, $K' = (u^y)^x \bmod p$, a tedy, K = K'.

Výměna klíčů podle D-H schématu lze analogicky realizovat i s větším počtem zájemců o bezpečnou výměnu sdílených klíčů.

Bezpečnost

Toto neautentizované schéma je náchylné k útokům man-in-the-middle. Útočník může zůčastněným stranám předstírat identitu druhé strany, přijmout jejich zprávy a zároveň zaslat příjemcům zprávy své. Tímto získá útočník klíče všech zúčastněných. Problém je zřejmý – v autentizaci. Nedostatek řeší např. použití autentizovaného D-H protokolu - Station-to-Station - uvedeného až v roce 1992, případně využití odlišných autentizovaných schémat, nebo schémat využívajících podpis nezávislou a důvěryhodnou třetí stranou.

4.3.8 ECDH (Elliptic Curve Diffie-Hellman)

Určitou analogií ze vztahu mezi DSA a ECDSA můžeme odvodit vztah mezi protokolem Diffie-Hellman a ECDH (Elliptic Curve Diffie-Hellman). Komunikační schéma uvedeno ve schématu 4-3.

1) Strany A i B si vymění veřejné klíče – parametry eliptické křivky a bod na křivce.
2) Obě strany si vyberou velká náhodná čísla *a* a *b* (privátní klíče).
3) S využitím operace přičtení bodu *P* vypočte na křivce A i B *a*P* resp. *b*P*.
4) Výsledky zašle každý účastník druhé straně.
5) Každá strana roznásobí svým privátním koeficientem příchozí údaj, čímž získá analogicky se schématem D-H stejnou sdílenou hodnotu - klíč.

Schéma 4-3 Komunikace ECDH

4.4 Hashovací funkce

Ze všech ostatních kategorií kryptografických algoritmů nebylo během posledních let tak živo právě jako v oblasti hashovacích funkcí. Během posledních let se povedlo postoupit v kryptoanalýze mílovými kroky a úspěšně prolomit mnohé známé a často používané hashovací funkce. Vzhledem k faktu, že na jejich kvalitě stojí do značné míry dnešní bezpečnost mj. i v oblasti elektronického podpisu, lze se domnívat, že výběr dostatečně kvalitního a odolného algoritmu dobudoucna bude důležitý, protože bývá součástí legislativní úpravy, která jak známo nereaguje na vzniklé změny dostatečně rychle.

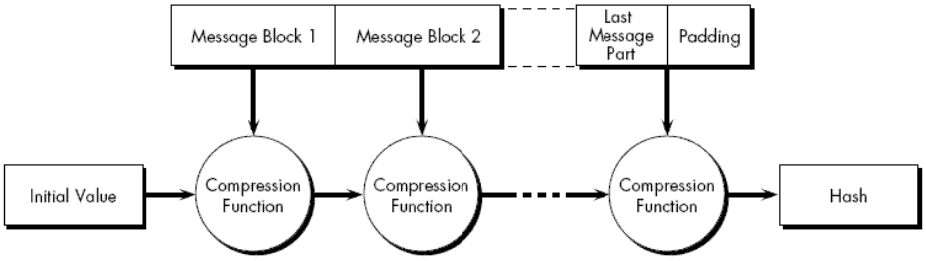
Rád bych ještě úvodem kapitoly vymezil pojem *blízké kolize* (near collisions), který definujeme pro libovolné vstupy *x, y* tak, že příslušné otisky hashovacích funkcí - *hash(x)* a zároveň *hash(y)* se liši pouze v malém počtu bitů [51]. Podobně odolnost proti blízkým kolizím rozšiřuje požadavek na odolnost hashovací funkce proti kolizím, kdy vymezuje nesmírnou výpočetní náročnost ke dvěma daným zprávám najít 2 patřičné otisky tak, že se výstupy liší pouze v několika málo bitech.

V kapitole 4.5 jsme provedli shrnutí doporučených odhadů výběru délky klíčů šifrovacích algoritmů, kategorie algoritmů a délky otisků výstupu hashovacích funkcí. Podle dostupných analýz

by měla dnešní nová generace hashovacích funkcí střednědobý časový horizont 20 let vydržet. Probereme si nyní přehled a vývoj některých hashovacích funkcí.

4.4.1 MD2

Funkce MD2 (Message Digest 2) patří ke starším hashovacím funkcím s délkou otisku 128 bitů, tvořenou 18 rundami. Funkce byla navržena již několikrát zmiňovaným Ronem Rivestem v roce 1989. Hashování zprávy probíhá ve třech rozdílných fázích. Zpráva se nejprve doplní na násobek 16 bajtů a provede se výpočet 16-bajtového kontrolního součtu, který je ke vstupní zprávě připojen. Zpráva se rozdělí na bloky o velikosti 16 bajtů. Následně je postupně počítána tzv. kompresní funkce z aktuální proměnné a vstupního 16-bajtového bloku s tím, že úvodní Initial Value je konstantní hodnota vstupující do první kompresní funkce. Výstup poslední kompresní funkce je samotný otisk celé zprávy [52]. Tomuto principu se říká Damgard-Merklova konstrukce.



Obrázek 4-2 Schéma hashovací funkce (zdroj [53])

Z tohoto schématu vycházejí všechny hashovací funkce z rodiny Message Digest a také funkce rodiny SHA. Co se týče samotné funkce MD2, jedná se z bezpečnostního hlediska již o zlomenou funkci – prof. Muller v roce 2004 uveřejnil zprávu, ve které popisuje útok na jednosměrnost MD2 funkce s časovou složitosti 2^{104} [52], načež následně o rok později Knudsen prezentoval [54] vylepšenou metodu útoku na jednosměrnost se složitostí 2^{95} a také ukázal několik nalezených kolizí kompresní funkce MD2. Ačkoli je MD2 návrhově starší a částečně odlišná v porovnání s novější MD4 a MD5, jedná se o relativně bezpečnější hašovací funkci.

4.4.2 MD4

Jedná se o následníka MD2 se stejnou délkou otisku se 3 rundami navrženého o rok později. Cílem původního návrhu MD4 byl rozumný poměr mezi rychlostí a bezpečností. Bohužel u MD4 byly nalezeny kolize ještě dříve než u jeho předchůdce. Každá runda je tvořena 16 kroky, které obsahují operace – rotace, bitový posun atp.

První kolize byly nalezeny Dobbertinovou metodou již v roce 1996 s pravděpodobností 2^{-22}, které se daly najít v řádu minut na tehdejších PC. Dobbertin ukázal, že 2 rundy z celkových 3 nejsou jednocestné. V roce 2004 prof. Wangová zveřejnila další útok [55], který relativně s velkou

pravděpodobností od 2^{-2} do 2^{-6} nalezení kolize nepřesáhne 256 operací. Nový kolizní útok [56] obsahuje 3 části: nalezení kolizní diference, podle které M způsobí kolizi, následně odvození z diference množiny patřičných podmínek a konečně vytvoření modifikace pro jakoukoliv zprávu, která nalezeným podmínkám odpovídá.

4.4.3 MD5

Poslední hashovací funkce z rodiny Message Digest stejného autora produkující opět 128-bitový otisk. Schéma činnosti MD5: Provede se rozdělení vstupní zprávy na 512 bitů, postupně se aplikuje kompresní funkce na tyto bloky vstupních zpráv a zároveň výstup z předchozích kompresních funkcí. MD5 dělí bloky na čtyři 32-bitová slova, na která postupně aplikuje 4 cykly, složené z 16 operací (celkem tedy 64 rund). Operace sestávají z nelineární funkce, rotací a modulárních součtů.

Cílem autora algoritmu bylo vytvořit opravu MD4, která byla znatelně méně bezpečná než samotná původní hashovací funkce MD2. V roce 2004 se podařilo čínskému týmu prof. Wangové [56] zveřejnit první kolizi funkce MD5 s výpočetní složitostí 2^{37}. Po zveřejnění metody útoku následně představili vylepšenou metodu český kryptolog dr. Klíma (2^{33} operací) [57] a poté japonský tým kryptologů dotáhl [58] hledání kolizí na výpočetní složitost 2^{30}. Hlavním problémem algoritmu MD5 je fakt, že byl a stále ještě je velmi populární a je neustále používán ve velkém množství aplikací, které mohou být v určitých případech ohroženy. Je vhodné poznamenat, že v roce 2006 dr. Klíma zveřejnil článek popisující tzv. tunelový útok a zdrojový kód programu, který zvládne najít kolizi MD5 na běžném počítači do jediné minuty [59].

Vzhledem k faktu, že u všech hashovacích funkcí z rodiny MD byly v rozumném čase nalezeny kolize [111], nedoporučuji žádnou z nich používat tam, kde je kladen důraz na bezkoliznost. Bude nejlepší tyto funkce nepoužívat vůbec a nahradit je novějšími.

4.4.4 RIPEMD

Hashovací funkce byla vytvořena v roce 1996, standartně produkuje 128-bitový resp. 160-bitový výstup v rozšířené verzi. RIPEMD byla navržena s cílem nahradit stávající rodinu hashovacích funkcí MDx. Bohužel tyto cíle nebyly naplněny. Schématicky se návrh funkce RIPEMD přibližuje návrhu MD4 s tím, že obsahuje, až na několik vnitřních konstant, dvě její paralelní kopie [55]. Dále obsahuje RIPEMD jinou kompresní funkci a má navíc jednu rundu.

Dnes je RIPEMD stejně jako rodina MD hodnocena jako zastaralá. Složitost úspěšného hledání kolizí se podle zveřejněných útoků týmu prof. Wangové [55] odhaduje na 2^{16} a nepřesáhne 2^{18} operací. Využívá se při ní diferenciální kryptoanalýza a samotná kryptoanalytická práce týmu prof. Wangové byla přímo odvozená z útoku na funkci MD4. Díky uvedeným informacím není doporučená pro budoucí použití.

4.4.5 SHA-0

SHA-0 je první návrh americké agentury NSA, standardizován organizací NIST. Funkce produkuje otisk dlouhý 160 bitů a koncepčně vychází z Rivestovy rodiny MDx. Rozšířen byl počet slov v kompresní funkci, kterých je celkem 5 a celkový počet rund je 80.

První úspěšný útok na odolnost proti kolizím byl prezentován na konferenci CRYPTO v roce 1998, který měl časovou složitost 2^{61}. Na plnou verzi hashovací funkce SHA-0 se podařilo provést útok v roce 2004 kryptologem Jouxem [60] v čase 3 týdny, útok sám potřebuje 2^{51} operací. V následném roce Wangová prezentovala útok 2^{39} a útok na near collision lze provést 2^{33} [51]. Není tedy vhodné doporučit hashovací funkci SHA-0 tam, kde se bere v potaz požadavek bezkolizního využití.

4.4.6 SHA-1

Hashovací funkce SHA-1, vytvořená vládní agenturou NSA je dnes spolu s ustupující MD5 používána nejčastěji. Ve svém návrhu využívá podobného principu co rodina funkcí MDx a je velmi podobná SHA-0.

Složitost běžného narozeninového útoku je kolem 2^{80} hashovacích operací SHA-1, což je pro dnešní výpočetní sílu stále vysoká laťka. Team prof. Wangové dokázal přijít s podobným postupem, jako v útoku na SHA-0 a ukázat oslabení funkce nejprve ukázkou na 2^{69} a poté na konferenci CRYPTO 2005 dokonce na 2^{63} [61] výpočtů funkcí SHA-1. Připomeňme, že distribuovaný útok obdobné časové náročnosti byl úspěšně proveden při útoku na 64 bitový klíč blokové šifry RC5 již před několika lety. Plná kolize k SHA-1 doposud není veřejně známa.

Akashi Satoh publikoval v roce 2005 návrh hardwaru, který by provedl úspěšný kolizní útok [57] na SHA-1 v rozumném čase. SHA-1 je z hlediska bezkoliznosti velmi ohrožena. Ohrožením však v tuto chvíli pro SHA-1 není myšleno bít na poplach, ale při nejbližší příležitosti plynule přejít na hashovací funkce novější generace.

4.4.7 SHA-2

Nejlépe dostupnou a v dnešních implementacích nejčastější používanou náhradou algoritmů starších rodin SHA a MDx se stává bezesporu SHA-2, publikovaná v roce 2001. Ve variantách podle délky otisku 256 (resp. 224) a 512-bitů (resp. 384) a se souhrnně označuje jako rodina SHA-2 [64]. Varianty 256 a 512 počítají s maximální zprávou o velikosti 2^{64}-1 resp 2^{128}-1 bitů a s velikostí zpracovaného bloku 512 resp. 1024 bitů. Počet rund je 64 resp. 80 rund. Kratší varianty SHA-224 a SHA-384 se vypočtou ořezem delší varianty jejich algoritmů. V každé rundě se provádí sada aritmetických a logických operací, celkové schéma vychází ze starších verzí SHA.

Pro žádné varianty rodiny SHA-2 nebyly doposud v žádných známých kryptoanalýzách nalezeny plné kolize varianty SHA-2. V souvislosti se zmíněnými útoky na hashovací funkce standartu MD5 [59] Klíma předpokládá, že vzhledem k obdobnému schématu třídy SHA-2 by mohlo být v budoucnu využito principu tunelování i pro útoky na hashovací standarty SHA-2.

SHA-2 není však úplně bezchybná. Nejnovější kryptoanalýza z roku 2007 [62] ukazuje 16 nových lokálních kolizí, oproti zatím 9-rundové kolizi [63]. Pravděpodobnost hledání kolizí je ale velmi malá, 2^{-66} u útoku, resp. s lepší pravděpodobností 2^{-39} [64].

4.4.8 SHA-3

V listopadu 2007 byla americkou agenturou NIST vyhlášena (podobně jako u AES) nová soutěž na hashovací standart, který ve Spojených státech amerických dobudoucna nahradí všechny předchozí šifrovací standarty. Jak bylo řečeno dříve, třída funkcí SHA-2 je v současné době považována za bezpečnou, přesto lze považovat vyhlášení soutěže na nový standart jako prozíravé řešení vzhledem k vývoji v oblasti hashovacích funkcí uplynulých let. Sběr žádostí na standart SHA-3 končí v listopadu 2008.

Na SHA-3 jsou kladeny nároky produkovat otisky dlouhé 224, 256, 384 a konečně 512 bitů.

4.4.9 TIGER

Hashovací funkce Tiger patří k méně známým, avšak z bezpečnostního hlediska k mnohem zajímavěji řešeným hashovacím funkcím, u kterých zatím nebyly publikovány žádné kolize. Tvůrci Eli Biham a Ross Anderson navrhli tuto funkci už v roce 1995 jako 24-rundovou s otiskem délky 192-bitů s úmyslem vytvořit náhradu pro SHA-1. Koncepce je odlišná od většiny zmíněných funkcí [65], které doposud většinou obsahovaly kompresní funkce. Tiger se tváří jako bloková šifra s 512-bitovým klíčem a stejně dlouhým blokem, z něhož vytváří klíče pro jednotlivé rundy.

Zavádí zvýhodnění pro 64-bitové archtektury a všechny aritmetické operace jsou počítány nad slovy modulo 2^{64}. K výpočtu se v rámci jednoho kroku Feistelovy šifry využívá tří 64-bitových větví, mezi kterými a na které jsou aplikovány aritmetické operace a rotace. Nejznámější útok na odolnost proti kolizím je pokus, který s úspěchem s 80% pravděpodobností nachází blízké kolize u celých 20 rund při 2^{48} operacích hashování. Autoři již však finalizují návrh nástupce - Tiger2.

Co se týče míry bezpečnosti, lze se domnívat, že jako náhrada pro starší šifrovací algoritmy postačí. Vzhledem k maximální délce výstupu hashovací funkce (192 bitů) ji z dlouhodobého hlediska nedoporučuji.

4.4.10 HMAC

Ze tříd hashovacích funkcí, po kterých je vyžadována bezkoliznost, zbývá k dnešnímu dni samostatně SHA-2. Případně je možné použít některých technik message autentizace – např. HMAC,

která přidává do výpočtu kromě existující hashovací funkce ještě další klíč, ze kterého produkuje výsledný otisk [66]. HMAC má narozdíl od běžných hashovacích funkcí možnosti využití také při autentizaci a prokázání znalosti sdíleného tajemství. Funkce HMAC je obecný algoritmus pro provádění klíčových autentizačních kódů a samotná hashovací funkce je funkcí HMAC volána externě a používá se spolu s dalšími hashovacími funkcemi, např. se MD5. Pak hovoříme o algoritmu HMAC-MD5, který provádí hashovací autentizační kód s pomocí hashovací funkce MD5. V tomto případě tzn. použití kolizní hashovací funkce v algoritmu HMAC je ještě hodnoceno jako bezpečné (útočník nemá odkud znát klíč, který do algoritmu HMAC vstupuje).

4.4.11 Návrhy na zlepšení hashovacích funkcí

Úplné oproštění od stávající koncepce Merkle-Damgard nebo podobných, ze kterých vycházejí dnešní typy šifer, nejspíš nebude tím hlavním zlepšujícím krokem, který zvýší bezpečnost hashovacích funkcí v brzké době. Hashovací funkce budou nejspíš i dobudoucna využívat stejného nebo podobného schématu jako je zmíněná koncepce, lišit by se mohl pouze charakter vnitřní kompresní funkce. Zde se domnívám, že by se mohly rozšířit zdokonalené a více bezpečné metody kompresní funkce algoritmu, případně celkové nahrazení iterativního principu kompresní funkce– což bude směr budoucího vývoje. Tento směr vývoje dokládá existence relativně nové hashovací funkce *Whirlpool*, publikované autory Vincentem Rijmenem a Paulem Barretem v roce 2003, u které lze vidět použití nového způsobu výpočtu vnitřní kompresní funkce – konkrétně Miyaguchi-Preneel koncepce [67], která nahrazuje původní Davies-Meyer koncepci, známou už z rodiny hashů MDx.

Davies-Meyer koncepce (kompresní funkce užitá u *MD5*):

$$H_i = BŠ(H_{i-1}) \oplus H_{i-1} \qquad \text{(alg. 4-15)}$$

kde H_i je výstup z i-tého bloku hashovací funkce, BŠ je bloková šifra, šifrující vstup klíčem i-tý blok zprávy H_{i-1}.

Miyaguchi-Preneel koncepce (kompresní funkce užitá u *Whirlpool*):

$$H_i = BŠ(M_i) \oplus H_{i-1} \oplus M_i \qquad \text{(alg. 4-16)}$$

kde BŠ je bloková šifra, šifrující výstup z $g(H_{i-1})$ – funkce g(x) provádí konverzi jejího vstupu na velikost klíče, H_i viz výše

Obě koncepce uvádějí pro každou H_0 rovnou IV, definovanou konkrétní hashovací funkcí.

Pokud bych tedy provedl shrnutí a doporučení z hlediska využití hashovacích algoritmů, ze všech úhlů bych doporučil funkci Whirlpool, vycházející z blokové šifry AES anebo hashovací funkce z rodiny SHA-2, která je sice oproti Whirlpoolu generačně starší, nicméně u ní nebyly a očekává se, že i v dohledné době nebudou nalezeny kolize. Whirlpool produkuje 512-bitový výstup nad maximální délkou zprávy 2^{256}. Obě třídy hashovacích funkcí poskytují kvalitní bezkolizní výsledky, pro dnešní softwarové implementace bych doporučil s ohledem na dostupnou podporu a rychlost [68] spíše SHA-2.

Pro vytváření autentizovaných hashů zpráv lze stále doporučit autentizační schémata z rodiny MAC, které jsou i s použitím starších a zlomených funkcí z hlediska kolizí - MD5 či SHA-0 - stále považovány za bezpečné (zde figuruje i autentizační klíč uživatele, oslabená vlastnost bezkoliznosti nemá na mechanismus vliv).

4.5 Doporučené délky klíčů

Délka klíče je velmi zavádějící pojem. Hlavní důležitost by měla být věnována v první řadě především návrhu algoritmu a až pak lze uvažovat délku jeho klíče jako další bezpečnostní aspekt. V případě nekvalitní šifry může být jakkoli dlouhý klíč zbytečný, pokud je špatně navržena. Ale na druhou stranu, pokud bude šifra kvalitní a bude pracovat pouze s klíči omezené délky, může je útočník získat pomocí útoku hrubou silou. Do následující úvahy budu proto počítat s obecně uznávanými šiframi, které se budou opírat o odhady různých světových autorit.

Aplikace kryptografických algoritmů vede boj s aktuální rychlostí používané výpočetní techniky. Ve snaze zajistit uživateli vhodný kompromis mezi rychlostí a bezpečností se kryptografie pokouší nejvíce přiblížit jejím ideálům. V roce 2000 byla zveřejněna studie prof. Lenstra [69], ve které provedla odhady reálné zlomitelnosti dostupnými metodami a možnostmi vývoje techniky do následujících let. Počítá se, že stávající algoritmy a možnosti kryptografie budou dostatečně bezpečné i pro následujících 20 let (viz také tabulka 4-1).

Autorita (rok vydání odhadu)	**Symetrické šifry**(v bitech)	**Asymetrická šifra RSA** (v bitech)	**Hashovací funkce (pro dig. podpis)** (v bitech)
Lenstra (2004) zdroj [69]	74 / 87	1077 / 1958	147 / 174
NIST(2007) zdroj [70]	80 / 112	1024 / 2048	160 SHA1 / 224 SHA-2
ECRYPT(2007)zdroj [71]	80 / 128	1248 / 3248	160 / 256
DCSSI (2007)	80 / 100	1536 / 4096	160 / 256

Tabulka 4-1 Přehled doporučených kryptografických algoritmů (hodnoty v buňkách pro rok 2008 / 2028)

Osobně se domnívám, s příhlédnutím k nedávnému vývoji kolem hashovacích funkcí, že ve střednědobém horizontu nebude šifra SHA-256 pro bezkolizní prostředí stačit. Nezávisle na některých odbornících ([72] a [69]) doporučuji již dnes čím jak nejdříve hashovací funkce celkově nahradit třídou SHA-2 a pokud se bude uvažovat o legislativní úpravě a bude to jenom trochu možné rovnou funkcí SHA-512. Více v kapitole 4.4.

4.6 Generátory pseudonáhodných čísel

Jak jsme si řekli v úvodní kapitole, generátory pseudonáhodných čísel (dále také PRNG) jsou deterministické funkce, které produkují výstup se statisticky obdobnými vlastnostmi, jako mají reálné náhodné generátory (dále také TRNG). K tomu, abychom přinutili PRNG funkci vytvářet požadovaná pseudonáhodná data, musíme do systému generování zavést určitý prvek nedeterminismu [4]. Často se jako zdroj nedeterminismu využívá určitý hardwarový vstup – např. šumová dioda, psaní na klávesnici, pohyb myší nebo systémový čas. Nedeterministický údaj pak vstoupí do PRNG funkce ve stádiu inicializace a na jejím základě funkce generuje posloupnost pseudonáhodných čísel.

V podkapitole 1.1.3 jsem položil otázku, jak souvisí kvalita PRNG s kryptografii. Nyní již mohu blíže tuto souvislost vysvětlit. Mějme obecnou šifru, která pracuje s klíčem 64 bitů dlouhým. Počet možných kombinací šifer je tedy 2^{64}. Vygenerujeme-li klíč nekvalitním generátorem, který z požadovaného rozsahu bude generovat pouze omezenou třídu např 2^{16} klíčů, útočníkovi značně ulehčí práci a dokáže velmi rychle zjistit klíč šifry s pomocí útoku hrubou silou (viz dále podkap. 4.1.3). Složitost výpočtu klesne z původních 2^{64} možností na pouhých 2^{16} i přes původně kvalitní návrh samotného šifrovacího algoritmu. Útočník pak bude vědět, že klíč díky špatnému vygenerování padne do třídy klíčů generovaných nekvalitním pseudonáhodným generátorem.

PRNG lze ještě dále rozdělit na další třídy, pro naše účely bude nejzajímavější třída kryptograficky bezpečných PRNG (dále také CSPRNG). Tato třída musí kromě požadavku dostatečně kvalitních náhodných dat splňovat také požadavky na rezistenci proti kryptoanalýze a proti zjištění stavu, ve kterém se generátor nachází. Na základě zjištěného stavu pak může útočník zcela zrekonstruovat produkovaná data dobudoucna i zpětně, což má v kryptografii katastrofální důsledky.

PRNG se bude také zabývat norma PKCS #14 (Public Key Cryptography Standards společnosti RSA, v praxi se jedná o velmi používané standarty), která je už delší dobu ve stádiu návrhu. V poslední době začíná být zajímavá tendence k využívání šifrovacích a hashovacích algoritmů k tvorbě pseudonáhodných dat – kvalitní kryptografické algoritmy produkují také kvalitní rovnoměrné rozložení, které lze považovat za náhodná data.

Útoky na PRNG pak mohou probíhat v několika rovinách [73]. Útoky mohou být vedeny přímo – útok spočívá v rozlišení výstupu PRNG od náhodných dat, dále útočit na vstup (rozlišení vstupních dat od náhodných, případně možnost ovlivnit vstupní data) anebo na tajný stav – v něm

zjistíme aktuální stav generátoru, čímž lze provést dopřednou i zpětnou rekonstrukci jeho dat. Pojďme si představit některé typy PRNG.

4.6.1 Blum Blum Shub

Algoritmus Blum Blum Shub (dále také BBS) vytvořili v roce 1986 manželé Blumovi a Michael Shub. Z uvedených PRNG se jedná o jediný kryptograficky bezpečný algoritmus [74]. Princip generování pseudonáhodných čísel spočívá v následujícím vzorci:

$$x_{i+1} = (x_i)^2 \bmod M \qquad \text{(alg. 4-17)}$$

M je pak definován jako součin dvou prvočísel (tzv. Blumova prvočísla). Z intervalu přirozených čísel menších než M je vygenerována inicializační hodnota x_0. Podle vzorce je pak vygenerováno další pseudonáhodné číslo. Problém pro případné útočníky spočívá v problému tzv. kvadratického rezidua. Pokud existuje x takové, že platí:

$$x^2 \equiv a \bmod M \qquad \text{(alg. 4-18)}$$

nazýváme všechna x jako a kvadratickým reziduem modulo M. Není-li znám rozklad čísla M na prvočísla, je obtížné rozhodnout, zda dané a náleží do množiny kvadratických reziduí modulo M. Algoritmus BBS je díky složitosti tohoto problému kryptograficky bezpečný a vhodný pro kryptografické použití, bohužel jeho rychlost je značně omezená, a tak není vhodný pro jiné použití.

4.6.2 Fortuna

Fortunu navrhli kryptologové Niels Fergusson a Bruce Schneier [75] jako následníka šifry Yarrow. Stejně jako předchůdce, tak i následník je považován za kryptograficky bezpečný generátor. Generátory *Yarrow* byly využity v linuxovém jádře (nově také *Fortuna*), jádře Mac OS a FreeBSD - pro logický soubor */dev/random*. . Základní design sestává z generátoru, akumulátoru a mechanismu uložení entropie (tzv. *seed file*), díky kterému pak v případě restartu nečeká na její nový sběr.

Akumulátor Fortuny využívá hashovací funkci SHA-256. Účelem akumulátoru je využití několika různých zdrojů entropie (Yarrow má 2 zdroje, Fortuna 32 zdojů). Akumulátor se sám znovu nastavuje buďto periodicky nebo po nasbírání dostatečného množství entropie (tzv. operace *reseed*). Záznam *seed file* obsahuje 64 bajtů náhodných dat. Systém po nastartování přečte náhodná data, provede *reseed* a znovu uloží aktuální hodnotu v *seed file*.

4.6.3 Lineární kongruentní generátor

Lineární kongruentní generátory (dále také LCG) jsou nejjednodušší, ale bohužel také nejméně kvalitní generátory pseudonáhodných čísel. Perioda generátoru je nejvýše rovna hodnotě *m*. Definice LCG:

$$x_{i+1} = (a \cdot x_i + b) \bmod m \qquad \text{(alg. 4-19)}$$

kde a,b,m jsou kladné, celé konstanty a x_0 je inicializační konstanta generátoru

Je zřejmé, že výstup generátoru produkuje hodnoty od 0 do *m* [76]. Nevýhodou generátoru je produkce velmi nekvalitního rovnoměrného rozložení, ze kterého je zřejmé, že jde o algoritmus lineární. Pro běžné účely postačí, ale ke kryptografickým účelům nelze v žádném případě algoritmus doporučit.

4.6.4 Mersenne twister

Relativně moderní (první návrh v roce 1997), a velmi kvalitní PRNG [4], navržený japonským týmem Makotem Matsumotem a Takuji Nishimurou, pojmenovaný po typu prvočísla (tzv. Mersennovo prvočíslo). Mersennovo prvočíslo je také velikost periody tohoto algoritmu (obecně takové prvočíslo *x*, které splňuje podmínku $x = 2^n - 1$).

Nespornou výhodou algoritmu Mersenne twister (dále také MT) je kromě kvalitního pseudonáhodného výstupu také rychlost její produkce. Přes všechny vyjmenované výhody tvůrci nedoporučují algoritmus využívat ke kryptografickým účelům. Důvodem pro tento závěr je fakt, že je MT postaven na lineární rekurzi, jejíž vlastností je po dostatečně dlouhém sledování vstupních dat možnost předvídání výstupu. Pro vhodné použití s kryptografickými algoritmy by bylo potřeba s pseudonodnými hodnotami provést ještě některé dodatečné operace – např. provést hashování výsledků MT. Na druhou stranu by to celý proces značně zpomalilo a tím by byla výhoda algoritmu zcela potlačena.

5 Kryptografie v praxi

V uplnynulých kapitolách byla nastíněna kryptografie především z teoretického a kryptoanalytického hlediska. Jak bylo popsáno dříve – útoky na samotné kryptografické algoritmy jsou na většinu popsaných algoritmů z dnešního praktického pohledu téměř nereálné nebo velmi obtížné až prakticky neproveditelné.

Samotné algoritmy však pro praxi nestačí. Kryptografický algoritmus je pouhá malá a nejčastěji modulová část celého bezpečnostního systému. Přidáváním dalších logických vrstev, kterými je modelový kryptografický svět přibližován do praxe, však většinou klesá bezpečnost celého systému. V moderních mechanismech jsme pak mohli být svědky špatných návrhů bezpečnostních protokolů, chybné implementace nebo nezajištění dostatečně bezpečného uložení šifrovacích klíčů. Chyby pak umožňují útoky pomocí postranních kanálů, oslabují původní návrh nebo umožňují obejít zabezpečení systému zcela jiným způsobem.

Příkladem takových systémů jsou např. autentizační protokoly, VPN servery, užívající protokol IPSec, systémy PKI nebo známé protokoly SSL. Na některé systémy se v této práci zaměřím.

5.1 Elektronický podpis, certifikáty

Elektronický podpis (v terminologickém smyslu tzv. *zaručeného elektronického podpisu*) jsou identifikační údaje autora v takové formě, která zaručuje bezpečnostní cíle – autenticitu, integritu i nepopíratelnost podepsané osoby [22]. Zaručení se děje zpravidla kryptografickými metodami, založenými na hashovacích funkcích a asymetrické kryptografii podle daného standartu. Nepopíratelností je zde myšlena možnost prokázat, že autorem zprávy je ten, kdo zprávu dříve podepsal, zatímco integritou prokazujeme nezměnitelnost zprávy – tedy útočník nemůže změnit zároveň zprávu i její podpis tak, aby byla prokazatelná nezměnitelnost zprávy a zároveň aniž by si toho uživatel nevšimnul. Pozorný čtenář zde jistě vidí spojitost s dříve zmíněným bezpečnostním cílem hashovacích funkcí - slabá odolnost proti kolizím. Často se ještě v elektronické komunikaci užívá také časové razítko (tzv. *timestamp*), které uvádí datum a čas vytvoření podpisu zprávy.

Zaručený elektronický podpis z hlediska zákona standartně vyžaduje kromě zmíněných certifikaci veřeného klíče a tedy povinné vystavení kvalifikovaného certifikátu certifikační autority [77].

5.2 Elektronický podpis

Motivací pro zavedení elektronického podpisu je bezesporu neustále se rozrůstající svět elektronického obchodu, internetu a potřeby rychlého formálního a úředního styku. Z technického hlediska je proces elektronického podpisu řešen provedením hashování autorovy zprávy a zašifrováním jejího otisku asymetrickým privátním klíčem autora, jejíchž výsledek je připojen ke zprávě.

Příjemce zprávy pak provede veřejným klíčem odesílatele dešifrování otisku a rovněž hashování zasílané zprávy a porovnání jejího otisku s dešifrovanou hodnotou. Pokud se výsledné otisky shodují a veřejný klíč je certifikován certifikační autoritou (viz také kapitola 7.2), je zaručena jednak integrita a také důvěryhodný certifikát (obsahující potvrzení, že veřejný klíč skutečně patří jeho majiteli) příjemci zaručí i nepopíratelnost zprávy.

5.3 Podpisové algoritmy

Jak bylo již dříve zmíněno, charakter rychlosti i bezpečnosti algoritmů pro digitální podpis u podpisových algoritmů závisí kromě hlavní metody především na charakteru hashovací funkce. Samotná rychlost asymetrických algoritmů, která je řádově mnohem menší než u algoritmů, postavených na symetrické kryptografii, pak umožňuje využít široce i u delších zpráv - tedy vzhledem ke konstantní délce výstupu hashovací funkce, řádově ve stovkách bitů, je lze v rozumném čase praktikovat pro různě dlouhé zprávy bez většího zdržení.

Níže zmíněné standarty se však stejně jako většina dalších algoritmů rozšířily do různých implementací, především pak v oblasti šifrování elektronické pošty jako PGP, OpenPGP i GnuPG. Podpisy však nejsou doménou pouze šifrování e-mailových zpráv. Dig. podpisy se využívají v mnohem širší míře. Například zajímavým je v tomto ohledu systém podepisování ovladačů WHQL a certifikace původu souborů autoritou v prostředí operačních systémů Windows.

Z obecného hlediska je útokem na digitální podpis především myšlen útok na vytvoření kolizního podpisu. Jako další možný způsob útoku je potřeba zmínit útok samotného autora podepsaných zpráv na nepopíratelnost [78].

6 Autentizační protokoly

K popisu dalších bezpečnostních mechanismů budu využívat pojmy – *identifikace* a *autentizace*, které blíže definuji. Identifikace je jednoduše tvrzení určité entity o své totožnosti – tedy např. tvrzení uživatele o tom, že je ten, za kterého se vydává. Autentizace je pak proces ověření identity toho, kdo se za něj vydává. V praxi se v elektronické komunikaci děje často prokázáním určité znalosti – např. heslo, PIN, vlastnictví – čipová karta, šifrovací klíč. V poslední době je pak často využíváno k moderní identifikaci biometrie, případně kombinací výše zmíněných metod. V souvislosti s využitím moderních kryptografických metod se často hovoří o tzv. *silné autentizaci*.

V rámci prakticky využívané a často populární terminologie se často vyskytuje ještě pojem autentifikace, který vznikl zažitým a nesprávným překladem z cizího jazyka a je často zaměňován za pojem autentizace, nicméně tento pojem je považován za nesprávný.

6.1 Obecné autentizační mechanismy

Možnosti autentizace, jak již bylo zmíněno, jsou velmi široké. Popisem nejčastějších metod uvedu čtenáře do problematiky prakticky využívaných mechanismů procesu ověření identity uživatele. Asymetrická kryptografie nabízí k autentizaci velice mocné a matematicky velmi odolné možnosti jak autentizovat. Nejdůležitějším otazníkem je v asymetrické kryptografii možnost bezpečného uložení privátního klíče. Doposud jsme se však blíže nezabývali způsobem uložení klíče takovým způsobem, aby bylo jeho použití dostatečně bezpečné.

6.1.1 Klasická hesla

Nejčastěji používaná metoda autentizace. Bohužel je to v praxi také nejméně spolehlivá a velmi zranitelná metoda, pokud není mechanismus v protokolu řádně implementován a také zajištěna jejich dostatečná složitost. Všechny útoky hrubou silou - slovníkové útoky nebo vyzkoušení všech možných hesel mohou úspěšně po čase prolomit. Útoky na databáze hesel na straně serveru jsou dnes také velmi populární – heslo se pak na straně serveru ukládá nejčastěji po průchodu hashovacích funkcí. I tato metoda však umožňuje slovníkový útok, při kterém se místo databáze hesel využívá porovnávání s položkami databáze otisků. Účinnému zamezení proti tomuto typu útoků však pomůže tzv. solení - generování náhodného řetězce, který odpovídá do páru hesla. Až po spojení s heslem je vypočítán otisk – čímž je zaručeno, že i stejné heslo různých entit má s největší pravděpodobností jiný otisk.

Protokoly a autentizační systémy ze strany serverové části pak nabízí ochranu proti útočníkům prostřednictvím omezení počtu připojení za vymezený čas, omezení celkového počtu zadání přístupových hesel, případně u autentizace ve webovém prostředí nutnost tzv. ověřovacích kódů, kdy musí uživatel ručně zadat obrázek, obsahující řetězec. Toto je pak pro stroj, útočící hrubou silou, obtížné.

6.1.2 Jednorázová hesla

Především v bankovnictví je populární metoda zasílání jednorázového hesla pomocí GSM sítě na mobilní telefony klientů. Často se takto jednorázovým heslem, ale i znalostí PINu, případně jiné kombinace autentizuje vlastník bankovního účtu. Další možnou autentizační možností je využití autentizačních seznamů či karet. Klient je dotázán na určitou položku v seznamu, kterou v rámci mechanismu klient druhé straně zadá. Ztráta tohoto seznamu je však velkým rizikem, proto je tato kombinace často doplněna o nutnost zadání klasického hesla. Odpadá tak možnost využití útoku odposlechu hesel. Tento systém se je již velmi dlouho používán a patří prakticky k nejlepším autentizačním metodám vůbec.

6.1.3 Autentizační kalkulátor

Autentizační kalkulátory jsou specializované pomůcky, zpravidla hardwarového charakteru, které umožní klientovi získání informace pro autentizaci. Princip spočívá v autentizaci kalkulátoru (nejčastěji zadáním PINu), který následně vypočítá jednorázové heslo. Klient pak heslo zadá k odeslání a serveru se autentizuje. I tento mechanismus je často doplněn nutností zadání další znalosti – např. PINu nebo hesla [79]. Autentizační kalkulátory mohou kromě autentizace klienta provádět také autorizaci dat, zadávaných uživatelem. Pak kalkulátor vypočítá otisk položek předávaných dat.

Kalkulátory z kryptografického hlediska využívají hašovacích funkcí pro výpočet kontrolních součtů a symetrických šifer, které jako klíč využívají vstupní řetězec, který je do kalkulátoru bezpečně umístěn a jehož znalost zná pouze server a kalkulátor.

Problematika kalkulátoru a hardwarových komponent naráží na tzv. tamper resistance – tedy odolnost samotného přístroje proti zneužití. Velikou výhodou kalkulátorů je pak nezávislost na bezpečnostním prostředí. Nevýhodou je pak možná ztráta samotného kalkulátoru uživatelem a nutnost zajištění zabezpečení k přístupu do kalkulátoru.

6.1.4 Jednorázové heslo S/KEY

S/KEY je autentizační algoritmus založen na tzv. *rekurentním* principu [80]. Stejně jako ostatní mechanismy - obě strany sdílí určité stejné tajemství – klíč. Tento jednoduchý a zajímavý

mechanismus avšak využívá několikanásobné aplikace hashovací funkce, čímž útočníkovi znemožní zpočítat předchůdce otisku.

Schéma činnosti:

1. Server i klient se dohodnou na sdíleném přirozeném čísle n.
2. Uživatel vyrobí $H^n(x)$ a zašle výsledek serveru, který si server uloží (typ hashovací funkce H je podle RFC-1760 MD4).
3. Při žádosti o autentizaci server pošle dotaz na $H^{n-1}(x)$.
4. Server provede porovnání přijaté hodnoty s vlastní vypočtenou hodnotou – $H(H^{n-1}(x))$
5. V případě shody je uživatel autentizován. Server uloží aktuální hodnotu přijatého hesla do databáze.

Schéma 6-1 Princip S/KEY

Proměnná n symbolizuje počet aplikací hashovací funkce na její předchozí výsledek. Příklad – $H^2(x) = H(H(x))$. $H^n(x)$ bude tedy odpovídat n-té úrovni hashování funkcí $H(x)$, kde x je vstupní řetězec, vstupující do funkce.

Při dalších autentizacích se server dotazuje na hodnoty nižších úrovní otisků($H^{n-2}(x)$, $H^{n-3}(x)$,…), které v autentizačním schématu fungují jako jednorázová hesla. Mechanismus umožňuje také autentizaci u více serverů, v tomto případě pak server odešle klientovi kromě výše uvedených údajů také náhodný řetězec (sůl), díky kterým se následně budou různé výzvy lišit.

Tento postup je však náchylný k Man-in-the-middle útoku. Nezbývá než se uchýlit k asymetrické kryptografii a využít jejího obalení k zabezpečení proti útoku.

6.1.5 Hardwarový klíč

Pro zajištění většiny zmíněných cílů se v moderních aplikacích čím dál častěji využívá jednoúčelových čipových karet, černých skříněk či jakýchkoli jiných zařízení, která zajišťují bezpečné uložení a případnou skartaci klíče v případě neoprávněného vniknutí. Např. čipová karta, která po celou dobu její životnosti zajišťuje, že privátní klíč neopustí hardware karty (této vlastnosti se říká tamper resistance). Hardwarové klíče obecně umožňují [79]:

1) generování dvojic klíčů,
2) generování žádostí o certifikát,
3) šifrování soukromým klíčem.

Bezpečnost těchto hardwarových tokenů je v podnicích a bankách klíčová. Jedním z nejznámějších zástupců karet je společnost RSA. Hardwarový autentizační klíč společnosti RSA (SecurID) obsahoval nedokonalou implementaci hashovací funkce SecurID Hash Function [81], která byla ukázána jako nedostatečně odolná proti kolizím, stejně tak jako bloková šifra. Zdrojové kódy funkcí byly utajovány jako obchodní tajemství, nicméně v roce 2000 byly anonymně zveřejněny na internetu (zjištění zdrojového kódu se povedlo pomocí reverzního inženýrství). Hlavní chybou karty je také špatná bloková šifra, jejíž zlomení podle autorů [81] vyžaduje nejvýše 2^{48} operací šifrování pomocí CPA útoků.

6.1.6 Biometrické systémy

Autentizace pomocí biometrických vlastností je duchem akčních a sci-fi filmů z 2. pol. 20. století a nejspíš běžnou realitou 21. století. I přes největší podíl běžných autentizačních metod, jakými jsou především hesla, biometrie proniká do cestovních médií jako rychlá a automatická autentizační metoda na základě určitých specifických vlastností člověka.

Pro zamýšlený pokus zastřešit biometrické autentizační metody slouží společný projekt několika světových firem - BioAPI, který od verze 2.0 podporuje také doplňující bezpečnostní mechanismy. Umožňují zabezpečit ucelené bloky biometrických dat z hlediska integrity i důvěrnosti [82]. BioAPI využívá autentizačních hashovacích funkcí (MAC nebo HMAC-SHA-1) pro zápatí datových záznamů a autentizaci datových bloků, dále datové bloky záznamu umožňují podpis (podporované podpisové algoritmy RSA, DSA, ECDSA) a k šifrování pak blokové šifry DES, 3DES a konečně AES.

6.2 Password Authentication Protocol

V úvodním popisu autentizačních protokolů by bylo vhodné začít podle klasického modelu ISO/OSI na co možná nejnižší logické komunikační vrstvě, hojně používané v Point to Point protokolu (PPP), který je dodnes klíčovým protokolem jeho linkové vstvy a základem komunikace připojení bodových typů – často stanic koncových klientů - uživatelů.

Základním a nejstarším autentizačním protokolem je Password Authentication Protocol (PAP) [83]. Používá tzv. dvoucestný handshake, při kterém je vygenerována klientská výzva typu *Authentication-request*, následována kladnou či zamítavou odpovědí serveru. Klient odesílá rámce, dokud mu server nezašle odpověď, případně skončí své pokusy po maximálním počtu opakování.

Žádné šifrovací algoritmy nejsou v tomto protokolu dostupné. Bezpečnost je tedy tímto velmi ohrožená. Útočník může vypozorovat komunikaci mezi stranami (tzv. útok *sniffing*), zjistit díky otevřené formě zasílaných informací veškeré údaje, případně pak podvrhnout komunikaci téměř

jakýmkoli způsobem. I přes bezpečnostní úskalí je tento velmi triviální autentizační prostředek dnes nepochopitelně stálé hojně používán. Pokud není uživatel donucen, je vždy vhodné použít jiný autentizační protokol.

6.3 Challenge Handshake Authentication Protocol

Novějším protokolem nahrazujícím PAP je protokol Challenge Handshake Authentication Protocol (dále také CHAP) [83], který již používá hashovací funkci (v RFC dokumentu podle [84] je nejčastěji uváděna funkce MD5). Popis činnosti: obě strany stejně jako v případě PAP sdílí stejné tajemství. Server jako iniciátor spojení zašle klientovi výzvu – *Challenge*, ta sestává z náhodné výzvy serveru. Zprávu klient hashuje a zašle zpět serveru. Server získanou zprávu porovná s hodnotou vlastního výpočtu a klienta v případě shody přijme, v případě rozdílných hodnot odmítne. Tento postup server vyžaduje po klientovi v náhodných časových intervalech. Výhodou je, že každá výzva by měla být unikátní.

Bezpečnost je promyšlenější než v předchozím protokolu – technikou náhodných výzev – Challenge je odražen útok proti znovupřehrání zpráv a odchycení hesla. Nevýhodou pak je nutnost uložení hesla v otevřené podobě (viz také [84]).

6.3.1 Microsoft Challenge-Handshake Authentication protocol

Vylepšenou variantou protokolu CHAP, vyvinutou společností Microsoft ve dvou verzích, Microsoft Challenge-Handshake Authentication protocol (dále také MS-CHAP) verze 1 z roku 1998 a opravená verze 2, používaná v implementaci protokolu PPTP. První verze má totožnou výzvu s původním CHAP a umožňuje změnu hesla. Druhá verze MS-CHAP nahrazuje změnu hesla jediným paketem.

Rád bych protokoly analyzoval blíže z hlediska samotného mechanismu autentizace. Podle MS-CHAP-1 klient nejprve vyžádá výzvu od serveru. Server pošle náhodnou 8-bajtovou výzvu. Klient hashovací funkcí LM Hash odvodí tři 8-bajtové hodnoty, které použije jako klíče pro šifru DES. Každý z těchto bloků je následně využit pro zašifrování výzvya zaslání serveru [85]. Server použije vlastní hashovací hodnoty klientových hesel k rozšifrování odpovědi. V případě shody pak server zašle paket s úspěšným potvrzením autentizace zpět klientovi.

MS-CHAP ve verzi 2 rozšiřuje mechanismus autentizace takto:

1. Klient nejprve server požádá o výzvu.
2. Server vytvoří náhodný 16-bajtová výzva.
3. Klient vygeneruje 16-bajtové náhodné číslo.

4. Dále klient vygeneruje 8-bajtový Výzvu hashováním serverové výzvy, klientova uživatelského jména a svého náhodného čísla.
5. Klient pošle serveru náhodné číslo z bodu 3 a 24-bajtovou odpověď z bodu 4.
6. Server dešifruje bloky příchozí zprávy a pokud se shodují, je klient autentizován. Dále server vypočte z 16-bajtového klientské výzvy 20-bajtovou serverovou autentizační zprávu.
7. Klient nakonec spočítá stejnou sekvenci, kterou server podle bodu 6 vypočítá, a v případě shody s došlou odpovědí autentizuje server.

Schéma 6-2 Průběh autentizace v protokolu MS-CHAP 2

Je zřejmé, že stejně jako server tak i klient provádí autentizaci druhé strany.

Z bezpečnostního hlediska stojí za zmínku kryptoanalýza implementačního protokolu Bruce Schneiera a kol. [85], ve kterém autoři shrnují bezpečnost vylepšených protokolů na úroveň úměrně zvolenému autentizačnímu heslu - tedy bez dodatečné kontroly složitosti hesla představující velké riziko. Pokud bude při zadávání hesla v rámci protokolu zajištěna jeho dodatečná kontrola, lze jej do jisté míry považovat za bezpečný.

6.4 Extensible Authentication Protocol

Extensible Authentication Protocol (dále také EAP) je rodina autentizačních protokolů rozšiřujících možnosti autentizace v PPP a podporujících vícero autentizačních mechanismů (RFC 2248, nově 3748). EAP vznikl de facto jako odpověď na dřívější méně bezpečné autentizační protokoly PAP i CHAP a stal se přímo samostatným protokolem pro autentizaci, použitelným i mimo PPP (není použit čistě ve fázi LCP jako dřívější protokoly, v této fázi se strany pouze dohodnou na použití protokolu EAP). Právě definice protokolu jako definice zpráv a komunikačního schématu dělají užití rozšiřitelné. EAP je hojně používán v různých variantách. Příklady podporovaných autentizačních způsobů jsou: MD5-challenge, jednorázové heslo či např. generický token [86].

MD5-challenge připomíná do značné míry protokol CHAP. Jednorázové heslo uživatel zadá ze svého speciálního slovníku na základě dotazu a konečně u generického tokenu klientská strana zašle vygenerované řetězce na základě přijatých řetězců ze serveru. Po ukončení autentizační fáze, která je opět řešena metodou výzva-odpověď, je druhé straně zasláno povolení.

K centrálnímu ověření protokol umožňuje využít také EAP serverů, či jiných pokročilejších metod – např. RSA karta, případně ověření přes LDAP.

Nejprve se strany dohodnou na metodě ověření, následně pak probíhá samotná autentizace. V autentizaci mohou obě komunikující strany vystupovat zároveň jako autentizátor (přístupový bod) i jako autentizovaný (klient). Komunikační schéma je popsáno ve schématu 6-3 (pozn. přístupový bod je zodpovědný za znovuzaslání ztracených paketů).

1. V žádosti přístupový bod vybere typ autentizace a zašle žádost autentizaci klientovi.
2. Klient zašle paket s odpovědí na žádost bodu, pakliže souhlasí
3. Přístupový bod zašle klientovi výzvu.
4. Klient zasílá přístupovému bodu odpověď na výzvu.
5. Přístupový bod potvrdí či zamítne klienta.

Schéma 6-3 Autentizace EAP (RFC 3748)

Různé varianty protokolu našly uplatnění i v mobilní komunikaci, autentizují se pomocí něj i klienti v GSM a UMTS sítích. Některé varianty protokolu budou popsány níže.

6.4.1 Lightweight Extensible Authentication Protocol

Proprietární odlehčená verze protokolu EAP, navržená a používaná firmou CISCO. Lightweight Extensible Authentication Protocol (dále také LEAP) vznikla původně jako modifikace protokolu MS-CHAP verze 2 (dále také MS-CHAPv2). Podpora je zabudovaná přímo v zařízeních firmy CISCO, stejně tak jako v některých jiných kompatibilních zařízeních, především v sítích WLAN.

V autentizačních mechanismech protokolu LEAP byly nalezeny zranitelnosti. Wrightovi se v roce 2003 podařilo vytvořit a později publikovat nástroj Asleap [87], který pomáhá slovníkovým útokem zjistit z odchycených paketů hesla. Pro doplnění v roce 2007 Wright nástroj upravil [87], aby byl schopen provádět podobné útoky také na MS-CHAPv2 a dokazuje tak tvrzení z výše zmíněné kryptoanalýzy, že je protokol bezpečný, stejně tak, jako zadávané heslo.

Z možnosti dnes velmi efektivních slovníkových útoků tedy nedoporučuji používat zařízení autentizující se pomocí protokolu LEAP a raději přejít na kvalitnější varianty např. PEAP, EAP-FAST.

6.4.2 Protected Extensible Authentication Protocol

Protected Extensible Authentication Protocol (dále také PEAP) je jedním z nejrozšířenějších variant protokolu EAP, která vznikla jako společné úsilí společností RSA, Microsoftu a Cisco jako protokol, orientující se na bezdrátové sítě. Existuje více variant PEAP. Především rozšířenější varianta PEAPv0 hojně podporovaná společnostmi Microsoft, Apple Computers a Cisco. Cisco pracovalo na prosazení vlastního standartu PEAPv1, nicméně také díky tomu, že Microsoft protokol PEAPv1 do svých produktů protokol nezačlenil, celková podpora tohoto protokolu je mnohem menší.

Protokol funguje jako rozšíření protokolu EAP, kdy vytváří tunel pro druhý ověřovací algoritmus EAP. Dalším rozšířením oproti původnímu protokolu EAP jsou chráněné hlavičky výstupu autentizace ve fázi zamítnutí nebo povolení spojení (viz také 5. bod schématu 6-3) [88].

Donedávna byl PEAP považován za bezpečný protokol. Poslední informace však ukazují, že některé implementace protokolu PEAP nejsou bezpečné [89] a umožňují útoky typu Man-in-the-middle - může být přeskočena fáze ověření u důvěryhodného serveru třetí strany. Tímto způsobem se opět omezuje bezpečnost mechanismu na úroveň zadaného hesla.

6.4.3 Další varianty

Nejširší autentizační podporu zajistil především protokol EAP-TLS [86]. Ten je hojně využíván u protokolu WPA, WEP a také v síťových technologiích Microsoftu. Protokol podporuje práci s certifikáty (viz také podkap. 7.1) a je dodnes považován za bezpečný. Doposud je veřejnosti známa pouze chyba v implementaci - konkrétně Cisco Secure ACS [90], která umožní přeskočit fázi autentizace. Vrstva TLS je sama o sobě bezpečná, jediným známým způsobem jak jej prolomit je získání privátního klíče jinou cestou. Systémy, používající protokol EAP-TLS tedy doporučuji používat např. čipové karty s vysokou mírou fyzické odolnosti, odkud je velmi problematické odpovídající privátní klíč získat. Samotný protokol TLS spolu bude podrobněji rozepsán v kapitole 8. Jako negativní vlastnost - protokolu je vyčítána velká režie během přenosu (opět díky protokolu TLS).

Další zmíněnou variantou protokolu je EAP-IKEv2 [91]. Protokol zajišťuje především vzájemnou autentizaci, založenou na Internet Key Exchange Protocol v.2 (dále také IKEv2). Autentizační schéma IKEv2 podporuje autentizaci pro každý směr odlišnou metodu (viz také tabulka 6-4).

Server	Klient
asymetrický klíčový pár	asymetrický klíčový pár
asymetrický klíčový pár	symetrický klíč
asymetrický klíčový pár	heslo
symetrický klíč	symetrický klíč

Tabulka 6-4 Režimy Autentizace EAP

Z bezpečnostního hlediska lze tento nový protokol (RFC z roku 2008) doporučit. Podporuje důvěrnost přenosu, vzájemnou autentizaci, ochranu integrity a ochranu proti slovníkovým útokům.

Posledními variantami, které bych vyjmenoval a již v rámci práce dále nerozváděl:

- EAP-OTP (dostupné v RFC 2289) - Autentizace je v protokolu řešena jednorázovými hesly. Podle kvality použitých lze hodnotit jako bezpečná.
- EAP-MD5 (dostupné v RFC 2284) – Autentizace je řešena podobně jako v protokolu Challenge Response Protocol výzvou a jednostranným ověřením klienta pomocí MD5.

- EAP-TTLS (zatím pouze draft v [92]) - Modifikace EAP-TLS, popsaného výše. Nepodporuje certifikáty a vytváří tunel pro vnitřní.

EAP je závěrem zajímavý rozšiřitelný autentizační koncept, který mohu jako základ doporučit. Vždy se vyplatí pečlivě vybrat správné rozšíření ve formě vhodného protokolu ideálního pro cílové prostředí.

6.5 Kerberos

Dlouhodobě ověřeným a mocným autentizačním protokolem modelu klient/server, zajišťujícím datovou integritu, postaveným na symetrické kryptografii, je právě Kerberos. Protokol pro autentzaci používají operační systémy Windows. Nyní existuje ve verzi 5, definovanou v několika RFC dokumentech (RFC 1510, následně v revizi RFC 4120). V širším kontextu hovoříme o Kerberu jako o celé Kerberos infrastruktuře. Historicky vznikl již v 80. letech 20. století v projektu Athena, která je spojeným využitím sil několika amerických institucí (např. MIT, IBM a DEC), ale verze 1-3 nebyly nikdy veřejně vydány. Aktuální revize z roku 2005 podporuje účinnější a bezpečnější metody autentizace [93]. Protokol využívá nezávislé třetí strany pro ověření a prevenci proti útokům typu Man-in-the-middle a jiných.

Kryptografické zázemí Kerbera bylo ve verzi 4 založeno na šifře DES ve standartním režimu, ve verzi 5 byla rozšířena na podporu šifry 3DES v CBC režimu, tedy dostatečně bezpečné kombinaci i dnes. Problémem návrhu protokolů Kerberos ve verzích 4 i 5 jsou známé implementační zranitelnosti, které často způsobují tzv. buffer overflow, jíž může útočník zneužít ke spuštění vlastního podvrženého kódu.

Návrh protokolu ve verzi 4 má několik potenciálních slabin [94]. Možné zneužití nabízí tzv. Replay attack, kdy útočník může zneužít autentizátor a délky časového razítka. Zneužitím charakteru protokolu TCP je možné za určítých podmínek tohoto využít ve stavu navazování TCP spojení.

6.5.1 Průběh autentizace

Klient nejprve zadá své přihlašovací jméno a heslo, ze kterého se vypočte otisk. Do hry vstupuje více entit – *server* – tj. server, který zprostředkuje autentizaci (v Kerberos terminologii AS), *tiket server* – server (TGS), který vydává tzv. *tikety* a *servisní server*, který poskytuje služby.

1. Klient zašle serveru zprávu s žádostí o autentizaci (v plaintextu a s identifikací daného uživatele)
2. Server zašle klientovi v případě úspěšného vyhledání v databázi následující zprávy:
    a. klíč sezení zašifrován privátním klíčem uživatele,

b. tzv. *tiket*, obsahující informace o uživateli, zašifrován soukromým klíčem Tiket serveru.

3. Klient první zprávu dešifruje a získá svůj klíč sezení, který použije pro komunikaci s Tiket serverem.
4. V případě, že chce klient využívat služeb serveru, zašle Tiket serveru zprávy:
    c. zprávu b) a identifikaci požadované služby,
    d. tzv. *autentizátor* – zpráva s časovou značkou, zašifrovanou jeho klíčem sezení.
5. Tiket server vybere tiket, který v rámci d) klient zaslal a dešifruje jej. Získaným klíčem dešifruje zprávu d) a klientovi následně zašle zprávy:
    e. Klient-Server Tiket, zašifrovaný privátním klíčem služby,
    f. Klient-Server klíč, zašifrovaný klientským klíčem sezení.

Pro vyžádání služby:

g. Klient se může autentizovat Servisnímu serveru zasláním zprávy e) a nové zprávy, sestávající z klientské identifikace a časové známky, zašifrované Klient-Server klíčem.

Schéma 6-5 Autentizační schéma systému Kerberos

Servisní server dešifruje svým soukromým klíčem tiket a novou zprávu z g). Dále pošle v případě úspěchu výslednou zprávu s novým časovým razítkem, opět zašifrovaným Klient-Server klíčem.

6.6 RADIUS, TACACS

RADIUS (akronymem z Remote Authentication Dial In User Service) je autentizační protokol, umožňující podobně jako Kerberos přístup k různým síťovým zdrojům. Je např. hojně využíván poskytovateli VoIP připojení pro autentizaci koncových stanic a také poskytovateli internetového připojení k přístupovému serveru např. prostřednictvím tzv. Dial-up připojení (s tím souvisí také možnosti účtování příchozích spojení a další možnosti protokolu). Autoři protokolu navrhli přenos informací řešit pomocí UDP datagramů – především z důvodu rychlé odezvy. Cílem protokolu je centralizovaná autentizace uživatelů, kteří přistupují přes přístupové servery. Nutno však poznamenat, že byl systém RADIUS navržen tak, aby autentizační informace nemusely být uloženy pouze na jednom serveru – RADIUS klient pak informace zjistí u jiného serveru, kde se nachází [79].

Přístupový bod – klient protokolu RADIUS se dotáže serveru, zda může uživatele přihlásit a za jakých podmínek. Typy autentizačních zpráv v prostředí RADIUS [95]:

Access-Challenge – RADIUS server zasílá požadavek klientovi na zadání jednorázového hesla .

Access-Request – přístupový bod – RADIUS klient zasílá žádost o autentizaci uživatele RADIUS serveru.

Access-Reject – RADIUS server zamítá žádost o přístup uživatele.

Access-Accepted – RADIUS server přijímá a povoluje přístup uživateli.

Konkrétní metody autentizace uživatele jsou v RADIUS podporovány několika autentizačními mechanismy – kombinace jména uživatele a hesla, dále protokoly PAP, CHAP i podpora jednorázových hesel. Komunikace pak nejčastěji probíhá jako dotaz/výzva klienta na uživatele a jeho odpovědi, které jsou pak pomocí zpráv přeneseny serveru. Uživatel může také pro komunikaci s klientem použít protokol PPP [79].

Podobné protokoly - TACACS, TACACS+ a XTACACS jsou varianty jednoduchého autentizačního síťového protokolu, který se rozšířil především v UNIXových prostředích a směrovačích, přístupových serverech (tzv. NAS). Protokol je navržený a podporovaný společností CISCO.

Následník protokolu RADIUS je DIAMETER [95]. Výhody nového protokolu bych shrnul na podporu transportního protokolu TCP a SCTP, dále také podporuje zabezpečený důvěrný přenos pomocí IPsec (podkap. 9.1) a TLS (viz také kapitola 8).

7 Public Key Infrastructure

Public Key Infrastructure (také hojně používaný akronym PKI nebo česky méně používaný překlad Infrastruktura veřejných klíčů) jako množina standartů a opatření, které jsou spojeny s kryptografickými klíči a především certifikáty - jejich správou, vydáváním a odvoláváním. PKI je dále popsán jako soubor různých standartů pro kryptografii s veřejnými klíči a specifikuje konkrétní parametry určené pro využití v komunikaci v počítačové síti. PKI vychází z norem X.500 – (níže zmíněné normy X.509 o certifikátech). Dále se PKI dotýká problematiky dalších souvisejících a navazujících norem jako S/MIME a protokolů typu SSL, TLS.

7.1 Certifikáty

Aplikací asymetrické kryptografie v kombinaci s certifikační autoritou získáváme tedy plně dostačující platformu pro základní bezpečnostní cíle – autentizaci, důvěrnost i nepopíratelnost. Z praktického a velmi hrubě popsaného hlediska je zaručený certifikát jakousi ochranou proti podvržení klíče a jeho důsledcích.

V širším pohledu je digitální certifikát struktura, která jednoznačně identifikuje toho, komu je vydána. Vždy je digitálně podepsána certifikační autoritou. Tato struktura je dána normou řady X.509, data jsou popsána v abstraktním jazyce ASN.1/DER. Norma X.509, jejíž první verze byla vydaná r. 1988, obsahuje základní informace. Verze 2 doplnila položky jednoznačné identifikace subjektu a vydavatele. Nicméně samotná verze 2 není v praxi moc využívaná. Novější verze 3 z roku 1996 obsahuje položky dle schématu 7-1 [96], v roce 2000 byl vydán draft aktuální verze certifikátů X.509 verze 4, která rozšiřuje revokační možnosti a přidává podporu pro verifikaci certifikačních řetězců [97].

Číslo verze (version)

Sériové číslo (serialNumber)

ID podpisového algoritmu (signature)

Vydavatel (issuer)

Platnost (validity)

Od (notBefore)

Do (notAfter)

Subjekt (subject)

Informace o veřejném klíči subjektu (subjectPublicKeyInfo)

Algoritmus veřejného klíče subjektu (algorithm)

Veřejný klíč subjektu (subjectPublicKey)

Jednoznačná identifikace vydavatele – od X.509 v. 2 (issuerUniqueID)

Jednoznačná identifikace subjektu – od X.509 v. 2 (subjectUniqueID)

Rozšíření certifikátu (extensions)

ID podpisového algoritmu certifikátu (signatureAlgorithm)

Digitální podpis certifikátu (signatureValue)

Schéma 7-1 X.509 verze 3 (zdroj [77])

Pro vytvoření certifikátu uživatel nejprve vygeneruje klíčový pár veřejného a soukromého klíče. Následně sestaví strukturu žádosti o certifikát, ve kterém vyplní patřičné údaje o svém jméně, svůj veřejný klíč a následně jej podepíše svým soukromým klíčem a zašle certifikační autoritě. Certifikační autorita pak na základě žádosti certifikát podle výše zmíněné struktury vydá.

Praktická komunikace mezi uživateli funguje tak, že subjekt X odešle uživateli Y svůj certifikát, který si ověří. Pokud je certifikát i podpis v pořádku, Y může použít veřejný klíč uživatele X, uvedený v jeho certifikátu. Komunikace pak dále probíhá standartně podle schématu asymetrické kryptografie.

7.1.1 Útoky na certifikát X.509

Dříve zmíněné útoky v sekci 4.4 byly zatím prováděny většinou na nepraktických příkladech. Nyní si ukážeme, že existence zmíněných kolizí poskytuje značné riziko pro bezpečnost elektronické komunikace. Pro ukázku uvádím útok na podvržení certifikátu X.509, který využívá hashovacího algoritmu MD5 [98]. Se znalostí originálního certifikátu lze podle [98] provést následující útok:

1. Vytvoření šablony pro RSA certifikát – vyplnění všech hodnot, až na veřejný klíč RSA, a podpisu.
2. Provedení MD5 funkce na první část podepisových dat, až do místa, kde začínají data RSA veřejného klíče.
3. S využitím principu útoku Wangové vytvoříme dva 1024-bitové řetězce b_1 a b_2, které vedou u funkce MD5 s IV z předchozího kroku ke kolizi.
4. Dále vytvoříme veřejné klíče RSA z původních řetězců b_1 a b_2 připojením nově nalezeného řetězce b.
5. Vložíme část veřejného klíče n_1 k certifikátu a spočítáme celkový MD5 otisk.

Schéma 7-2 Útok na certifikát X.509

Autoři této metody [98] uvádějí, že je hledání veřejných klíčů při požadovaných kolizích proveditelné v řádu několika dní. Ukázali jsme si, že tedy v případě nalezení kolizí je bezpečnost elektronické komunikace přímo ohrožována. Proto v rámci bezpečnosti vytvářených elektronických podpisů doporučuji přejít na kombinaci asymetrických šifer pouze a jen s doporučenými hashovacími funkcemi.

7.2 Certifikační autority

Stejně jako základní autority z hlediska občanskoprávního – např. stát, garantující a tvořící normy, lze si autoritu z pohledu certifikátů představit jako subjekt, který zajišťuje a garantuje faktické ověření identity a vydávající certifikát, který toto potvrzuje. Pochopitelně můžeme za autoritu z hlediska PKI považovat kterýkoli subjekt, který se za autoritu prohlásí a certifikát vystaví – proto legislativa Evropské unie, potažmo České Republiky pamatuje na tzv. *kvalifikované certifikáty*, jejichž použití má také právní důsledky a autorit, vydávající tento typ certifikátů je pouze omezené množství a zároveň tyto subjekty garantují vydání certifikátů pouze oprávněným držitelům.

Pokud tedy bude proces verifikace bezvadný, certifikační autority (dále také CA) by měly velmi pomoci praktické realizaci. Zpravidla nejsou CA samostatně vedené a tvoří určitou distribuovanou strukturu, která směruje od CA až k tzv. kořenové certifikační autoritě. Za důvěryhodný se nejčastěji považuje kořenový certifikát a cesta, kterou se certifikáty odkazují, by měla končit právě u něj. Dále se může stát, že jednotlivé CA mohou tvořit cyklický graf.

7.2.1 Revokace

Revokace certifikátů je zneplatnění certifikátu v případě, že by mohla existence certifikátů uvést nějakou stranu v omyl, neodpovídala realitě, případně mohla být zneužita ve prospěch útočníka.

Revokace se prakticky provádí vyplněním a zasláním žádosti certifikační autoritě, která spravuje tzv. CRL (Certificate revocation list).

Z obecného pohledu je vhodné revokaci provést, pokud došlo k jedné z následujících podmínek:

1. Kompromitace privátního klíče uživatele (byl ukraden, nebo se jej podařilo zlomit), v širším pohledu můžeme brát jako riziko ztrátu PINu, případně přístupového hesla ke klíči.
2. Dojde k nahrazení starého certifikátu novým – není tedy už potřeba využívat starší certifikát.
3. Změna faktických informací v certifikátu – zde by tedy neodpovídal certifikát realitě.
4. Revokace nadřazeného certifikáuů – v případě revokace nadřazeného certifikátu je objektivní důvod pro změnu certifikátu.
5. Změna účelu užití klíče – v tomto případě by platný certifikát neměl smysl.

Proces revokace z bezpečnostního hlediska může při takto nastaveném mechanismu znamenat určité riziko. Zejména pak v případě kompromitace klíčů útočníkem v případě zaručeného certifikátu, kde se jedná o legislativní proces, důležitost a klíčové rozhodnutí. Časová prodleva obnovení vnitřních záznamů a vydání nového revokačního záznamu znamená možný problém hledání viníka za případné zneužití revokovaného klíče. Riziko existuje mezi časem x, ve kterém došlo k zaslání uživatele a dále časem $x+t$, kdy došlo k vydání CRL certifikační autoritou. Je zde tedy vidět kritický čas mezi oficiální revokací a časem, kdy podal uživatel žádost. V čase x až $x+t$ může útočník uživatelský klíč zneužít, z pohledu třetí strany však legitimní uživatel svou povinnost revokovat kompromitovaný klíč včas splnil.

Částečnou obranou proti tomuto mechanismu může být využití klient/server protokolu OCSP (Online Certificate Status Protocol), který může CA poskytovat a jehož výsledky pak digitálně podepisuje. Služba zaznamenává on-line aktuální stav daného certifikátu, čímž oproti CRL (a tedy potenciálně procházení velkému počtu položek revokovaného seznamu) pracuje efektivněji. Zde je ještě vhodné poznamenat, že původní systém CRL byl navržen koncem 80.let 20. století, kdy ještě tyto problémy nebyly tolik aktuální, oproti tomu standart OCSP pochází z revize RFC 2560 z roku 1999.

8 SSL/TLS

SSL je již dlouholetý standart pro bezpečnou komunikaci a směle si dovoluji napsat, že se jedná o prakticky nejvíce používaný bezpečnostní protokol vůbec, který se dá v modelu ISO/OSI zařadit do relační vrstvy. První verze byla vydána společností Netscape už v roce 1994 a v podobě následníků je dodnes velmi rozšířena (resp. v jeho nejnovější verzi SSL 3.0 a jeho následníka TLS). V tomtéž roce byla vydaná opravná verze 2.0 a o dva roky později Netscape ve spolupráci s Paulem Kocherem vydali verzi 3.0 protokolu SSL a konečně v roce 1999 první standart [99], založený na SSL – TLS ve verzi 1.0 (RFC 2246). Z protokolu SSL 2.0 a 3.0 ještě vznikly odnože společností Microsoft - PCT resp. STLP, ve kterých zavádí některá drobná vylepšení. Nicméně těmito větvemi SSL se blíže zabývat nebudu.

Protokol ve své nejjednodušší podstatě zajišťuje obálku bezpečného přenosu informací z hlediska důvěrnosti, autentizace i integrity. Protokol SSL využívá k autentizaci možnosti asymetrické kryptografie, k důvěrnosti komunikace zase symetrické kryptografie podle vyjednaného nastavení. Komunikační schéma úvodní fáze – handshake je uvedeno ve schématu 8-1:

1. Klient zašle server zprávu *ClientHello,* ve které navrhuje SSL nastavení.
2. Server zašle klientovi zprávu *ServerHello*, v níž vybere SSL nastavení.
3. Server zašle také zprávu *Certificate*, obsahující certifikát s veřejným klíčem.
4. Server požádá klienta o jeho certifikát ve zprávě *CertificateRequest*.
5. Poslední zprávou *ServerHelloDone* potvrdí konec úvodní sekvence vyjednání serveru s klientem.
6. Nyní klient zašle certifikát ve zprávě *Certificate* serveru.
7. Klient zašle ve zprávě *ClientKeyExchange* klíč sezení (tzv. *session key*) zašifrován veřejným klíčem serveru.
8. Dále klient serveru potvrdí zprávou *CertificateVerify* svou autentizaci podpisem svým soukromým klíčem. Server verifikuje podpis dešifrováním veřejným klíčem klienta.
9. Klient serveru opětuje zprávu *ChangeCipherSpec*, která zajistí aktivování dohodnutých zabezpečení pro všechny klientské budoucí zprávy.
10. Dále klient serveru zašle zprávu *Finished*, aby server mohl zkontrolovat nově aktivované nastavení.
11. Server zašle klientovi zprávu *ChangeCipherSpec*, která obdobně jako klientská strana z 9. bodu zasílá všechny budoucí zprávy již v zašifrované podobě podle smluveného nastavení.
12. Poslední zprávou v rámci úvodního dialogu zasílá server klientovi zprávu *Finished*, kterou zkontroluje nastavení, obdobně jako klient v bodě 10.

Schéma 8-1 Úvodní fáze navazování spojení protokolu SSL

Zprávy typu *ClientHello* a *ServerHello* obsahují položky *Version*, identifikující verzi protokolu (1.0, 2.0, 3.0 a v případě TLS 3.1, 3.2 či 3.3), položku *RandomNumber*, která je využita jako výchozí hodnota pro vstup kryptografických funkcí. Dále jsou to *SessionID*, *CipherSuite* a *CompressionMethod*. Položka *SessionID* obsahuje identifikační údaje pro sezení, *CipherSuite* údaje o podporovaných kryptografických algoritmech, velikostech klíčů pro komunikaci a *CompressionMethod* údaje související s kompresí komunikace. U *CipherSuite* je v normě SSL specifikováno, že si server vybírá z klientské nabídky šifer.

Zprávy typu *Certificate* obsahují certifikační řetězec, který obsahuje certifikáty klienta až certifikát nejvyšší certifikační autority. Komunikační strany vzájemně ověří certifikáty a zjistí, zda odpovídají správným stranám, nevypršela platnost nebo nejsou v revokačních seznamech (viz 7.1). Konečně zpráva *CertificateRequest* obsahuje seznam CA a seznam typů certifikátů, které jsou pro server akceptovatelné [77].

Dalším schématem, který SSL využívá, je tzv. *record protocol*, jehož hlavním účelem je zajištění zabezpečení komunikace mezi serverem a klientem po fázi handshakingu pomocí vyměněného klíče. Schéma komunikace probíhá podle schématu Schéma 8-2 Komunikační schéma.

1. Fragmentace dat do 16 KB bloků.
2. Beztrátová komprese fragmentů.
3. Autentizace fragmentů:
    a. Výpočet MAC přes *čítač* (prevence proti útoku opakováním), *délku* a *datový obsah*.
    b. Připojení hodnoty za konec komprimovaného fragmentu.
4. Doplnění výplně k zarovnání na násobek délky bloku pro přípravu k zašifrování symetrickou šifrou.
5. Zašifrování celého obsahu (vč. výsledku MAC).

Schéma 8-2 Komunikační schéma protokolu SSL

Takto zašifrované fragmenty jsou zaslány transportním protokolem TCP.

Pro případ ukončení komunikace je zasílána speciální zprává *ClosureAlert*. Pokud je zaslaná vzájemně ze strany serveru klientovi a klientem serveru, je komunikace řádně ukončena. Tato zpráva je do protokolu zavedena právě až od SSL verze 3.0 a byla zavedena jako prevence proti tzv. truncation útoku.

8.1 SSL 1.0

Již v roce 1994 se společnost Netscape snažila o vytvoření protokolu SSL, který ve své první verzi neměl veřejný standardizační dokument ani praktickou implementaci. Společnost Netscape jej uchovávala jako interní dokument a až o několik měsíců později, kdy dokument nahradil standart SSL 2.0, tento standart zveřejnila.

8.2 SSL 2.0

První zveřejněný SSL standart. Bezpečnost verze 2.0 stále padá na několika koncepčních zranitelnostech. SSL 2.0 nemá žádnou kontrolu pro handshake, útočník může podvrhnout obě strany – již známý *man-in-the-middle útok*, aniž by si toho legitimní strany všimly. Dále je možné provést tzv. *truncation útok*, pomocí kterého útočník ukončuje spojení pomocí flagu v TCP hlavičce – FIN. Tímto krokem může útočník ukončit legitimní SSL spojení [100].

Posledním a úmyslným bezpečnostním oslabením je velké omezení velikosti délky otisku autentizační hashovací funkce na 40 bitů. Omezení je vpraveno uměle z důvodů vojenských exportních zákonů, které byly na přelomu 20. a 21. století v USA zrušeny. Vzhledem k faktu, že seznam šifer poskytovaný serverem klientovi není autentizován, může útočník podstrčit právě lehce luštitelný seznam slabých šifer, které jsou i pro útočníka luštitelné.

Není tedy vůbec vhodné SSL verze 2.0 používat.

8.3 SSL 3.0

Další verze SSL, která opravuje nedostatky z verze 2.0, byla vydaná v listopadu roku 1996. SSL 3.0 zavedla podporu podpisových algoritmů RSA/DSS a výměnu klíčů pomocí algoritmu DH, přidala autentizaci klientů i odstranění možnosti truncation útoků. Protokoly, které bylo možno použít na symetrické šifrování: DES, 3DES, IDEA, RC2, RC4 a šifru Fortezza (Skipjack) – především v režimu CBC. Hashovací funkce MD5 byla rozšířena o SHA-1 v možné kombinaci s MAC pro zajištění autentizace a integrity zpráv.

Vzhledem ke zpětné kompatibilitě protokolu umožňuje SSL této verze útok *Version Rollback*. Útok spočívá v oklamání serveru tím, že klient bude předstírat ve fázi navazování spojení podporu protokolu pouze ve verzi 2.0, ve kterém je možné využít jeho slabin [100].

Přidává podporu několika režimů výměny klíčů – *autentizovaný server*, *autentizovaný server i klient* a konečně *anonymní režim* (Anonymní režim z podstaty povoluje *man-in-the-middle* útok). Velmi zajímavým přístupem k řešení budoucí otázky bezkoliznosti je použití obou podporovaných hashovacích funkcí (MD5 i SHA-1) v jeden okamžik [101].

Nicméně i přes možné zmíněné problémy je celkově novější verze protokolu SSL velkým zlepšením oproti předchůdci.

8.4 TLS 1.0

Protokolu SSL 3.0 se chytla v roce 1997 i skupina IETF a jako jeho derivát se v roce 1999 vynořil protokol TLS, specifikován blíže v RFC 2246. Z pohledu porovnání SSL zjednodušuje výpočet hashovací funkce ve zprávě *CertificateVerify* – SSL počítá otisk z šifrovacího klíče, zprávy a výplně, zatímco TLS pouze ze zprávy. Důvodem zjednodušení může být zbytečné vystavení šifrovacího klíče a částečně také zbytečná složitost výpočtu, když se předpokládá bezkoliznost (což jak již víme, není u funkce MD5 pravda).

Dalším rozšířením v porovnání se SSL 3.0 je použití autentizační funkce HMAC a vynuceného použití 3DES šifry namísto původně volitelné možnosti. Podle RFC 4346 je možné využít CPA (Chosen-plaintext) útoků. Dřívější verze využívaly normy PKCS #1 verzi 1.5, které umožňovaly útoky pomocí vybraných textů (k dešifrování stačilo cca 1 milion pokusů u 1024-bitového klíče). Problém byl použitím PKCS#1 verze 2.0 vyřešen. Mým doporučením by bylo omezení množiny použitých algoritmů na doporučované, což částečně použitím šifry 3DES řeší (viz také kap. 10).

8.5 TLS 1.1

Novější standart TLS 1.1 z dubna roku 2006 již nevyžaduje restartování sezení v případě nevyžádaného pádu spojení. Oproti dřívější verzi se také zlepšila důvěrnost ve smyslu zákazu užití exportních algoritmů, změnila se práce s inicializačními vektory jako ochrana proti CBC útokům. Objevila se také změna použití PKCS #1 protokolu na novější verzi 2.1 (RFC 3447).

8.6 TLS 1.2 a budoucnost

Zatím technický standart v podobě draftu TLS 1.2 z března 2008. Změny, o kterých bylo v návrhové komisi diskutováno, [102] se především týkají hashovacích funkcí. Spekuluje se o použití návrhu SHA-256 jako implicitní variantě. Zajímavým návrhem je omezení použití původních MAC funkcí ve prospěch nového režimu GCM, který by jednotným klíčem řešil jednak autentizaci a také šifrování. Autor a koordinátor nového standartu Eric Rescorla hovoří o dokončení nového TLS protokolu 1.2 do konce roku 2008.

Z bezpečnostního hlediska se v posledních letech vychytávají už pouhé detaily, které opravují nedostatky třetích stran a reagují na nově objevené hrozby. Standarty TLS proto hodnotím jako dostatečně bezpečné pro všechny způsoby použití. Opět bude velmi záležet na implementacích, které budou protokol uvádět do praxe.

9 VPN

VPN (Virtual private network) je pojem pro bezpečné virtuální privátní sítě, využívající pro své vybudování tunelovací komunikační protokoly a komunikující standartně prostřednictvím internetu nebo jiného typu sítě. Často se tak propojují sítě mezi sebou, nebo připojují domácí počítače do kanceláře. Lze jej tedy využít také k získání plnohodnotného přístupu do firemní sítě. VPN pomocí dílčích protokolů často zajišťuje prostředky, které v rámci VPN umějí řešit autentizaci, integritu a také důvěrnost. VPN využívá tunelující protokoly, které si následně podrobněji rozebereme.

Pro připomenutí - samotný pojem *tunelování* znamená přenášení dat v zapouzdřené podobě prostřednictvím jiného protokolu. Nejběžnějšími metodami pro tunelování jsou PPTP (Microsoft), L2TP, SSL a IPsec.

9.1 IPsec

IPsec je rozšíření protokolu IP na síťové vrstvě a zároveň množina komunikačních protokolů. Lze jej použít pro zvýšení bezpečnosti komunikace a tvorbu VPN. IPsec byl publikován v roce 1995, následně v roce 1998 v RFC 2401 a 2412. Protokol IPv4 volitelně rozšiřuje a v IPv6 je jeho povinnou součástí. Protokol z bezpečnostního hlediska zajišťuje důvěrnost, integritu a nepopíratelnost (pomocí HMAC-SHA1 a pro důvěrnost šifry 3DES a AES v CBC režimu) s autentizační hlavičkou a ESP (encapsulating security payload).

Pro prozkoumání je nutné ještě zmínit 2 režimy činnosti IPsec – transportní a tunelovací režim. Transportní režim (využíván nejčastěji pro spojení 2 klientů) využívá stávající IP hlavičku, kde je IPsec jako rozšíření, na druhou stranu tunelovací režim (tento režim lze využít pro VPN) využívá nového IP paketu na zabalení hlavičky IPsec. Data jsou v prvním případě šifrovaná samostatně, hlavičky jsou otevřené a ve druhém případě je šifrován celý zabalený IP datagram. V rámci IPsec existují 2 typy paketů – autentizace a integrita IPsec je zajišťována pomocí tzv. AH (viz dále) a šifrování ESP. Protokol poskytuje sekvenční čísla, pro ochranu proti tzv. útokům pomocí zopakování zpráv. Podobný mechanismus ochrany využívá taky ESP.

Bezpečnostní politiky (příchozí a odchozí) jsou uloženy v položce SPD. Pro každý paket, připravený k odeslání se zjišťuje, jak bude paket zpracován – co bude v rámci IPsec zpracováno, jak a co bude odmítnuto.

9.1.1 Authentication Header

Autentizace je v IPsec realizována pomocí protokolu Authentication Header (dále také AH). Bližší popisy uvádějí v RFC 2403 a 2404, využívající hashovací funkce HMAC-MD5 a HMAC-SHA1

a RFC 2857 HMAC-RIPEMD-160. Jak bylo zmíněno v podkap. 4.4.10, funkce HMAC zajišťuje autentizaci hashovací funkce uživatelským klíčem, takže výše zmíněné funkce jsou považovány za bezpečné. Výstup funkce tedy zajišťuje kromě autentizace také integritu – útočník sice může vypočítat běžný otisk, ale nemůže k tomu navíc podvrhnout klíč vlastníka, který je k výpočtu potřeba. V protokolu AH a ESP se označuje hodnota výstupu autentizačního otisku pod položkou ICV (Integrity Check Value).

AH může být použit samostatně nebo spolu s ESP.

9.1.2 Encapsulated security payload

Encapsulated security payload (dále také ESP) je protokol určený především pro poskytnutí zabezpečení přenášených dat. Hlavičky uvádějí v RFC 2406, pak 2405, 2403, 2451, 4309. Dokumenty RFC nevyhrazují nutnost použití šifrování, teoreticky je možné použít ESP i bez šifrování (NULL), což ovšem není účel samotného ESP. ESP hlavička je v transportním režimu uložena hned za IP hlavičku nebo v tunelovacím režimu před zapouzdřený IP datagram. Za ESP hlavičkou následuje šifrovaný blok dat.

Hlavička sestává z položek, které jsou klíčové v procesu navazování spojení – např. pomocí protokolu IKE. SPI určuje identifikaci SA (viz dále) – v kombinaci s cílovou adresou se používá k identifikaci zabezpečení. Padding je položka datové výplně pro některé šifrovací funkce, které vyžadují přesnou velikost bloků tak, aby měly požadované velikosti násobku bloku dat. ICV je počítáno z položek: SPI, sekvenční číslo, Payload data, Padding, Pad length a Next header.

ESP a AH zajišťují autentizaci, nicméně ESP nezajišťuje autentizaci dat IP hlavičky pro případ použití transportního režimu.

9.1.3 Internet Key Exchange

Internet Key Exchange (dále také IKE) je protokol využívající protokolů ESP a AH k navazování bezpečné komunikace a sdílení informací mezi klienty v rámci IPsec – tzv. Security Association (dále také SA). Lépe řečeno – SA nám udává vztah mezi n entitami ($n > 1$), který popisuje, jak budou v komunikaci využívat bezpečnostní služby [104]. SA můžeme identifikovat pomocí položky SPI, cílové adresy a bezpečnostního protokolu (ESP nebo AH).

IKE je používána v rámci ISAKMP (IPsec key management protocol), který spravuje SA. Tento model je dobrý z hlediska návrhu, pokud bude nalezena chyba v dílčím protokolu, může ISAKMP pro výměnu klíčů použít protokol jiný. ISAKMP pak sám o sobě řeší především ovládání SA.

SA na každé straně spojení zajišťuje uložení informací pouze pro jeden směr (příchozí nebo odchozí) spojení – protokol, algoritmy a klíče. IKE může provést autentizaci výměnou bezpečných klíčů. IKE využívá UDP pakety, spojení standartně na portu 500.

Bezpečnost IKE bohužel umožňuje několik útoků. Útok odrazem, při kterém útočník zopakuje hodnoty iniciátora, aby mohl přeskočit autentizaci [103]. Další útok spočívá v tom, že útočník může modifikovat SA odpovědi a vybrat nejslabší (40 bitový) klíč a následně útokem hrubou silou najít klíč.

V roce 2005 byl ohlášen nový standart IKEv2 [104], který by měl především bránit DoS útokům. Zavádí podstatná zjednodušení, jednotlivé zprávy mají sekvenční číslování, iniciátor je zodpovědný za znovuzaslání zpráv, pokud dojde k jejich ztrátě.

9.1.4 IPsec a bezpečnost

IPsec jako koncept poskytuje podle Schneiera výborné bezpečnostní vlastnosti [103], nicméně kritizuje zbytečnou složitost návrhu. Jak jsme si řekli, IPsec poskytuje protokoly AH a ESP v tunelovacím a transportním režimu. Pokud si strany přejí autentizovat paket, mají 4 možné způsoby použití: transportní/AH, tunnel/AH, transport/ESP s žádným šifrováním, a tunelovací/ESP s žádným šifrováním. Doporučuji nepoužívat transportní režim, stejně tak jako AH protokol, upřednostňoval bych protokol ESP s povinnou autentizací a s doplňkovou možností šifrování. Dalším spíše drobnějším nešvarem je umožnění použití slabých klíčů IPsec (např. šifra Blowfish má skupinu známých slabých klíčů). Rovněž bych doporučil nepoužívat dnes zastaralou blokovou šifru DES a jasně definovat množinu důvěryhodných šifer, které lze v rámci ESP využít.

Dalším faktem je to, že antireplay ochrana, kterou protokol umožňuje zapnout, je závislá na použití autentizace a její volba závisí na příjemci. Příjemce ji nemusí kontrolovat a v rámci komunikace musí být povolena explicitně.

IPsec lze tedy považovat za kvalitnější protokol než PPTP nebo L2TP nejenom pro budování VPN, nicméně díky nalezeným zranitelnostem a možnostem praktického zneužití, které byly objeveny, je nelze doporučit [103].

9.2 Point-to-Point Tunneling Protocol

Point-to-Point Tunneling Protocol nebo také PPTP je tunelovací technologie společnosti Microsoft pro vytváření VPN pomocí protokolu PPP. PPTP se standartně zapouzdřuje pomocí GRE (General Routing Encapsulation) a k zasílání je využito IP paketu. Pro úplnost – GRE je tunelovací protokol vyvinutý původně společností CISCO. PPTP bylo poprvé představeno v RFC 2637. Autentizace je v PPTP řešena prostřednictvím protokolu MS-CHAP a EAP-TLS. Po publikování úspěšné kryptoanalýzy [104] byla Microsoftem následně vydána nová verze MS-CHAP2, která opravovala některé chyby předchozí verze (viz také 6.3.1). Samotná implementace doplňkového šifrování pomocí protokolu MPPE (Microsoft Point to Point Encryption) bude rozebrána dále.

Původní PPTP má 3 autentizační možnosti: 1) pomocí hesla v otevřené podobě, 2) otisk hesla (hashovací funkcí), 3) MS-CHAP protokol. (Microsoft autentizace).

První dvě možnosti neobsahují volbu zašifrování, druhá stála čistě na bezpečnosti tzv. Lan Manager hashovací funkce.

9.2.1 Lan Manager hashovací funkce

Lan Manager hashovací funkce (dále také LM) neznamenala pro útočníka v případě použití v prostředí Microsoft sítí velké úsilí. Samotný výpočet se dá shrnout jako:

1) Se vstupním heslem se provede jednoduchá konverze na řetězec o konstantní délce 14 bajtů (oříznutí delšího hesla, doplnění kratšího hesla nulami).

2) Konverze na upper case.

3) Rozdělení 14 bajtů na dvě 7-bajtové poloviny.

4) Hash je spočítán jako zašifrování konstantní hodnoty klíči z kroku 3), výsledek zašifrování je spojen a je prezentován jako 16-bajtový výsledek LM hashe.

Schéma 9-1 Výpočet Lan Manager hashovací funkce

U LM nebyla použita žádná sůl, často používaná v UNIXových operačních systémech pro ukládání hesel v bezpečné podobě. Ke zjištění hesel byl potřeba pouze slovníkový útok případně útok hrubou silou. Navíc - hesla jsou vnitřně konvertována na upper case, což dále omezuje velikost množiny možných použitých znaků zadaných hesel jednak u útoku hrubou silou, a také u slovníkového útoku. Zajímavostí je, že je použita šifra na základě DES a sedmice jsou šifrovány samostatně.

9.2.2 NT hashovací funkce

NT hashovací funkce (dále také NT) je hashována oproti LM jako celek [105], funkce pracuje opět s 14-bajtovým vstupem, který konvertuje na Unicode. Pro výpočet otisku používá hashovací funkci MD4, jejímž výstupem je 16-bajtový otisk. Výhodou oproti LM je používání zohlednění v rozdílu velikosti znaků (case sensitivity), bohužel solení NT opět nepoužívá, nabízí se tedy podobné útoky jako u LM.

9.2.3 Bezpečnost

První verze PPTP byla podle [105] bezpečnostně velmi slabá. Nenejnom autentizační protokol MS-CHAP nabízel možnost slovníkového útoku, design u MPPE 40 a 128-bitové šifry je také nedostatečně navržen. Především 40-bitový klíč se spočítá pomocí LM, 128-bitový pomocí

hashovací funkce SHA, aplikované na řetězec, složený z výstupu NT a náhodného slova u MS-CHAP. MPPE používá šifru RC4, u 40-bitového klíče šifry lze spočítat vzory pro odchycené hlavičky PPP v rozumném čase. Nicméně bezpečnost generovaného klíče neodpovídá teoretickým možnostem délky klíče. Pomocí vstupních dat – nekvalitních otisků z hesel, ze kterých se klíče generují, jsou praktické vyhlídky mnohem méně bezpečné, např. oproti náhodnému klíči stejné délky. Dále je pak nutné vzít na vědomí teoretické možnosti šifry RC4 (viz 4.2.7).

Další možnosti útoku nabízel protokol PPTP během vyjednávání spojení, ve kterém mohl způsobit DoS útok na kohokoli, kdo protokol využíval. Novější verze protokolu PPTP přinesla zlepšení – omezení LM ve prospěch NT, přidání autentizace serveru, MPPE používá unikátní klíče pro každý směr komunikace. Podporované velikosti klíčů však zůstaly.

Pro úplnost uvádím odvození klíčů, které se provede takto:

1) Provedení hashování SHA na NT hash, 24-bajtové odpovědi z MS-CHAPv2 a 27-bajtové konstanty.
2) Oříznutí výsledku z bodu 1) na 16 bajtů – z toho získá klíč session.

Použití protokolu MS-CHAPv2 pro autentizaci stále nedoporučuji. Pokud je použita kombinace autentizačního protokolu EAP-TLS, může tento protokol bezpečnou komunikaci mnohem zlepšit. Pak lze tuto kombinaci doporučit pro využití protokolu PPTP.

9.3 Layer two tunelling protocol

(Layer two tunelling protocol (dále také L2TP) představen v r. 1999 v RFC 2661 opět jako rozšíření PPP společností CISCO, Microsoft a dalšími. Jeho koncepce čerpá jednak z protokolu PPTP a také z tunelovacího protokolu L2F společnosti CISCO [106]. Protokol je tedy podporován operačními systémy Windows 2000 a novějšími a Unixovými systémy, stejně jako zařízeními Cisco.

Komunikace je vyjednávána mezi klientem (LAC – iniciátor spojení a přístupový koncentrátor) a koncovým L2TP serverem (LNS – zřizuje konkrétní relace).

Módy komunikace mohou obsahovat několik režimů – nepovinný tunel, model povinného tunelu a L2TP s více přechody. U nepovinného modelu si vytváří tunel klient, u povinného provádí vzdálené vytáčení domovská brána (zde je ještě rozlišen režim příchozího volání a vzdáleného vytáčení). Posledním typem připojení pak brána L2TP funguje jako LAC pro daný LNS a jako LNS pro skupinu LAC.

Protokol sám neposkytuje silnou autentizaci nebo šifrování. Právě proto je často doplňován na vyšší vrstvě protokolem IPsec (RFC 3193), aby v rámci komunikace zajistil všechny základní

bezpečnostní cíle – důvěrnost, autenticitu a integritu. Takto se tak děje u procesu navázání spojení L2TP/IPsec (viz schéma 9-2).

1) Vyjednání SA v rámci IPsec (např. pomocí IKE).
2) Navázání spojení ESP v transportním (tedy klient-klient) režimu spojení.
3) Vytvoření L2TP tunelu mezi koncovými body v rámci SA.
4) Komunikace v šifrovaném režimu mezi stranami.

Schéma 9-2 Navázání spojení u protokolu L2TP

Protokolu je často vyčítán slabý výkon a nadměrná režie [106]. V nejnovějším návrhu verze (RFC 3931) protokolu (L2TPv3) je vidět obecnější návrh. Kromě PPP je podporována celá řada linkových protokolů (např ATM, Frame Relay atp.) a rozšíření adresovatelného tunelovacího a session identifikátoru z 16 na 32 bitů.

Z toho, co zde bylo napsáno, bezpečnost protokolu závisí do značné míry na IPsec, který je spolu s L2TP nejčastěji používán.

9.4 Secure Socket Tunneling Protocol

Jednoznačně nejnovějším protokolem pro vytváření VPN společnosti Microsoft z roku 2007, který je také částečnou odpovědí na protokoly PPTP či L2TP. Namísto původních standardů se Microsoft rozhodl implementovat VPN řešení pomocí rozšířeného protokolu HTTP a SSL [107], zajišťuje pomocí něj autentizaci, důvěrnost i integritu. Odpadají tím problémy řešit zapouzřování IP paketů pomocí GRE, které mnozí internetoví poskytovatelé filtrují. Microsoft plánuje Secure Socket Tunneling Protocol (dále také SSTP) využít od nové verze systému Windows – Longhorn a v posledních vydáních Windows Vista.

IP	TCP	SSTP	PPP	IP

Schéma 9-3 Zapouzdření SSTP

Struktura paketu napovídá o způsobu zabalení SSTP, nejprve je IP paket (verze 4 nebo 6) zabalen do PPP a následně do SSTP. Tato šedá tabulka pak označuje, která část je šifrována pomocí principu SSL. Činnost protokolu SSTP lze ve stručnosti popsat ve schématu 9-4.

1. Navázání spojení se serverem (port 443), zaslání požadavku na vytvoření SSL sezení.
2. Server zašle klientovi certifikát.

3. Klient provede validaci certifikátu, zašle informace o využívaném šifrovacím algoritmu pro sezení a vygeneruje klíč sezení, kterým zašifruje obsah zpráv.

Následně je veškerá komunikace šifrována klíčem sezení.

4. Klient zašle http požadavek se zprávou SSTP server.
5. Klient dále vyjedná se serverem bezpečný SSTP tunel.
6. SSTP klient vyjedná PPP spojení se serverem - proběhne autentizace klienta a následně začne může probíhat klasická komunikace se zapouzdřenými IP pakety.

Schéma 9-4 Činnost protokolu SSTP

Žádná bližší bezpečnostní analýza protokolu nebyla v době psaní této práce k dispozici. Vzhledem k návrhu se tedy domnívám, že bezpečnost může být stejná nebo horší, než je bezpečnost jednotlivých komponent – tedy způsob využití SSL pro VPN, který také popíši dále. Domnívám se, že možné útoky mohou směřovat aktivity spíše na odepření spojení či zneužití implementačních slabin – např. DoS, než na možnost ohrožení důvěrnosti či integrity protokolu.

Tento krok je podle mého názoru správným směrem – výměna klíče pomocí SSL podle tohoto návrhu nedodává systému závažnější bezpečnostní slabiny – bude tedy záležet na konkrétních implementacích, případně na zveřejnění detailnějšího postupu činnosti jednotlivých kroků protokolu. Autentizace na úrovni PPP je řešena „uvnitř“ šifrovaného spojení, hrozba podvržení je opět omezena i v případě použití slabšího autentizačního protokolu.

9.5 Multiprotocol Label Switching

Protokol Multiprotocol Label Switching (dále také MPLS) je v této skupině výjimečný [108]. MPLS se umisťuje mezi linkovou a síťovou vrstvu a byl dříve vytvořen jako sjednocující technologie pro přenos dat (lze využít pro sítě IP, ATM atd.). Naproti výše zmiňovaným protokolům dokáže vytvořit spojení jednak typu bod-bod, a jednat také vícebodové spojení.

MPLS tvoří VPN sítě na síťové vrstvě (L3VPN), linkové (L2VPN) a také pro vícebodé VPN (VPLS). VPLS umožňuje vybudování mezi uzly kompletní sítě bod-bod připojení. V multipoint VPN je VPN identifikováno pomocí VPN ID.

MPLS Layer-2 VPN poskytuje oddělení mezi sítí internetového poskytovatele a zároveň soukromou sítí. Hlavní myšlenka tedy spočívá ve vložení krátké 32-bitové MPLS hlavičky před pakety. V MPLS sitích existují směrovače LER, jejichž úkolem je práce s návěstími MPLS hlaviček – vkládají MPLS návěstí do vstupních, resp. vyjímají z výstupních paketů. Pakety se třídí do tříd podle vlastního návěstí – závisí na typu dat – např. IP adresa, typ dat, aj. V síti ještě existují přepínače (tzv.

LSR), které si pravidelně vyměňují informace o vzájemné dostupnosti a upravují pak cesty. Pro potřeby VPN komunikace ještě definujeme směrovače poskytovatele - tzv. PE (Providers edge – směrovač na hranici poskytovatele), směrovače klienta CE (Customer Edge – zákaznické směrovače, viditelné z páteřní sítě) a samotné vnitřní směrovače poskytovatele – směrovače P v jádru MPLS sítě. PE a P (souhrnně tvoří jádro) takové budované VPN sítě. Směrovač P nemusí vědět nic o uživatelských CE směrovačích [108].

L3VPN podporuje pouze IP. Cílová adresa je vyhledána ve směrovacích tabulkách a pakety jsou tak zasílány pomocí protokolu BGP (Border Gateway Protokol). Transport pak probíhá prostřednictvím virtuální cesty. PE a CE si vymění seznam cest, které se využijí na transport v rámci VPN. VPN komunikace klientů se tedy udržuje na úrovni koncových CE s tím, že tok informací (a řídících dat) řeší směrovače P a PE v rámci jádra.

VPN u MPLS neumožňuje na své logické úrovni vidět jádro protokolu [109], vrstvy jsou určitým způsobem odděleny. Směrovače P nemohou být z VPN sítě napadeny, protože jak bylo zmíněno, nejsou dostupné. Z praktického hlediska napadení směrovače typu PE by mělo za následek drastické ohrožení veškeré komunikace ve všech VPN, proto se klade na zabezpečení směrovačů velký důraz.

Jako jeden z návrhů opatření proti potenciálním DoS útokům je zavedení tzv. ACL (Access Control List), které omezí přístup pouze na nutné porty služby a pouze ze strany CE. Dále MPLS disponuje ochranu proti tzv. spoofingu. Nicméně protokol MPLS kromě schopnosti budovat VPN neumožňuje žádnou garanci důvěrnosti, integrity a autentizace, je tedy nutné komunikaci zajistit další vrstvou komunikace. Pro tyto účely se nejčastěji využívá IPsec (pro komunikaci koncových stran CE), nicméně vzhledem k charakteru IPsec, je podle mého názoru třeba mít na paměti jeho reálnou bezpečnost.

9.6 SSL/TLS a VPN

Z hlediska důvěrnosti, autentizace i integrity se s použitím SSL/TLS jedná jednoznačně o velmi zajímavou možnost řešení pro budování virtuálních privátních sítí. Využitím známého protokolu SSL resp. TLS jako nástupce, které se pro tento účel přímo nabízí se ve VPN automaticky odstraní hned několik překážek.

1. Využije se systém, který kompletně řeší bezpečnost, autentizaci i integritu na relační úrovni OSI.
2. SSL a TLS díky své skoro patnáctileté historii poskytuje sám o sobě prověřený a dlouhodobě spolehlivý mechanismus pro vytvoření bezpečného bodového spojení (prakticky již od r. 1996 a verze SSL 3.0 s příhlédnutím na možné ošetření potenciálních útoků lze nad protokolem přemýšlet jako nad vhodným kandidátem pro účely VPN).
3. Obecná kompatibilita - přístup je konfigurovatelný často připojením webového klienta s podporou SSL/TLS na VPN server, jako web server (tak je možné se připojit opravdu odkudkoli bez dodatečných voleb).

Díky vysoké úrovni (OSI – relační úroveň) může být nevhodné použití SSL/VPN tam, kde je vhodnější použít tunelování na nižších vrstvách. Celý systém mohou narušit DoS útoky, které mohou zneužít poměrně zdlouhavý proces navazování šifrovaného spojení (viz také kap. 8).

Ze všech řešení, která se doposud nabízela k dispozici, je využití na bázi SSL/TLS z provedeného přehledu technologiemi pro tvorbu VPN podle mého názoru zaslouženým vítězem.

9.6.1 Implementace OpenVPN

Vzhledem k širokému praktickému uplatnění VPN systémů, rozhodl jsem se ještě v rámci této práce provést bližší pohled na nejznámější VPN implementaci na bázi SSL/TLS, která je open-sourcové filosofie - OpenVPN, která podporuje celou řadu operačních systému (Windows 2000 a vyšší, Unixy – *BSD systémy, Mac OS, Linux) a je dostupná ve verzi 2.0. OpenVPN server naslouchá implicitně na UDP portu, na který se jednotliví klienti připojují. Činnost systému odpovídá principu SSL/TLS z kap. 9 a podkap 8.6.

Jak zmiňuje Hosner [110] ve své práci, stojí Yonanovo dílo – OpenVPN na tom nejnovějším, co může běžná kryptografie obsáhnout. Porovnáme-li však řešení SSL/TLS s IPsec řešením, získáme v komunikaci na symetrickém šifrování srovnatelné možnosti (implicitně Blowfish v CBC režimu se 128-bitovým klíčem, SHA-1 jako hashovací funkce) s velkou možností konfigurace a často zbytečnou podporou historických šifer.

Otevřenost open-sourcového řešení OpenVPN podle mého názoru pomáhá a směřuje k naplnění Shannonova principu [8], funguje však jako dvojsečná zbraň. Historie projektu OpenVPN ukazuje množství zranitelností, které mohou způsobit přímé ohrožení a zneužití hostitelských systému komunikačních stran. Hlavním bezpečnostním rizikem i při bezchybném a bezpečném návrhu

schématu spočívá v jednodušším hledání chyb v implementacích v porovnání s hledáním zneužitelností v kryptografických konceptech. Útočník může disponovat klíčovou znalostí, která dokáže obejít bezpečnost dané verze softwaru, a kterou se může veřejnost dozvědět až mnohem později, než samotný útočník (často je útočník první, kdo chybu objeví a staží se své znalosti patřičně využít). Na rozdíl od kryptoanalýzy, kdy má veřejnost velmi omezenou možnost proniknout do tajů kryptoanalýzy, jsou útoky na implementace otevřeny mnohem širšímu počtu útočníků i samotných útoků.

Pro úplnost a objektivitu je nutné dodat, že toto není problém pouze OpenVPN, ale veškerého softwaru, používaného kdekoli v praxi.

10 Závěr

Jedním z konečných výstupů této práce jsou přehledové tabulky, které jasně ukazují zjištěné přínosy práce ve smyslu porovnání kryptografických algoritmů a protokolů z různých hledisek. Ještě jednou bych zde odkázal metodiku hodnocení algoritmů podle kapitoly 4.5 a jiných, ze kterých bych své názory a výsledná doporučení opřel. Metodika hodnocení protokolů vychází především z jednotlivých vědeckých publikací, na které je odkazováno a také vlastních názorů autora této práce, podložených odkazovanými dokumenty.

10.1 Symetrické šifrovací algoritmy

U symetrických algoritmů je parametr bezpečnosti dán především dostatečně kvalitním návrhem bezpečnostního algoritmu, podporou dostatečně dlouhého klíče a také časovým prověřením samotného algoritmu. Velmi dobře v celkových výsledcích dopadly algoritmy starší.

Symetrické šifrovací algoritmy	Délka klíče (bity)	Subjektivní doporučení	Nejlepší publikovaná kryptoanalýza (složitost výpočtu klíče pro implicitní velikost klíče)	Poznámky
DES	56	Ne	2^{48} DES operací	Nedostatečná délka klíče
3DES	112, 168	Ano	2^{90} operací	
SKIPJACK	80	Ne	2^{44} operací	
IDEA	128	Ano	2^{43} KPA a zároveň 2^{115} šifrovacích operací	Zmíněný útok je účinný pouze pro 7 rund algoritmu
RIJNDAEL(AES)	128	Ano		
RIJNDAEL(AES)	192	Ano		
RIJNDAEL(AES)	256	Ano	2^{230} operací	
RC2	8-128	Ne	2^{32} CPA	
RC4	8-2048	Ne	2^{33} operací	
RC5	8-2048	Ano	2^{44} operací u 64-bit klíče	Doporučeno pouze pro delší klíče
RC6	128	Ano	2^{74} operací / 2^{118} CPA	
RC6	192	Ano		
RC6	256	Ano		
SERPENT	128	Ano	2^{118} CPA	
SERPENT	192	Ano		
SERPENT	256	Ano		
BLOWFISH	32-448	Ano	2^{129} KPA, pro slabé klíče 2^{65} KPA	Doporučeno pro klíče 128 a delší. Doporučeno eliminovat slabé klíče
TWOFISH	128	Ano	Teoreticky 2^{51} CPA	
TWOFISH	192	Ano		
TWOFISH	256	Ano		

Tabulka 10-1 Zhodnocení symetrických algoritmů

10.2 Asymetrické algoritmy

Nejznámější v praxi používané asymetrické šifrovací a podpisové algoritmy, jsou podle svého charakteru rozlišeny na ty, které využívají problémů diskrétního logaritmu, eliptických křivek a faktorizace. Nejlepším modelem pro každou šifru je její matematicky prokazatelná bezpečnost – příkladem může být šifra Rabin, u které je dokázána výpočetní ekvivalence úlohy zlomení s jinou velmi obtížně řešitelnou úlohou.

Algoritmus	Doporučená minimální délka klíče (bity)	Subjektivní doporučení	Poznámky
RSA	2048	Ano	Nejlepší metodou je General number field sieve (GNFS) (exponenciální časová složitost)
Elgamal	2048	Ano	
Knapsack	-	Ne	Algoritmus řešení knapsack problému byl nalezen s polynomiální časovou složitostí.
DSA	2048	Ano	Pollardova metoda – časová složitost $(\prod * r/2)^{1/2}$
ECSDA	256	Ano	Pollardova metoda – časová složitost $(\prod * r/2)^{1/2}$
Rabin	2048	Ano	

Tabulka 10-2 Zhodnocení asymetrických algoritmů

10.3 Hashovací funkce

U hashovacích funkcí se zkoumaly vlastnosti jednosměrnosti a především odolnost proti kolizím. Tam, kde je uvedena hodnota narozeninového útoku, je myšleno, že doposud nebyly nalezeny materiály, které by prokazovaly oslabující útok více, než umožňuje narozeninový paradox viz 4.1.4. Poznámkou typu *Pouze s MAC* je myšleno, že hashovací funkce není vhodná z hlediska odolnosti proti kolizím, ale její použití v rámci autentizačních klíčových algoritmů typu MAC je stále bezpečné.

Algoritmus	Délka otisku (bity)	Subjektivní doporučení	Odolnost proti kolizím (časová složitost)	Poznámky
MD2	128	Ne	2^{104} (nalezen pouze útok na jednosměrnost)	Nalezeny kolize kompresní funkce / pouze s MAC
MD4	128	Ne	2^{8}	
MD5	128	Ne	2^{30}	Pouze s MAC
RIPEMD	160	Ne	2^{18}	Pouze s MAC
SHA-0	160	Ne	2^{33}	Pouze s MAC
SHA-1	160	Ne	2^{63} operací	Pouze s MAC
SHA-2	224	Ano	Narozeninový útok	
SHA-2	256	Ano	Narozeninový útok	
SHA-2	384	Ano	Narozeninový útok	
SHA-2	512	Ano	Narozeninový útok	
TIGER	160, 192	Ne	2^{48}	Pouze s MAC
WHIRLPOOL	512	Ano	Narozeninový útok	

Tabulka 10-3 Zhodnocení hashovacích funkcí

10.4 Generátory pseudonáhodných čísel

U generátorů pseudonáhodných čísel jsou do subjektivního pohledu hledány takové generátory, které splňují vlasnosti tzv. CSPRNG (tedy kryptograficky bezpečném pseudonáhodném generátoru). Generátory vstupují do algoritmů, komunikačních protokolů a prostupují různými oblastmi. Pomocí generátorů se tvoří soli, generují klíče, inicializační vektory a použití zahrnuje také náhodné vstupy pro jiné použití. Vlastnosti kryptograficky bezpečných algoritmů např. spojuje to, že každý CSPRNG by měl projít tzv. *bit-next* testem – kdy na daném vzorku *k* bitů další bit nesmí mít polynomiální závislost na předchozích bitech s pravděpodobností vyšší než 0.5.

Algoritmus	**Subjektivní doporučení**	**Poznámky**
LCG	Ne	Rovněž se nedoporučuje u žádných podobných derivátů
Mersenne Twister	Ne	Po úpravě výstupu hashovací funkcí ano
Blum Blum Shub	Ano	
Yarrow	Ano	
Fortuna	Ano	

Tabulka 10-4 Zhodnocení pseudonáhodných algoritmů

10.5 Autentizační protokoly

Autentizace byla hodnocena z hlediska možností útočníka podvrhnout spojení, odposlechnout heslo/klíč spojení či jinak ohrozit bezpečnost autentizace.

Systém	**Subjektivní doporučení**	**Poznámky**
PAP	Ne	Žádné šifrování
CHAP	Ne	Možnost slovníkových útoků a útoků hrubou silou
MSCHAP	Ne	Modifikovaný CHAP
MSCHAP v2	Ne	Bezpečnost systému závisí na kvalitě hesla
EAP-TLS	Ano	Závislá na konceptu TLS
EAP-IKEv2	Ano	
LEAP	Ne	Bezpečnost systému závisí na kvalitě hesla
PEAP	Ano	

Tabulka 10-5 Zhodnocení autentizačních protokolů

10.6 VPN systémy

Cílem každé virtuální privátní sítě je zajištění bezpečného tunelu pro účely vybudování určité virtuální sítě, např. pro umožnění přístupu do vnitřní sítě podniku z vnějšího prostředí pomocí vytvoření bezpečného tunelu. VPN jsou určitým pohledem speciálním případem síťového tunelování. Součástí budování VPN je celá řada dílčích protokolů, které je potřeba pro bezpečnost takové sítě zajistit - autentizaci, důvěrnost i integritu.

Báze VPN	**Subjektivní doporučení**	**Poznámky**
IPsec	Ne	Příliš složitá množina protokolů
PPTP	Ano	Pouze s EAP-TLS
SSTP	Ne	Protokol není dostatečně zdokumentován
L2TP	Ano	Pouze s EAP-TLS či PEAP
SSL/TLS	Ano	Vhodné s verzí aspoň SSL 3.0 nebo lépe TLS 1.0

Tabulka 10-6 Zhodnocení VPN systémů

10.7 Zhodnocení

V rámci diplomové práce se povedlo zmapovat historický vývoj kryptografie od nejstarších dob až po období moderní kryptografie, stejně tak jako vliv klasické kryptografie na současnou koncepci kryptografických metod, vznik, důvod a oblasti užití různých kryptografických algoritmů. Ukázal jsem, že kryptografie měla už od pradávna značný vliv na důležité historické události a zejména v moderní době hraje čím dál důležitější roli. V práci jsou dále probrána nová paradigmata návrhu moderní kryptografie, míra jejich dodržování a důsledky nedodržování.

V oblasti moderní kryptografie, na kterou je tato diplomová práce zaměřena, se povedlo zhodnotit bezpečnost aktuálních symetrických i asymetrických šifer, hashovacích funkcí i generátorů pseudonáhodných čísel. V práci jsem také definoval a popsal základní teoretické předpoklady počítačové bezpečnosti, slabiny jednotlivých algoritmů a metody útoků na zmíněné algoritmy. Bylo ukázáno, že návrh kvalitního a bezpečného algoritmu zdaleka není triviální záležitost. Provedl jsem zhodnocení možnosti kryptografických algoritmů, které nebyly doposud prolomeny a sestavil pravděpodobný odhad stavu aktuálně používaných kryptografických metod z hlediska použitelnosti i ve střednědobém horizontu dobudoucna. Nad rámec zadání jsem vytvořil a zmapoval nejčastěji používané autentizační metody, jejich protokoly a protokoly pro tvorbu virtuálních privátních sítí a jejich úroveň zabezpečení.

Povedlo se dále hlouběji zhodnotit bezpečnostní vlastnosti digitálních podpisů, certifikátů i protokolů pro výměnu klíčů a některých běžně používaných protokolů, užitých

pro bezpečnou síťovou komunikaci a autentizaci uživatelů. Během práce jsem provedl také několik doporučení k některým protokolům i jednotlivým algoritmům.

Dualita vývoje, kterým se lidstvo přibližuje na jedné straně k zajištění stále více kvalitních kryptografických algoritmů a na straně druhé ke kryptoanalýze, jako snaze k jejímu prolomení, vede k celkově společnému cíli – nalezení prokazatelného postupu, kterým se lze co nejblíže přiblížit k dosažení základních bezpečnostních cílů. Bohužel ne všechny bezpečné algoritmy vycházejí z přesného matematického modelu a z hlediska složitosti lidstvo stále neví, jaké jsou základní vztahy, na kterých kryptografie stojí. Částečnou útěchou může být fakt, že stejně nejasné informace má i útočník. O tom, zda-li jsou však popsané bezpečné algoritmy opravdu bezpečné, rozhodne až budoucí vývoj oblasti kryptografie a v neposlední řadě také budoucí výzkum v teoretické informatice.

Literatura

[1] **Schneier, Bruce.** *Applied Cryptography, Second Edition.*: John Wiley & Sons, 1996. ISBN: 0-471-11709-9 .

[2] **RSA Laboratories.** What is cryptography? *www.rsa.com.* [Online] 2007. [Citace: 3. 12 2007.] http://www.rsa.com/rsalabs/node.asp?id=2157.

[3] **Pinkava, Jaroslav.** Základy kryptografie I. *Crypto-World.* 1998, 05.

[4] **Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone.** *Handbook of Applied Cryptography.* s.l. : CRC Press, 2001.

[5] **Pinkava, Jaroslav.** Základy kryptografie II. *Crypto-World.* 1998, 06.

[6] **Zvi Gutterman, Benny Pinkas, Tzachy Reinman.** Analysis of the Linux Random Number Generator. *Cryptology ePrint Archive.* [Online] 06. 03 2006. [Citace: 26. 03 2008.] http://eprint.iacr.org/2006/086.pdf.

[7] *Kryptologií se lidé zabývají už 4000 let.* **Klíma, Vlastimil.** 11, Praha : Hospodářské noviny, 2005, Sv. 1.

[8] **Klíma, Vlastimil.** Šifru v pytli neutajíš. *CHIP.* 1999, 01.

[9] **Phan, R. Chung-Wei.** *Cryptanalysis of full Skipjack block cipher.* [Document] Sarawak : Swinburne Sarawak Institute of Technology, Malaysia, 2002. ISSN: 0013-5194.

[10] **Singh, Simon.** *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography.* New York : Anchor Books, 2000. ISBN: 9780385495325 .

[11] Obrázek Vigenerovy šifry. [Online]: [Citace 12.12.2007.] http://www.braingle.com/brainteasers/codes/images/vigenere.gif.

[12] Obrázek Feistelovy šifry. [Online]: [Citace: 15.12.2007.] http://content.answers.com/main/content/wp/en/d/d2/Feistel.png.

[13] **Pinkava, Jaroslav.** Základy kryptografie III. *Crypto-World.* 1998, 07.

[14] —. Základy kryptografie VII. *Crypto-World.* 1998, 11.

[15] **M. Češka, T. Vojnar, A. Smrčka.** *Teoretická informatika - studijní opora.* [PDF Dokument] Brno : FIT VUT, 2007. http://www.fit.vutbr.cz/study/courses/TIN/public/Texty/oporaTIN.pdf.

[16] **Gasarch, William I.** *SIGACT News Complexity Theory Column 36.* [PDF Document] Maryland : Department of Computer Sciences, University of Maryland, 2002.

[17] **Shannon, Claude** *Communication Theory of Secrecy Systems..*: Bell System Technical Journal, 1949, Sv. 28.

[18] **Heys, Howard M.** A Tutorial on Linear and Differential Cryptanalysis. *www.engr.mun.ca.* [Online] [Citace: 20. 12 2007.] www.engr.mun.ca/~howard/PAPERS/ldc_tutorial.ps.

[19] **Biham, Eli.** *On Matsui's Linear Cryptoanalysis.* [PDF Document] Haifa : Computer Science Department - Israel Institute of Technology, 1998.

[20] **Krawczyk, Hugo.** *The order of encryption and authentication for protecting communications (Or: how secure is SSL?).* [PS Document] Haifa : EE Department, Technion, 2001.
[21] **FIPS.** *DES MODES OF OPERATION (FIPS PUB 81).* [Document] 1980.
[22] **Petr Hanáček, Jan Staudek.** *Bezpečnost informačních systémů.* Praha : Úřad pro státní informační systém, 2000. ISBN: 80-238-5400-3.
[23] **Klíma, Vlastimil.** *Základy moderní kryptologie - Symetrická kryptografie III.* [PDF Document] 2005.
[24] **Vlastimil Klíma, Tomáš Rosa.** GCM: Nové zabezpečení IPSec. *Sdělovací technika.* 2007, 6.
[25] **Nicolas T. Courtois, Gregory V. Bard.** Algebraic Cryptanalysis of the Data Encryption Standard. *eprint.iacr.org.* [Online] 2006. [Citace: 20. 12 2007.] http://eprint.iacr.org/2006/402.pdf.
[26] **Nicolas T. Courtois, Josef Pieprzyk.** Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. *eprint.iacr.org.* [Online] 09. 11 2002. [Citace: 22. 12 2007.] http://eprint.iacr.org/2002/044.pdf.
[27] **Alex Biryukov, Christophe De Canniere, Michael Quisquater.** On Multiple Linear Approximations. *Alex Biryukov (home page).* [Online] [Citace: 15. 04 2008.] http://homes.esat.kuleuven.be/~abiryuko/mla.pdf.
[28] **Paul C. van Oorschot, Michael J. Wiener.** A Known-Plaintext Attack on Two-Key Triple Encryption. *www3.sympatico.ca.* [Online] 29. 06 1990. [Citace: 22. 12 2007.] http://www3.sympatico.ca/wienerfamily/Michael/MichaelPapers/TwokeytripleDES.pdf.
[29] **Eli Biham, Orr Dunkelman, Nathan Keller.** New Cryptanalytic Results on IDEA. *Nathan Keller homepage.* [Online] 2006. [Citace: 01. 04 2008.] New Cryptanalytic Results on IDEA.
[30] **Klíma, Vlastimil.** Šifra pro třetí tisícíletí. *CHIP.* 2002, 05.
[31] **Smith, Warren D.** 1. AES seems weak. 2. Linear time secure cryptography. [Online] 08. 06 2007. [Citace: 22. 12 2007.] http://eprint.iacr.org/2007/248.
[32] **Rivest, Ron.** A Description of the RC2(r) Encryption Algorithm. *Request for Comments.* [Online] 1998. [Citace: 25. 12 2007.] http://tools.ietf.org/html/rfc2268.
[33] **Ronald Rivest, Vincent Rijmen, Lars Knudsen.** On the Design and Security of RC2. *http://citeseer.ist.psu.edu.* [Online] 1998. [Citace: 22. 12 2007.] citeseer.ist.psu.edu/knudsen98design.html.
[34] **Anonym.** *RC4 Source Code.* [Email] 1984. http://cypherpunks.venona.com/date/1994/09/msg00304.html.
[35] **Alexander L. Grosul, Dan S. Wallach.** A Related-Key Cryptanalysis of RC4. [Online] 06. 06 2000. [Citace: 26. 12 2007.] http://www.wisdom.weizmann.ac.il/~itsik/RC4/Papers/GrosulWallach.ps.
[36] *Šifra, která míchá karty.* **Klíma, Vlastimil.** 09, Praha : CHIP, 1999.
[37] **Erik Tews, Ralf-Philipp Weinmann, Andrei Pyshkin.** Breaking 104 bit WEP in less than 60 seconds. [Online] 04. 04 2007. [Citace: 15. 04 2008.] http://eprint.iacr.org/2007/120.pdf.

[38] **Johan Borst, Bart Preneel, Joos Vandewalle.** *Linear Cryptanalysis of RC5 and RC6.* Heverlee : Dept. Elektrotechniek-ESAT/COSIC.

[39] **Eli Biham, Orr Dunkelman, Nathan Keller.** *Linear Cryptanalysis of Reduced Round Serpent.* [PDF Document] Haifa : Computer Science department, Technion, 2001.

[40] **Jung Hee Cheon and MunJu Kim, Kwangjo Kim, Jung-Yeun Lee, SungWoo Kang.** Improved Impossible Differential Cryptanalysis of Rijndael and Crypton. *citeseer.ist.psu.edu.* [Online] 2001. [Citace: 22. 12 2007.] http://citeseer.ist.psu.edu/501105.html.

[41] **Schneier, Bruce.** Description of a New Variable-Length Key, 64-Bit Block Cipher. *Fast Software Encryption, Cambridge Security Workshop Proceedings.* [Online] 12 1993. [Citace: 4. 1 2008.] http://www.schneier.com/paper-blowfish-fse.html.

[42] **Orhun Kara, Cevat Manap.** A New Class of Weak Keys for Blowfish, p. 167-180. [autor knihy] Cevat Manap Orhun Kara. *Fast Software Encryption.*: Springer Berlin / Heidelberg, 2007.

[43] **Shino Moriai, Lisa Yin.** Cryptanalysis of Twofish (II). [Online] 2000. [Citace: 06. 01 2008.] http://www.schneier.com/twofish-analysis-shiho.pdf.

[44] **Jing-mei Liu, Xiang-guo Cheng, Xin-mei Wang.** *Methods to Forge ElGamal Signatures and Determine Secret Key.* Washington, DC : IEEE Computer Society, 2006.

[45] **Briggs, Matthew E.** *An Introduction to the General Number Field Sieve.* [PDF Document] Virginia : Virginia Polytechnic Institute and State University, 1998.

[46] **Shor, Peter W.** Polynomial-Time Algorithms for Prime Factor Factorization and Discrete Logarithms on a Quantum Computer. *Quant-ph - arxiv.* 1996, 9508027.

[47] **Rosa, Tomáš.** Od bitů ke qubitům. *CHIP.* 2002, 05.

[48] DIGITAL SIGNATURE STANDARD (DSS). [Online] 19. 05 1994. [Citace: 25. 01 2008.] http://www.itl.nist.gov/fipspubs/fip186.htm.

[49] **Weisstein, Eric.** Pollard rho Factorization Method. *MathWorld - A Wolfram Web Resource.* [Online] MathWorld. [Citace: 15. 02 2008.] http://mathworld.wolfram.com/PollardRhoFactorizationMethod.html.

[50] **Marcel Medwed, Elisabeth Oswald.** Template Attacks on ECDSA. *eprint.iacr.org.* [Online] 02 2008. [Citace: 28. 02 2008.] http://eprint.iacr.org/2008/081.pdf.

[51] **Eli BIham, Rafi Chen.** *Near-Collisions of SHA-0.* [PDF Document] Haifa, Israel : Israel Institute of Technology, 2004. http://citeseer.ist.psu.edu/cache/papers/cs/31691/http:zSzzSzwww.cs.technion.ac.ilzSzuserszSzwwwbzSzcgi-binzSztr-get.cgizSz2004zSzCSzSzCS-2004-09.pdf/near-collisions-of-sha.pdf.

[52] **Muller, Frédéric.**: *The MD2 Hash Function is Not One-Way.* ASIACRYPT, 2004.

[53]. **RSA LABORATORIES**, RSA BULLETIN #4.: strana 3, online: 20.12.2007. ftp://ftp.rsasecurity.com/pub/pdfs/bulletn4.pdf.

[54] **John E. Mathiassen Lars Knudsen.** Preimage and Collision Attacks on MD2. *Fast Software Encryption*: Springer Berlin / Heidelberg, 2005.

[55] **Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, Xiuyuan Yu.** *Cryptanalysis of the Hash Functions MD4 and RIPEMD.*: Cryptology ePrint Archive, 2004. Report 2004/199.
[56] **Xiaoyun Wang, Dengguo Feng, Xuejia Lai, Hongbo Yu.** *Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD.* [PDF Document] Jinan : The School of Mathematics and System Science, Shandong University, China, 2004. http://eprint.iacr.org/2004/199.pdf.
[57] **Vondruška, Pavel.** Nové útoky na hashovací funkce SHA-1 a MD5. *www.root.cz.* [Online] 15. 11 2005. [Citace: 31. 11 2007.] http://www.root.cz/zpravicky/nove-utoky-na-hashovaci-funkce-sha-1-a-md5/.
[58] **Yu Sasaki, Yusuke Naito, Noboru Kunihiro, Kazuo Ohta.** *Improved Collision Attack on MD5.* [PDF Document] místo neznámé : The University of Electro-Communications, Japan, 2005.
[59] **Klíma, Vlastimil.** Tunely v hašovacích funkcích: kolize MD5 do minuty. *http://cryptography.hyperlink.cz.* [Online] 18. 03 2006. [Citace: 20. 12 2007.] http://cryptography.hyperlink.cz/2006/tunely.pdf.
[60] **James Randall, Michael Szydlo.** Collisions for SHA0, MD5, HAVAL, MD4, and RIPEMD, but SHA1 Still Secure. *www.rsa.com.* [Online] 31. 08 2004. [Citace: 30. 11 2007.] http://www.rsa.com/rsalabs/node.asp?id=2738.
[61] **Szydlo, Michael.** http://www.rsa.com/rsalabs/node.asp?id=2927. *RSA.com.* [Online] 19. 08 2005. [Citace: 01. 12 2007.] http://www.rsa.com/rsalabs/node.asp?id=2927.
[62] **Somitra Kumar Sanadhya, Palash Sarkar.** *New Local Collisions for the SHA-2 Hash Family.* [PDF Document] Kolkata : Applied Statistics Unit Indian Statistical Institute, 2007.
[63] **Henri Gilbert, Helena Handschuh** *Security analysis of SHA-256 and sisters.*: Springer, 2003.
[64] **P. Hawkes, M. Paddon, and G.G. Rose.** On Corrective Patterns for the SHA-2. [Online] 2004. [Citace: 02. 04 2008.] eprint.iacr.org/2004/207.pdf .
[65] **John Kelsey, Stefan Lucks.** *Collisions and Near-Collisions for Reduced-Round Tiger.* Mannheim : Springer LNCS, 2006.
[66] **Bellare, Mihir.** *New Proofs for NMAC and HMAC Security without Collision-Resistance.* [PDF Document] San Diego : Dept. of Computer Science & Engineering University of California San Diego, 2006.
[67] **Baretto, Paulo.** The Whirlpool hash function. *The Whirlpool hash function.* [Online] 28. 10 2006. [Citace: 15. 1 2008.] http://paginas.terra.com.br/informatica/paulobarreto/WhirlpoolPage.html.
[68] **Dai, Wei.** Speed Comparsion of Popular Crypto Algorithms. *CRYPTO++.* [Online] 5. 5 2007. [Citace: 17. 1 2008.] http://www.cryptopp.com/benchmarks.html.
[69] **Lenstra, Arjen K.** *Selecting Cryptographic Key Sizes.* [PDF Document] North Gate Road : Citibank, N.A., and Technische Universiteit Eindhoven,PricewaterhouseCoopers, GRMS Crypto Group, 2000.
[70] **NIST** Recommendation for Key Management. *www.nist.gov.* [Online] 03 2007. [Citace: 2007. 12 17.] http://csrc.nist.gov/groups/ST/toolkit/index.html.

[71] **Christian Gehrmann, Mats Naslund.** *ECRYPT Yearly Report on Algorithms and Keysizes.* [PDF Document] Leuven : European Network of Excellence in Cryptology, 2007.

[72] *Mécanismes cryptographiques - Règles et recommandations.* [PDF Document] Paris : DCSSI, 2006.

[73] **John Kelsey, Bruce Schneier, David Wagner, Chris Hall.** *Cryptanalytic Attacks on Pseudorandom Number Generators.* [PDF] Berkeley : University of California Berkeley, 1998. http://www.schneier.com/paper-prngs.pdf.

[74] **Junod, Pascal.** The Blum-Blum-Shub Generator. [Online] 08 1999. [Citace: 27. 12 2007.] http://crypto.junod.info/bbs.pdf.

[75] **Bruce Schneier, Niels Fergusson.** *Practical Cryptography.* Wiley & Sons, 2003. ISBN: 978-0471223573.

[76] **Pisharath, Jagannath.** Linear Congruential Number Generators. *www.math.rutgers.edu* . [Online] 2003. [Citace: 26. 12 2007.] http://www.math.rutgers.edu/~greenfie/currentcourses/sem090/pdfstuff/jp.pdf.

[77] **Dostálek Libor, RNDr.** Certifikační autorita. *Velký průvodce protokoly TCP/IP: bezpečnost.* Brno : Computer Press, 2001. ISBN: 80-7226-849-X

[78] **Rosa, Tomáš.** Nepopíratelnost digitálních podpisů. *Personal page: Dr. Tomáš Rosa.* [Online] [Citace: 4. 04 2008.] http://crypto.hyperlink.cz/files/rosa_ZMVS04.pdf.

[79] **Dostálek Libor, RNDr.** Autentizace uživatele a autorizace dat. [autor knihy] Libor, RNDr. Dostálek. *Velký průvodce protokoly TCP/IP: bezpečnost.* Brno : Computer Press, 2001. ISBN: 80-7226-849-X

[80] **Haller, Neil M.** THE S/KEY ONE-TIME PASSWORD SYSTEM. *Department of Computer Science, University of Tennessee at Knoxville.* [Online] 1994. [Citace: 25. 03 2008.] http://www.cs.utk.edu/~dunigan/cns04/skey.pdf.

[81] **Alex Biryukov, Joseph Lano,Bart Preneel.** Cryptanalysis of the Alleged SecurID Hash Function (extended version). *Cryptology ePrint Archive.* [Online] 2003. [Citace: 1. 4 2008.] http://eprint.iacr.org/2003/162.pdf.

[82] **Young, Matthew.** BioAPI Consortium. *BioAPI Consortium.* [Online] 06 2005. [Citace: 26. 03 2008.] http://www.bioapi.org/history.asp.

[83] **B. Lloyd, W. Simpson.** RFC 1334. *www.networksorcery.com.* [Online] 10 1992. [Citace: 15. 03 2008.] http://www.networksorcery.com/enp/rfc/rfc1334.txt.

[84] **Simpson, W.** Internet Engineering Task Force (IETF). *RFC 1994* . [Online] 08 1994. [Citace: 15. 03 2008.] http://tools.ietf.org/html/rfc1994.

[85] **Bruce Schneier, David Wagner, Mudge.** www.schneier.com. *Schneier.com.* [Online] 19. 10 1999. [Citace: 16. 03 2008.] http://www.schneier.com/paper-pptpv2.pdf.

[86] **B. Aboba, D. Simon.** PPP EAP TLS Authentication Protocol. *IETF Tools.* [Online] The Internet Society, Microsoft, 10 1999. [Citace: 20. 03 2008.] http://tools.ietf.org/html/rfc2716.

[87]. **Wright, Joshua.** Asleap - exploiting cisco leap. *Joshua Wright home page.* [Online] 13. 07 2007. [Citace: 17. 03 2008.] http://www.willhackforsushi.com/Asleap.html.
[88] **Vivek Kamath, Ashwin Palekar, Mark Wodrich.** Microsoft's PEAP version 0. *The Internet Society.* [Online] 25. 10 2002. [Citace: 23. 04 2008.] http://tools.ietf.org/id/draft-kamath-pppext-peapv0-00.txt.
[89] **Ou, George.** Cisco confirms vulnerability in 7921 Wi-Fi IP phone. *ZDNET.COM.* [Online] ZDNET.COM, 23. 02 2008. [Citace: 20. 03 2008.] http://blogs.zdnet.com/security/?p=901.
[90] **CISCO.** Vulnerability in Cisco Secure Access Control Server EAP−TLS Authentication. *www.cisco.com.* [Online] 02. 11 2004. [Citace: 21. 03 2008.] http://www.cisco.com/warp/public/707/cisco-sa-20041102-acs-eap-tls.pdf.
[91] **H. Tschofenig, et al.** The Extensible Authentication Protocol-Internet Key Exchange Protocol version 2 (EAP-IKEv2) Method. [Online] 01 2008. [Citace: 02. 04 2008.] http://tools.ietf.org/html/rfc5106.
[92] **Paul Funk, Simon Blake-Wilson.** EAP Tunneled TLS Authentication Protocol Version 0. *Network Working Group.* [Online] 04 2008. [Citace: 30. 04 2008.] http://tools.ietf.org/html/draft-funk-eap-ttls-v0-05.
[93] **C. Neuman, S. Hartman, K. Raeburn.** The Kerberos Network Authentication Service (V5). *IETF Tools.* [Online] The Internet Society, 07 2005. [Citace: 21. 03 2008.] http://tools.ietf.org/html/rfc4120.
[94] **S. M. Bellovin, M. Merritt.** Limitations of the kerberos authentication system. *nist.gov.* [Online] National Institute of Standards and Technology, 1990. [Citace: 20. 03 2008.] http://csrc.nist.gov/publications/secpubs/kerblim.ps.
[95] **Hill, Joshua.** An Analysis of the RADIUS Authentication Protocol. *Joshua Hill's homepage.* [Online] 2001. [Citace: 30. 03 2008.] http://www.untruth.org/~josh/security/radius/radius-auth.html.
[96] **Bitto, Ondřej.** Ukryto pod rouškou X.509. [Online] 06. 10 2005. [Citace: 20. 01 2008.] http://www.lupa.cz/clanky/ukryto-pod-rouskou-x-509/.
[97] **Gietz, Peter.** An LDAP/X.500 based distributed PGP Keyserver. *DAASI International.* [Online] 22. 05 2000. [Citace: 30. 03 2008.] http://www.daasi.de/pub/pg2000-018-PGPkeyserver.pdf.
[98] **Arjen Lenstra, Xiaoyun Wang, Benne de Weger.** *Colliding X.509 Certificates.* [PDF Dokument] místo neznámé : Lucent Technologies, Bell Laboratories, 2005. http://eprint.iacr.org/2005/067.pdf.
[99]. **Mureed Hussain, Ahmed Mehaqua, Dominique Seret.** VPN, Secure Socket Layer, Transport Layer Security. *Ahmed Mehaoua home page.* [Online] 2008. [Citace: 13. 04 2008.] http://www.math-info.univ-paris5.fr/~mea/cours/DU/TLS_SSL_DUsec.pdf.
[100] **Bruce Schneier, David Wagner.** Analysis of the SSL 3.0 protocol. *Bruce Schneier's page.* [Online] 11 1996. [Citace: 18. 04 2008.] http://www.schneier.com/paper-ssl.pdf.

[101] **Weith, Loren.** Welcome to yaksman.org. *Differences Between SSLv2, SSLv3, and TLS.* [Online] 03. 06 2006. [Citace: 20. 04 2008.] http://www.yaksman.org/~lweith/ssl.pdf.
[102] **Eric Rescorla, .** TLS WG. *TLS WG.* [Online] 2008. [Citace: 20. 04 2008.] http://www.ietf.org/proceedings/06nov/slides/tls-0.pdf.
[103] **Niels Ferguson, Bruce Schneier.** www.schneier.com. *A Cryptographic Evaluation of IPsec.* [Online] [Citace: 02. 04 2008.] http://www.schneier.com/paper-ipsec.pdf.
[104] **Dan Harkins, Charlie Kaufman, Radia Perlman.** Overview of IKEv2. *The Internet Engineering Task Force.* [Online] [Citace: 01. 04 2008.] http://www.ietf.org/proceedings/01dec/slides/ipsec-10.pdf.
[105] **B. Schneier, Mudge.** Cryptanalysis of Microsoft's Point-to-Point Tunneling Protocol (PPTP). *Bruce Schneier's website.* [Online] [Citace: 03. 04 2008.] http://www.schneier.com/paper-pptp.pdf.
[106] **Wafaa Bou Diab, Samir Tohme, Carole Bassil.** Critical VPN Security Analysis and New Approach for Securing VoIP Communications over VPN Networks. *http://delivery.acm.org/.* [Online] 2007. [Citace: 04. 04 2008.] http://delivery.acm.org/10.1145/1300000/1298238/p92-boudiab.pdf.
[107] **Davies, Joseph.** The Cable Guy The Secure Socket Tunneling Protocol. *TechNet Magazine.* [Online] [Citace: 10. 04 2008.] http://technet.microsoft.com/en-us/magazine/cc162322.aspx.
[108] **Petřík, Michal.** Technologie MPLS. *Distribuované systémy a sítě.* [Online] [Citace: 11. 04 2008.] https://dsn.felk.cvut.cz/wiki/_media/vyuka/cviceni/x36mti/prezentace2007/petrim2-doc.pdf.
[109] **Pultz, J.** MPLS Security Analysis. *Cisco book.* [Online] [Citace: 15. 04 2008.] http://searchnetworking.techtarget.com/searchNetworking/Downloads/CCNP_BCMSN_ch3.pdf.
[110] **Hosner, Charlie.** OpenVPN and the SSL VPN Revolution. [Online] 08. 08 2004. [Citace: 28. 04 2008.] http://www.ispl.jp/~oosaki/lecture/isn-design/2007/18_Hosner_OpenVPN.pdf.
[111] **V., Klíma.** Co se stalo s hašovacími funkcemi?, část 2. *Crypto-World.* 2005, 4.
[112] —. Co se stalo s hašovacími funkcemi?, část 1. *Crypto-World.* 2005, 3.