Kryptografie

Jan Paseka

11. února 1998

Úvod

Od chvíle co existují tvorové obdaření řečí, existují i důvěrná sdělení, jež jsou určena jen pro jedinou osobu nebo jen pro zcela určitý okruh lidí, přičemž pro ostatní osoby mají být nedostupná.

Jakým způsobem lze „bezpečně“ předat zprávu, tedy tak, že *nikdo nežádoucí nezjistí obsah této zprávy*? S tím pak souvisí téměř ještě důležitější otázka: Jak můžeme dosáhnout toho, aby zpráva skutečně dorazila k příjemci a sice *právě v tom stavu, jak byla odeslána?*

Obvykle jsou dvě možnosti, jak tyto problémy vyřešit. První z nich je zamlčení *existence* zprávy. Mohli bychom napsat důvěrnou zprávu například pomocí neviditelného inkoustu nebo ji poslat pomocí důvěryhodné osoby. O toto se ve všech dobách pokoušeli tajně zamilovaní – a téměř všechny klasické tragédie svědčí a selhání těchto snah.

Jiná možnost spočívá v *zakódování* důvěrné zprávy. V tomto případě se neutajuje její existence. Naopak: Zpráva se předá pomocí nejistého kanálu, ale tak „zašifrovaná“, aby nikdo – mimo skutečného příjemce – nemohl zprávu „rozšifrovat“. Zde se jedná o vědomě proradnou výzvu soupeři; takovéto výzvy bývají zpravidla také přijaty – a ne zřídka se hra obrátí.

Dále je zajímavý problém *integrity* a *autentičnosti* dat. Zde se jedná o to, aby zpráva byla chráněna proti neoprávněné změně.

Až do nedávna byly armáda a tajné služby jediný subjekt, který se profesionálně zabýval s takovýmito systémy utajení. Pouze v tomto odvětví byla dostatečná motivace – a odpovídající peněžní prostředky – na vyvinutí šifrovacích strojů, což byly chytré mechanické zázraky.

Skutečnost, že se kryptologie účastnila zrodu moderních počítačů, má symbolický charakter. S ohromným rozšířením elektronického zpracování dat se kryptologie postavila na nový základ. To má různé příčiny a to:

- Při pokusu o prolomení nepřátelského systému se musí zpracovat obrovská množství dat (řetězce písmen, sloupce cifer); musí se srovnat data, spočítat průměrné hodnoty, standartní odchylky a mnoho jiného – to všechno jsou věci, které počítač zvládne mnohem rychleji a lépe než člověk. Důsledkem je pak, že kryptosystémy, které se v současnosti používají, musí být podstatně komplexnější než jejich předchůdci před dvěma generacemi.

- Z druhé strany umožňuje moderní hardware a software implementaci komplexních a náročných matematických algoritmů. Pomocí těchto algoritmů lze dosáhnout takového stupně jistoty, který v historii nemá obdoby: malý přírůstek komplexity algoritmu vede k obrovskému nárůstu zdrojů nutných k prolomení systému. Vtip moderní kryptologie spočívá v tom, že počítač není jenom příčinou mnoha problémů, nýbrž i klíčem k jejich řešení.

- Vstupem elektronického zpracování dat a obzvláště elektronické komunikace do stále více odvětví se otevírají zcela nová pole působnosti pro kryptologii. Vedle „klasického“ vojenského použití nastupují na kryptologii zcela nové požadavky. Není nutné mít mnoho věšteckého nadání k předpovědi, že kryptologie (která se nyní stává seriózní vědou) zažije v příštích letech další razantní rozvoj.

Z novodobých problémů kryptologie vyjmenujme následující:

- Mnoho telefonních hovorů je v současnosti zprostředkováno přes satelit. Tímto je umožněno, že v principu kdokoliv je schopen tyto hovory odposlouchávat. Tedy alespoň tajné telefonní hovory je nutno tak šifrovat, že odposlouchavač slyší pouze slátaninu tónů.

- Podobný problém se týká placené televize. Zde spočívá problém v tom, že se neautorizovaný uživatel chce podívat na svoje oblíbené filmy, aniž by za to platil. Pomocí prostředků na rozpoznání autentičnosti uživatele se tomuto lehce zabrání.

- V stále rostoucí míře se provádí převody peněz elektronicky („Homebanking, electronic cash“). Zde je nutná elektronická náhrada obvyklého podpisu vlastní rukou. Přitom tzv. elektronický podpis je v mnohém směru lepší než obvyklé podepsání.

- Většina nynějších středních a větších počítačů je tak uzpůsobena, že mnoho uživatelů může navzájem nezávisle pracovat s počítačem (multiuser systémy). V takovýchto situacích se počítač musí přesvědčit o identitě uživatele. V současné době se to děje pomocí hesla; v budoucnosti toto bude u situací souvisících s obzvláštní bezpečností nahrazeno „čipovou kartou“.

- Zároveň se můžeme zmínit o „počítačových virech“. To jsou programy, které prakticky nepozorovaně vniknou do počítačového programu; mají schopnost se samostatně reprodukovat a to je důvodem k vzniku velkých škod na programovém, datovém i hardwarowém vybavení počítače. Zhruba řečeno, virus změní svůj „hostitelský program“; lze tedy použít s úspěchem k rozpoznání viru metodu autentičnosti programu.

Každý, kdo má něco společného s těmito nebo jim podobnými aplikacemi, bude souhlasit: „Ovšemže potřebujeme bezpečnost! Ale – proč má být kryptologie všelék? Nejsou k dispozici jiné metody, abychom získali potřebnou jistotu?“ Přirozeně, jsou jiné metody! Pomyslete na techniky vyvíjené po staletí, aby byly naše bankovky bezpečné proti falšování: speciální papír, komplexní (mnohdy dokonce krásné) obrazy, perfektní tisk, vodoznak a mnoho jiného. Tedy ještě jednou: Proč kryptologie?

Odpověď je jednoduchá: *Kryptologie je lepší!* Důvodem proto je: *Kryptologie je matematická disciplína.* To může znít nadneseně, ale není tomu tak: Matematika poskytuje – alespoň v principu – teoretické odůvodnění pro sílu algoritmu nebo šifrovacího protokolu. S matematikou lze (v ideálním případě) *dokázat*, že kryptografický algoritmus má jistý stupeň bezpečnosti. A jakmile je jednou bezpečnost algoritmu jednou matematicky dokázána, není žádných pochyb o tom, že algoritmus je skutečně bezpečný; nemusíme se pak spoléhat na (zpravidla rozporuplné) posudky expertů, nepotřebujeme se odvolávat při posuzování bezpečnosti na „technologii dneška“ (která může být zítra úplně jiná), atd.

Je však nutno přiznat, že takovéto důkazy se podařily jen ve velmi málo případech. Avšak přesto: matematika je důvěrohodný nástroj pro systematické zkoumání kryptosystémů (tzn. návrh a analýza). To je důvod, proč dáváme kryptologickým mechanismům v případě pochyb přednost před jinými bezpečnostními systémy: *In dubio pro mathematica*

Věda, která se zabývá se všemi těmito problémy (a mnoha dalšími) se nazývá **kryptologie** nebo také **kryptografie**. Jí jsou věnovány následující stránky tohoto učebního textu, který si neklade žádné nároky na úplnost či původnost. Případné komentáře či kritické připomínky k textu očekávám nejlépe na e-mailové adrese

`paseka@math.muni.cz`

či jinou formou. Text je průběžně doplňován a měněn a je umístěn k volnému použití na ftp serveru oboru matematika PřF MU. Části textu jsou tvořeny referáty či obrázky a grafy zpracovanými studenty M. Kučerová, M. Misáková, J. Mráka, A. Rozsypal, R. Sedláček, Svitel a E. Žáčková v rámci stejnomenné přednášky na Přírodovědecké fakultě Masarykovy univerzity. Veškerá zodpovědnost za styl a obsah však padá na moji hlavu.

Obsah

Kapitola 1

Caesar neboli Každý začátek je lehký!

Nun-e-zuz-o-fuf-e-juj! Bub-u-dud-e-šuš o-sus-vuv-o-bub-o-zuz-e-nun!
Pup-o-zuz-o-rur! Tut-o juj-e tut-e-nun vuv-rur-a-huh!
Astrid Lingrenová

Při každém zakódování musí být příjemce vždy o něco před útočníkem. S pomocí této informace může příjemce zprávu rozšifrovat; tato informace nesmí být žádnému útočníkovi k dispozici, neboť by útočník byl schopen rozluštit zprávu zrovna tak lehce jako příjemce. O této exkluzivní informaci mluvíme jako o **klíči**. Klasické šifrovací metody jsou založeny na tom základě, že odesilatel a příjemce mají společný šifrovací klíč, se kterým odesilatel zprávu zašifruje a příjemce ji rozšifruje; takováto metoda se nazývá **symetrická**. V dalším uvidíme, že existují také asymetrické šifrovací algoritmy: v těchto systémech potřebuje pouze příjemce tajný klíč.

V této kapitole se budeme zabývat v jistém slova smyslu pouze těmi nejjednoduššími šifrovacími algoritmy a to takovými, že v nich je jedno a totéž písmeno nahrazeno jedním a tímtéž symbolem. Například písmeno **e** obsažené v textu by se zašifrovalo pomocí písmena **K**.

*

Nejdříve několik slov k terminologii. Pojmy **kryptologie** a **kryptografie** pochází z řeckých slov $\kappa\rho\upsilon\pi\tau o\sigma$ (tajný) a $\lambda o\gamma o\sigma$ (slovo, smysl) a $\gamma\rho\alpha\varphi\varepsilon\iota\nu$ (psát).Obě slova označují umění a vědu, která se zabývá rozvojem metod k utajení zpráv. (Mnozí autoři rozlišují mezi **kryptografií**, tj. vědou o vývoji kryptosystémů a **kryptoanalýzou**, uměním tyto kryptosystémy a prolomit a označují slovem **kryptologie**, spojení těchto věd.)

Text, řetězec znaků nebo písmen, který chceme zprostředkovat se nazývá **zpráva**; obvykle budeme zprávu reprezentovat pomocí malých písmen **a, b, c,**. . ..

Zašifrovanou zprávu budeme nazývat **kryptogram**; tento pak budeme psát pomocí velkých písmen **A, B, C,**.... Šifrovací postup nazýváme **šifrování**, odšifrovací postup **dešifrování**. Odesilatel tedy šifruje, zatímco příjemce musí dešifrovat, aby si mohl přečíst zprávu.

Texty, které budeme šifrovat, se skládají z **písmen**; tato písmena jsou prvky nějaké **abecedy**. V prvních dvou kapitolách bude obvykle naše abeceda přirozená **abeceda** {**a, b, c,**...}. Budeme také vybírat za abecedu např. množinu $\{1, \ldots, 26\}$, množinu $\{0, 1\}$ nebo také množinu $\{(a_1, \ldots, a_{64}) : a_i \in \{0, 1\}\}$ všech binárních posloupností délky 64 v případě, že to bude pro naše úvahy vhodné.

V rozporu s nadpisem kapitoly začínají dějiny kryptografie *před* Caesarem.

1.1 Spartská skytála

Historie začíná asi před 2500 lety. Jak dobře víme z díla řeckého dějepisce Plutarcha, používala vláda ve Spartě následující lstivou metodu pro přenos tajné zprávy pro své generály: odesilatel a příjemce museli mít oba tzv. **skytálu**: byly to dva válce o přesně stejném průměru. Odesilatel navinul úzkou pergamenovou pásku spirálovitě okolo své skytály a napsal pak podle délky svou zprávu na pásku. Po odmotání pásky mohla zprávu číst jen ta osoba, která měla skytálu stejného rozměru – doufejme, že to byl pouze příjemce.

Uvažujme nyní příklad převedený do moderního jazyka. Představme si, že jsme zachytili list papíru, na kterém čteme následující řetězec písmen:

UNDTLATEDZEEIOVEMEJKSSMYNZ.EOI
IELAENLTCTENLOIEKRZOAMKKIUENN

Skytála odesílatele má průměr, který můžeme vyjádřit pomocí počtu písmen. Můžeme tedy jednoduše vyzkoušet různé rozsahy u. Zvolíme-li $u = 7$, dostaneme následující nesmysl:

U	E	V	S	O	N	L	O	E
N	D	E	M	I	L	O	A	V
D	Z	M	Y	I	T	I	M	N
T	E	E	N	E	C	E	K	
L	E	J	Z	L	T	K	K	
A	I	K	.	A	E	R	I	
T	O	S	E	E	N	Z	U	,

zvolíme-li ale uspořádání textu pro $u = 9$, dostaneme:

U	Z	J	E	N	O	M
N	E	K	O	L	I	K
D	E	S	I	T	E	K
T	I	S	I	C	K	I
L	O	M	E	T	R	U
A	V	Y	L	E	Z	E
T	E	N	A	N	O	V
E	M	Z	E	L	A	N
D	E	.				

Skytála je prototyp **transpozičního algoritmu**; přitom písmena zůstávají, *co* jsou, nezůstanou však, *kde* jsou. Nejedná se o nic jiného, než o permutaci *míst* písmen.

Transpoziční algoritmy jsou důležitým stavebním kamenem pro moderní algoritmy. Jinou složkou jsou **substituční algoritmy**; u nich se zpráva stane nečitelnou tím, že každé písmeno se nahradí jiným, ale jeho pozice zůstane zachována.

A nyní nastává doba pro vstup Caesara na scénu.

1.2 Posouvací šifry

Jeden z prvních, kdo používal kryptologické techniky, byl římský vojevůdce a státník Gaius Julius **Caesar**(100-44 př. n. l.). U Suetona (Caes. LVI) můžeme číst

> Exstant et [epistolae] ad Ciceronem, item ad familiares de rebus, in quibus, si qua occultius perferenda erant, per notas scripsit, id est sic structo litterarum ordine, ut nullum verbum effici posset; quae si qui investigare et persequi velit, quartam elementorum litteram, id est D pro A et perinde reliquas commutet.

Překlad zní přibližně následovně

> Existují také [Caesarovy dopisy] Cicerovi a známým o věcech, v kterých psal tajným písmem, pokud něco muselo být důvěrně sděleno. Tzn. změnil pořadí písmen tak, že nešlo zjistit jediné slovo. Pokud někdo chtěl toto rozluštit o poznat obsah, musel dosadit čtvrté písmeno abecedy, tedy D, za A, a podobně toto provést se zbývajícími písmeny.

Šifru použitou Caesarem obdržíme tím způsobem, že místo abecedy zprávy budeme psát abecedu kryptogramu, ale o 23 míst doprava, což znamená totéž, jako posunutí doleva o 3 místa:

Zpráva:	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
Kryptogram:	ABCDEFGHIJKLMNOPQRSTUVWXYZ.

Šifruje se tím způsobem, že nahradíme písmeno zprávy pod ním stojícím písmenem kryptogramu. Například ze slova „**zprava**“se stane zdánlivě nesmyslné slovo „**CSUDYD**“. Dešifrování je zrovna tak jednoduché: Každé písmeno kryptogramu se nahradí nad ním stojícím písmenem zprávy.

Člověk se ovšem může ptát, proč Caesar zvolil právě posunutí o 3 místa. Odpověď je jednoduchá: nebyl na to vůbec žádný důvod! Samozřejmě můžeme posunout abecedu o libovolný počet míst. Protože se naše abeceda sestává z 26 písmen, existuje právě 26 takových šifrování; mluvíme o **posouvacích** neboli **aditivních šifrách** a mezi nimi je samozřejmě **triviální šifrování a $\rightarrow$ A, b $\rightarrow$ B, ..., z $\rightarrow$ Z**, které samozřejmě nikdo nebude používat k šifrování tajných zpráv.

Vyjasněme si na této nejjednodušší třídě šifer pojmy „šifrovací algoritmus“ a „klíč“. Šifrovací algoritmus je bezprostředně vidět na šifrování slova zprava. Naproti tomu klíč je např. počet míst, o která je nutno posunout abecedu. Klíč nám přesně popisuje, jak se algoritmus použije ve speciální situaci.

Partneři, kteří spolu komunikují, se musí dohodnout o šifrovacím algoritmu a o způsobu přenosu klíče. Šifrovací algoritmus a klíč mají dvě zcela rozličné funkce a musí být proto zcela jasně rozlišitelné. Algoritmus jako takový je zpravidla velmi „velký“. Přitom mnoho algoritmů se realizuje pomocí mechanického zařízení nebo spočívá na více či méně veřejně přístupném postupu. Z toho plyne, že algoritmus nelze v podstatě udržet v tajnosti. To pak znamená, že *celková bezpečnost kryptosystému leží na utajení klíče.*

Tento požadavek se zdá být přehnaný, je ale nanejvýš realistický: pro někoho, kdo chce neoprávněně číst naši zprávu, je srovnatelně lehké získat algoritmus (např. odcizit či zkopírovat přístroj). A pak tento zlosyn ví vše o algoritmu; nezná ale (doufejme) současný klíč.

Z toho nutně plyne, že klíč je nutno předat bezpečnou cestou. K čemu pak vůbec šifrování zprávy? V tom případě jsme mohli hned přenést celou zprávu touto bezpečnou cestou! – Tato námitka je plně oprávněná, lze ji však následujícími argumenty podstatně zmírnit.

1. Zpráva bývá zpravidla velmi dlouhá, klíč se obvykle volí co nejkratší. Dodatečná námaha pro spolehlivý přenos klíče je pak silně redukovatelná. Proto je pravděpodobnost, že klíč bude odposloucháván, relativně malá.

2. Odesilatel a příjemce si mohou libovolně vybrat dobu předání klíče. Klíč lze například dohodnout dny před přenosem zprávy. Naproti tomu se zpráva musí odeslat v okamžiku, který není ovlivnitelný komunikujícími partnery (uvažme politické události, vývoj na burze, atd.)

3. S pomocí tzv. Public-Key systémů lze klíče bez nebezpečí vyměnit, abychom pak provedli zakódování s pomocí konvenčních postupů.

Upozorněme ještě na další nebezpečí. Je-li klíč vyměněn, musí být spolehlivě uložen; nesmí nastat případ, že jej bude možno z přístroje zjistit. Experti souhlasí s tím, že klíč je pouze tehdy bezpečně uložen, pokud přístroj nelze najít fyzikálními prostředky.

Nyní ale nastala doba na změnu stran. Kryptologie se nezabývá pouze tím, že navrhuje bezpečnostní systémy pro přenos zpráv; jedna z jejich ústředních úloh je takové systémy „rozluštit“ (nebo se o to alespoň pokusit!).

Začněme tedy na okamžik hrát roli zlosyna – to lze vyjádřit trochu slušněji následovně: pracujeme jako **kryptoanalytik** a provádíme **kryptoanalýzu** kryptogramu (případně zkoumaného kryptosystému). Tvůrce kryptosystému musí vždy počítat s možností, že algoritmus je protivníkovi znám (alespoň po delší dobu). Kromě toho se doporučuje protivníka nepodceňovat a přisoudit mu co nejvyšší inteligenci; nazvěme je **Mr. X**.

Představme si, že Mr. X zachytil následující kryptogram:

BIBV HXZIDI CH VMVQ BIRVI

Na základě jistých indicií došel k domněnce, že tento text byl zašifrován pomocí posouvací šifry (např. by mohl „najít“ jeden z výše popsaných šifrovacích strojů). Takovýto text pak lze principiálně analyzovat dvěma způsoby.

1. *„Systematické“ prozkoušení všech možností*

Protože se jedná pouze o 26 posunutí, není naše námaha příliš velká. Mr. X ale může tuto námahu podstatně zredukovat, omezí-li se pouze na malou část zachyceného textu.

Uvažme např. „slovo“ **VMVQ**. Vyzkoušíme-li všechna „posunutí“ této posloupnosti písmen, zjistíme snadno, že ze všech možných ekvivalentů pouze slovo **neni** dává smysl. Je tedy více než pravděpodobné, že kryptogram byl získán posunutím o 8 míst. Mr. X pak prověří svou domněnku tím, že dešifruje celkový text:

tato zprava uz neni tajna.

Tato metoda pro prolomení posunovací šifry je proto tak dobrá, protože většina kombinací písmen je v češtině zcela bez významu. Ačkoliv je toto pozorování důležitý základ pro mnoho kryptoanalytických metod, má výše uvedená metoda velkou nevýhodu. Nelze ji totiž (nebo jen s neúměrně velkou námahou) automatizovat. Pokud by tato metoda měla být provedena počítačem samostatně, pak by bylo nutno uložit všechna (nebo v každém případě velmi mnoho) česká slova. I když je to v principu možné, používali bychom zbytečně silný nástroj. A to nelze následující metodě vytknout.

2. *Statistická analýza*

V češtině, němčině a angličtině (stejně jako v každém přirozeném jazyku) se nevyskytují všechna písmena se stejnou četností; spíše má každé písmeno svou charakteristickou četnost.

Tyto početnosti jsou uvedeny v následující tabulce:

Písmeno	Četnost v % němčina	Četnost v % angličtina
a	6.51	6.40
b	1.89	1.40
c	3.06	2.70
d	5.08	3.50
e	17.40	10.00
f	1.66	2.00
g	3.01	1.40
h	4.76	4.20
i	7.55	6.30
j	0.27	0.30
k	1.21	0.60
l	3.44	3.50
m	2.53	2.00
n	9.78	5.60
o	2.51	5.60
p	0.79	1.70
q	0.02	0.40
r	7.00	4.90
s	7.27	5.60
t	6.15	7.10
u	4.35	3.10
v	0.67	1.00
w	1.89	1.80
x	0.03	0.03
y	0.04	1.80
z	1.13	0.02

Můžeme pak písmena rozdělit v závislosti na četnosti jejich výskytu do čtyř skupin (např. v němčině). V první skupině budou nejpočetněji se vyskytující **e** a **n**, v druhé skupině budou písmena s ještě relativně velkou četností (cca. 7%); ve třetí skupině jdou uvedena písmena s malou, ale dosti podstatnou četností zatímco v poslední skupině jsou uvedena zanedbatelná písmena.

Skupina	Počet písmen skupiny v textu
e, n	27.18%
i, s, r, a, t	34.48%
d, h, u, l, c, g, m, o, b, w, f, k z	36.52%
p, v, j, y, x, q	1.82%

Co se stane, dešifrujeme-li nějakou (německou) zprávu? Pak samozřejmě zůstane četnost písmen zachovánana; avšak jednotlivá četnosti písmen nemusí být již přiřazeny svým odpovídajícím písmenům. Např. zašifrujeme-li ve zpráve písmeno **e** za **X**, pak bude písmeno **X** nejčetnějším písmenem kryptogramu, zašifrujeme-li **y** za **S**, nebude se **S** v kryptogramu téměř vyskytovat.

Konkrétně Mr. X ve své analýze zprávy

MRNBNA CNGC RBC WRLQC VNQA PNQNRV

vytvoří seznam jednotlivých četností

Písmeno:	ABCDEFGHIJKLMNOPQRSTUVWXYZ
Četnost:	2 2 4 0 0 0 1 0 0 0 0 1 3 6 O 1 3 4 0 0 0 2 1 0 0 0.

Písmeno s největší četností je **N**; můžeme tedy v prvním přiblížení vyjít z toho, že **N** v kryptogramu odpovídá **e** ve zprávě.

Pozor: Protože celá metoda je založená na statistice, není nic 100% jisté! Mr. X se musí zabývat dalším potvrzením své domněnky. Je-li jeho domněnka správná, jedná se o posunutí o 9 písmen. Pak by muselo **R** odpovídat písmenu **i** (to potvrzuje jeho hypotézu, že se **R** vyskytuje relativně často) a **W** by muselo odpovídat písmenu **n** – to je však v rozporu s tím, že **W** se vyskytlo pouze jednou. Z druhé strany se vyskytují **A, B, C** relativně často (měly by odpovídat **r, s, t**, a ekvivalenty **x, y, z** (totiž **G, H, I** se prakticky nevyskytují).

Mr. X se tedy pokusí provést posunutí zpět o 9 písmen a obdrží

dieser text ist nicht mehr geheim.

Výše uvedený text je smysluplný a tedy jeho domněnka je s konečným závěrem potvrzena.

Závěrem několik poznámek. Druhá metoda má bezespornou přednost, že ji počítač může sám lehce provést. Protože zde ale rozhodující roli hrají statistické úvahy, musí být zachována jistá obezřetnost. Obzvláště při krátkých textech může vést naivní hledání po nejčastěji se vyskytujícím písmenu do slepé uličky. Pokud ale uvážíme ještě četnosti několika jiných písmen, lze použít algoritmy, které jsou i při velmi krátkých textech velmi úspěšné.

1.3 Monoabecední šifrování

Šifrování se nazývá **monoabecední**, jestliže každé písmeno abecedy zprávy je zašifrováno jako nějaké písmeno téže abecedy. Monoabecední šifrování si můžeme představit tím, že pod abecedu zprávy napíšeme abecedu kryptogramu. Např. následující metody šifrování jsou monoabecední.

Zpráva:	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
Kryptogram:	QWERTZUIOPASDFGHJKLYXCVBNM.

Zpráva:	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
Kryptogram:	ȷ → Π $\alpha\kappa\beta$ ˘ ⊥ ﬃ ı ≤ ´ Ξ Σ Υ Φ Ω ≥ 3 ß 7 9 ¯ 2 4 5 .

Poslední příklad by nám měl připomenout, že zpráva a kryptogram nemusí být definovány nad stejnou abecedou. Je-li tomu ale tak, pak každému monoabecední šifrování odpovídá permutace písmen abecedy; obráceně lze každé permutaci přiřadit monoabecední šifrování. Z toho zejména plyne, že máme přesně

$$26! = 26 \cdot 25 \cdot \ldots \cdot 2 \cdot 1 \approx 4 \cdot 10^{26}$$

monoabecedních šifrování nad přirozenou abecedou $\{$**a, b, c,** ..., **z**$\}$.

1.4 Záměnné šifry

Chceme-li použít k zakódování písmen počítač, pak identifikujeme obvykle **a** (resp. **A**) s **1**, **b** (resp. **B**) s **2**, atd.; **x** (resp. **X**) s **24**, **y** (resp. **Y**) s **25** a **z** (resp. **Z**) s **0**. Pomocí této reprezentace lze posunovací šifry popsat obzvlášť dobře: posunutí o s míst odpovídá přičtení čísla s modulo 26. Konkrétně postupujeme následovně:

- Nejdříve převedeme písmena zprávy do odpovídajících číslic;
- pak připočteme k tomuto číslu číslo s;
- z výsledku uvažujeme pouze zbytek, který obdržíme po dělení 26; tento zbytek přeložíme zpátky na odpovídající písmeno.

Tímto způsobem získáme příslušný text kryptogramu.
Příklad. Nyní chceme zašifrovat písmeno **a** pomocí posunovací šifry, která posouvá o 3 místa.

- **a** se reprezentuje pomocí číslice **1;**
- **1 + 3 = 4;**
- **4 je reprezentace písmena D** kryptogramu.

Při dešifrování písmene **B postupujeme následovně:**

- **B** se reprezentuje pomocí číslice **2;**
- **2 – 3 = –1;**
- **Zbytek –1** po dělení**26** je **25** a to odpovídá písmenu **y** zprávy.

S pomocí této metody lze tzv. čísla *sčítat.*Celá věc se stává podstatně zajímavější, pokud budeme písmena *násobit.*To lze provést následovně:

Abychom mohli násobit písmena číslem t, budeme počítat opět modulo 26. Tzn., že vynásobíme číslo odpovídající zadanému písmenu číslem t a uvažujeme zbytek po dělení 26. Pak je tomuto zbytku odpovídající písmeno výsledek tohoto „násobení“.

Vynásobíme-li hodnotu každého písmena zprávy číslem 2, obdržíme

Zpráva:	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
Kryptogram:	BDFHJLNPRTVX ZBDFHJLNPRTVXZ.

Vidíme, že vždy dvě písmena (např. **h**a **u**) nám dávají tentýž „součin“ (v našem případě **P**). Proto nemůžeme tuto substituci použít jako šifru. Pro každé šifrování musí totiž platit doposud nevyslovené ale zcela samozřejmé pravidlo, že *text zprávy musí být s pomocí klíče* **jednoznačně** *rekonstruovatelný z kryptogramu. Mnozí považují tozo pravidlo za příliš omezující; lze ho však zeslabit a zároveň odůvodnit:* Každý kryptogram musí být s pomocí klíče dešifrovatelný **nějakým počítačem.**

Pokusme naše štěstí ještě jednou a vynásobme všechna písmena číslem **3**:

Zpráva:	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
Kryptogram:	CFILORUXADGJMPSVYBEHKNQTWZ.

V tomto případě získáme skutečně monoabecední šifrování. Lehkým prozkoušením vidíme, že obdržíme monoabecední šifrování právě tehdy, když násobíme jedním z čísel **1, 3, 5, 7, 9, 15, 17, 19, 21, 23** nebo **25**; takovéto šifrování nazýváme **multiplikativní šifry**.

Máme tedy (včetně triviální šifry) právě 12 multiplikativních šifrování; jejich počet je tedy ještě menší než počet posouvacích šifer. Proto můžeme od těchto šifer očekávat velmi nepatrnou kryptografickou bezpečnost.

Můžeme ale navzájem kombinovat posouvací a multiplikativní šifry. Za tímto účelem přičteme nejprve k písmenu zprávy číslo s a výsledek vynásobíme dalším číslem t. Podle tohoto předpisu získáme opět šifru, tzv. **záměnnou (afinní)šifru**, kterou budeme označovat $[s, t]$.

Klíč záměnné šifry $[s, t]$ sestává z dvojice čísel (s, t) (přirozeně musí být pro každou záměnnou šifru číslo t zvoleno tak, že násobení číslem t je multiplikativní šifra; t tedy musí být jedno z výše uvedených čísel **1, 3, 5, 7, 9, 15, 17, 19, 21, 23** nebo **25**).

Počet všech záměnných šifer vypočteme jako počet všech posuvacích šifer vynásobený počtem všech multiplikativních šifer; tedy počet všech záměnných šifer je $26 \cdot 12 = 312$. Toto číslo je už tak velké, že při ruční kryptoanalýze nám systematické prověření všech možností dá pěkně zabrat.

1.5 Klíčová slova

Velké množství monoabecedních šifer získáme následujícím způsobem: **klíč** sestává ze dvou komponent – **klíčového slova** a **klíčového písmene**. Nejdříve vytvoříme z klíčového slova posloupnost písmen, ve které se každé písmeno vyskytne pouze jednou. To získáme následujícím způsobem, že každé písmeno se při svém druhém, třetím, … výskytu vyškrtne. Máme-li zvoleno například klíčové slovo

MATEMATIKA,

získáme posloupnost

MATEIK.

Napišme nyní tuto posloupnost pod abecedu zprávy, a to tím způsobem, že bude začínat právě pod klíčovým písmenem. Např., zvolili jsme jako klíčové písmeno „j", obdržíme

Zpráva:	abcdefghi j k l mn o pqrstuvwxyz
Kryptogram:	MATEIK .

Na závěr napíšeme zbývající písmena kryptogramu v abecedním pořádku, přičemž začneme po posledním písmenu klíčového slova. V našem případě pak získáme

Zpráva:	a b c d e f g h i j k l mno p q r s t uvwxyz
Kryptogram:	QRSUVWXYZMATEIKBCDFGHJLNOP.

Zřejmě lze *každé* monoabecední šifrování získat pomocí vhodného klíčového slova.

1.6 Kryptoanalýza

„Filozofii" moderní kryptoanalýzy lze popsat Kerckhoffovým principem; tento princip byl poprvé formulován v knize *La cryptographie militaire* (1883) holandským jazykovědcem Jeanem Guillaumem Hubertem Victorem Francoisem Alexandrem Augustem **Kerckhoffem von Nieuwenhofem** (1835–1903).

Kerckhoffův princip. *Spolehlivost kryptosystému nesmí záviset na utajení algoritmu. Spolehlivost je založena pouze na utajení klíče.*

To je zásadní varování pro tvůrce kryptosystémů. Nesmíme být tak naivní a přepokládat, že Mr. X nemá možnost získat znalost algoritmu. Dějiny kryptografie jsou plné příkladů, kdy objevitel kryptosystému založil důvěru na něm tím, že jeho algoritmus nikdy nemohl být znám.

Naopak: Cílem moderní kryptografie musí být vývoj systémů, které zůstanou bezpečné i v tom případě, že o algoritmu bylo dlouhou dobu veřejně diskutováno. „Příkladem“ je DES-algoritmus.

Kryptoanalytik rozlišuje následující případy útoku na šifru:

1. Known ciphertext attack:

Kryptoanalytik zná relativně velkou část kryptogramu. To je opravdu reálný předpoklad, protože je zpravidla celkem jednoduché zajistit si (libovolně dlouhé) části kryptogramu.

2. Known plaintext attack:

Kryptoanalytik zná relativně malou část související zprávy/kryptogramu. Tato hypotéza je reálnější, než se na první pohled zdá. Totož často „ví“ Mr. X, o co se jedná, a může proto uhádnout několik hlavních slov. Mimoto lze nalézt zpravidla standardní úvodní a závěrečné fráze atd.

3. Chosen plaintext attack:

Má-li kryptoanalytik přístup k šifrovacímu algoritmu (s aktuálním klíčem), může pak za účelem zjištění klíče kódovat vybrané části zprávy a pokusit se udělat z obdrženého kryptogramu závěry o struktuře klíče. Mohl by například do stroje vkládat pravidelná zdrojová slova, např. posloupnosti stejných písmen (**aaa** ...) za účelem jejich zakódování. Nebezpečí takovéhoto útoku spočívá v tom, že by se mohlo Mr. X podařit přimět šifrovací stroj k tomu, aby zakódoval *zdánlivě* neškodné zprávy, s jejichž pomocí pak Mr. X může zašifrovat zprávu, kterou by odesilatel nikdy nezašifroval. Jak nebezpečný může být takovýto útok, se obzvláště ukáže, když pomyslíme na to, že mnohé šifrovací přístroje pouze nešifrují, ale také „podpisují: Pokud je algoritmus tak slabý, že dovolí tento útok, pak by mohl Mr. X vytvořit z nevinně vyhlížejících podepsaných zpráv brizantní, platně podepsaný dokument.

*

Každé monoabecední šifrování přirozeného jazyka může být dosti lehce prolomena. Musíme si pouze ujasnit, že každé monoabecední šifrování (přirozeného jazyka) lze prolomit již za vysoce slabého (přičemž nanejvýš realistického) předpokladu 1. Předvedeme pouze „princip“ algoritmu.

Představme si, že Mr. X zachytil kryptogram o délce asi 500 písmen a že ví, že kryptogram byl zašifrován pomocí monoabecedního šifrování.

Krok 1. Nejdříve Mr. X zjistí četnosti písmen kryptogramu. Tím získá ekvivalent **e, n** spolu s množinou písmen {**i, s, r, a, t**}. Jednotlivá písmena z této množiny přitom zpravidla ještě nemůže identifikovat.

Krok 2. Nyní Mr. X spočte bigramy, tzn. páry po sobě sledujících písmen. Nejčastější bigramy německého jazyka jsou uvedeny v následující tabulce:

Bigram	Četnost	Bigram	Četnost
en	3.88%	**nd**	1.99%
er	3.75%	**ei**	1.88%
ch	2.75%	**ie**	1.79%
te	2.26%	**in**	1.67%
de	2.00%	**es**	1.52%

Tímto způsobem může Mr. X izolovat písmena o největším výskytu. Např. dvojice **er** má velmi velkou četnost, zatímco všechny jiné kritické kombinace s **e** se vyskytují dosti zřídka (**ea** a **et** jsou opravdu velmi řídké – pod 0.5%) a také **es** se vyskytuje se signifikantně malou četností. Konkurentem by mohla být dvojice **ei**; tu však lze vyřadit tím způsobem, že testujeme inverzní dvojice: jen u těchto dvojic je tomu tak, že jak v původní posloupnosti, tak i v obráceném pořadí se vyskytují s prakticky stejnou četností. Tímto způsobem může Mr. X nejprve izolovat rozlišitelná písmena skupiny { **i, s, r, a, t**} s druhou největší četností. Dále může rozpoznat písmena **c** a **h** podle toho, že se jako dvojice vyskytují relativně často ale samostatně velmi zřídka. Tímto způsobem může prakticky bezchybně identifikovat nejčastěji se vyskytující písmena; tj. písmena **e, n, i, s, r, a, t, h, c**, která tvoří více než dvě třetiny textu. A to všechno lze provést automaticky pomocí počítače!

Krok 3. Pak nechá Mr. X dosadit rozpoznaná písmena do celého textu. Jinak řečeno: počítač rozšifruje známé díly textu. Ten se objeví na obrazovce, příčemž nerozluštěná písmena se nahradí účelně prázdnými znaky.

Zpravidla tento text není rozšifrován, nebo se jeho šifrování provádí ještě stále náročně. Další písmena však může inteligentní Mr. X na základě kontextu lehce *hádat*! To provede a nechá si vždy po každém kroku ukázat změněný text. Po dvou nebo třech krocích dospěje k velmi dobře čitelnému textu.

*

Shrnutí: Monoabecední šifrování nad přirozeným jazykem jsou pozoruhodně nejistá (**přirozený jazyk** má málo písmen, jež jsou dost nerovnoměrně rozdělena). V současné době proto používáme buď monoabecední šifrování nad *umělým jazykem* nebo *poly*abecední šifrování.

Nejpopulárnější monoabecední šifrování je **DES** – **D**ata **E**ncryption **S**tandard. Tento algoritmus byl vyvinut firmou IBM a v roce 1977 se stal standardem. DES nešifruje písmena, nýbrž symboly 0 a 1, a to 64 naráz (v případě, že používáme DES k zašifrování obyčejného textu, musí být písmena nejdřív přeložena do řetězce bitů. Jednou z takových metod je použití ASCII kódu. Tedy DES je monoabecední algoritmus nad abecedou $\{(a_1, \ldots, a_{64}) : a_i \in \{0, 1\}\}$. Jako klíč lze se bere posloupnost binárních řetězců z 56 bitů, tj. všech klíčů je právě $2^{56} \approx 7 \cdot 10^{16}$.

Tento algoritmus byl od samého počátku úplně publikován – jednalo se o první algoritmus historii, kdy se to stalo. DES se převážně úspěšně používá v bankovnictví.

V roce 1990 předvedli Biham a Shamir, že DES je algoritmus vyvinutý podle vynikajících principů; všeobecně však převládá obecné přesvědčení, že by délka klíče měla být větší. Biham a Shamir zároveň ukázali, že mnoho algoritmů příbuzných DESu lze jejich chytrou analýzou rozbít.

Kapitola 2

Proč jednoduše, když to jde i složitě?

Slovo, věta a z šifer se zvedá
poznaný život, náhlý smysl.
Gottfried Benn

V této kapitole se budeme zabývat převážně **polyabecedními** šifrováními. U těchto šifrování se písmena zprávy nešifrují pomocí téže abecedy. Zejména tedy nelze polyabecední šifrování popsat jednoduše pomocí abecedy zprávy a pod ní napsané abecedy kryptogramu.

Přiřazení písmene zprávy k nějakému písmenu kryptogramu nesmí být prováděno svévolným způsobem. Šifrování musí splňovat silný požadavek **jednoznačnosti**; v opačném případě by nebylo možné žádné dešifrování. Jinak řečeno: kdyby nebylo šifrování jednoznačné, nenácházel by se příjemce principiálně v žádné lepší situaci než Mr. X!

Typickým příkladem algoritmu, u kterého není šifra jednoznačná, je **homofonní** šifra. Takovéto algoritmy budou předvedeny v následujícím odstavci. Největší část kapitoly bude však věnována zkoumání takovýchto polyabecedních šifer, které vzniknou z kombinací monoabecedních algoritmů; takovýmto charakteristickým příkladem je Vigienérevo šifrování.

2.1 Zneprůhlednění četností

Jak můžeme dosáhnout toho, že všechna písmena kryptogramu se v něm vyskytují se stejnou četností? Zcela jednoduše: Šifrovací předpis přiřadí každému písmeno nikoliv znak, nýbrž *množinu* znaků (v našem příkladu to budou dvojice čísel), a to tak, že jsou splněny následující podmínky:
– Aby bylo dešifrování jednoznačné, musí být množiny příslušející různým písmenů zprávy disjunktní.

– Počet písmen kryptogramu, které patří k nějakému písmenu zprávy, odpovídá četnosti tohoto písmene.

V následujícím příkladu homofonní šifry jsou znaky kryptogramu tvořeny 100 dvojicemi číslic 00, 01, . . . , 99:

Homofonní šifra	
Písmeno	Zašifrování (němčina)
a	10 21 52 59 71
b	20 34
c	28 06 80
d	04 19 70 81 87
e	09 18 33 38 40 42 53 54 55 60 66 75 85 86 92 93 99
f	00 41
g	08 12 97
h	07 24 47 89
i	14 39 46 50 65 76 88 94
j	57
k	23
l	16 03 84
m	27 11 49
n	30 35 43 62 63 67 68 72 77 79
o	02 05 82
p	31
q	25
r	17 36 51 69 74 78 83
s	15 26 45 56 61 73 96
t	13 32 90 91 95 98
u	29 01 58
v	37
w	22
x	44
y	48
z	64

Při zašifrování se přiřadí písmenu zprávy náhodně jeden z příslušných znaků kryptogramu. Příjemce pak může pomocí výše uvedené tabulky jednoduš dešifrovat následující text:

00097449599505371089483102139964713748836696427752.

Protože znaky byly vybrány náhodně, vyskytuje se každý znak (v našem případě dvojice číslic) stejně často (odtud jméno „homofonní“). Případný kryptoanalytik je tak postaven před podstatně těžší úlohu než při zkoumání monoabecedního šifrování.

Avšak neměli bychom příliš jásat, neboť **kryptoanalýza** je také zde možná. Analýza je založena na pozorování, že četnosti písmen kryptogramu jsou sice stejné, ale že z uvažování nad *dvojicemi* znaků kryptogramu můžeme stejně dobře získat novou informaci. Budeme diskutovat dva příklady

- Uvažuje-li Mr. X ekvivalent písmena **c**, tedy dvojici 28, zjistí, že v úvahu jako přirozený následník přichází pouze určitá písmena kryptogramu. Toto jsou dvojice 07, 24, 23, 47, 89, tedy ekvivalenty písmen **h** a písmen**k**.

- Zkoumáme-li ekvivalent písmene **e**, tedy např. 99, zjistíme, že jisté znaky kryptogramu se vyskytují jako předchůdci a následníci 99 – a to prakticky stejně početně. Musí se pak jednat o ekvivalenty písmene **i**.

2.2 Vigenerova šifra

Vigenerovo zašifrování bylo veřejnosti zpřístupněno v roce 1586 francouzským diplomatem Blaisem de **Vigenere** (1523–1596). Základní idea je používat střídavě různá monoabecední šifrování. Tato idea je tak přirozená, že variace Vigenerova zašifrování byly vícenásobně znovuobjeveny. Dva z nejdůležitějších předchůdců byli Johannes **Trithemius** (1462 – 1516), jehož knihy *Poligraphia* (1518) a *Steganographia* (1531) byly uveřejněny posmrtně, a Giovanni Battista **Della Porta** (1538-1615), vynálezce přístroje *Camera obscura*, který v roce 1558 ve své knize *Magia naturalis* zveřejnil polyabecední algoritmus, který vykazuje velkou podobu s Vigenerovou šifrou.

V této kapitole se budeme hlavně zabývat Vigenerovou šifrou, která je nejznámější mezi všemi periodickými polyabecedními algoritmy, a to ze dvou důvodů:

- Vigenerova šifra je prototyp mnoha algoritmů, které byly profesionálně používány až do našeho století (Caesarova šifra je zvláštním případem Vigenerovy šifry pro klíčové slovo délky 1).

- Při kryptoanalýze se seznámíme se dvěmi extrémně důležitými metodami, a to **Kasiského testem** a **Friedmanovým testem**.

Abychom mohli použít Vigenerův algoritmus, potřebujeme dvě věci: **klíčové slovo** a **Vigenerův čtverec**. Tento čtverec se skládá z 26 abeced, které jsou napsány pod sebou takovým způsobem, že první abeceda je obyčejná abeceda, druhá abeceda je o jedno písmeno posunutá, třetí o dvě atd. Jinak řečeno: Vigenerův čtverec sestává z 26 posouvacích šifer v přirozeném pořadí.

Vigenerův čtverec

Zpráva:	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
Kryptogram:	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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

Klíčovým slovem může být libovolná posloupnost písmen; pro náš demonstranční případ vybereme slovo „VENUSE“.

Klíčové slovo:	V	E	N	U	S	E	V	E	N	U	S	E
Zpráva:	**p**	**o**	**l**	**y**	**a**	**b**	**e**	**c**	**e**	**d**	**n**	**i.**

Při šifrování určí písmeno klíčového slova, které stojí nad určitým písmenem zprávy příslušnou abecedu tj. *řádku* ve Vigenerově čtverci a pomocí této abecedy bude písmeno zprávy šifrováno.

Celkem tedy máme

Klíčové slovo:	V	E	N	U	S	E	V	E	N	U	S	E
Zpráva:	**p**	**o**	**l**	**y**	**a**	**b**	**e**	**c**	**e**	**d**	**n**	**i**
Kryptogram:	**K**	**S**	**Y**	**S**	**S**	**F**	**Z**	**G**	**R**	**Y**	**F**	**M.**

Je jasné, že takováto šifrovací metoda staví Mr. X před podstatně větší problémy, než je tomu při monoabecedním šifrování. Četnost písmen je daleko rovnoměrnější, což lze poznat i na našem krátkém příkladu. Např. písmeno zprávy **e** bylo zašifrováno do **Z** a **R**, písmeno kryptogramu **S** vzniklo ze tří různých písmen zprávy (**o, y, a**).

2.3 Kryptoanalýza

Přirozeně lze i pomocí dnešních metod prolomit text zašifrovaný pomocí Vigenerovy šifry. Totiž dostatečně dlouhý text vykazuje mnoho statisticky zachytitelných pravidelností, které umožňují zjištění klíčového slova. První uveřejněný útok byl publikován v roce 1863 pruským majorem dělostřelectva Friedrichem Wilhelmem **Kasiským** (1805–1881). Druhá metoda se připisuje plukovníkovi Williamu Fredericku Friedmanovi (1891–1969). Obě metody slouží k určení délky klíčového slova. Protože oba testy mají i mimo speciální analýzu Vigenerovy šifry svůj dalekosáhlý a zásadní význam, představíme podrobně obě metody.

Předpokládejme, že Mr. X zachytil následující text, o kterém ví (nebo se domnívá), že je zašifrován pomocí Vigenerovy šifry:

Kryptogram

UEQPCVCKAHVNR ZURNLAO
KIRVGJTDVRVRI CVIDLMY
IYSBCCOJQSZNY MBVDLOK
FSLMWEFRZAVIQ MFJTDIH
CIFPSEBXMFFTD MHZGNMW

KAXAUVUHJHNUU LSVSJIP
JCKTIVSVMZJEN ZSKAHZS
UIHQVIBXMFFIP LCXEQXO
CAVBVRTWMBLNG NIVRLPF
VTDMHZGNMWKRX VRQEKVR

LKDBSEIPUCEAW JSBAPMB
VSZCFUEGITLEU OSJOUOH
UAVAGZEZISYRH VRZHUMF
RREMWKULKVKGH AHFEUBK
LRGMBJIHLIIFW MBZHUMP

LEUWGRBHZOLCK CWTHWDS
ILDAGVNEMJFRV QSVIQMU
VSWMZCTHIIWGD JSXEOWS
JTKIHKEQ

2.3.1 Kasiského test

Ačkoliv tato působivá metoda analýzy polyabecedních algoritmů byla poprvé publikována Kasiským, musíme se zmínit, že anglický matematik Charles **Babbage** (1792–1871), který je mimo jiné znám svou koncepcí předchůdce moderního počítače, provedl neuveřejněná zkoumání i v kryptografii. Mj. vyvinul Kasiského test už v roce 1854, tj. devět let před Kasiským.

Test je založen na následující myšlence: vyskytují-li si ve zprávě dvě posloupnosti stejných písmen (např. v němčině slovo **ein**), mohou obecně odpovídající posloupnosti v kryptogramu dopadnout různě. Jsou-li ale obě počáteční písmena posloupností zašifrována pomocí téhož písmene klíčového slova, jsou i obě písmena kryptogramu stejná. V tomto případě bude také druhé písmeno posloupnosti v zprávě zašifrováno pomocí téhož písmene klíčového slova; tedy obdržíme i v kryptogramu stejné písmeno. To tedy znamená: Budou-li obě počáteční písmena posloupností zprávy zašifrována pomocí téhož písmene klíčového slova, pak sestávají obě posloupnosti v kryptogramu ze stejných písmen.

Kdy může nastat případ, že dvě písmena jsou zašifrována pomocí téhož písmene klíčového slova? Právě tehdy, když se klíčové slovo mezi tato písmena n-krát vejde pro vhodné přirozené n.

*

Když nyní Mr. X najde v kryptogramu dvě posloupnosti skládající se se stejných písmen, pak se může domnívat, že jejich vzdálenost je několikanásobek délky klíče. Tato pravděpodobnost se řídí pravidlem „čím delší, tím milejší“: stejná písmena nevypovídají, že víme něco o délce klíče, a také dvojice složené ze stejných písmen by mohly vzniknout náhodně. Z posloupností třech nebo více písmen již může Mr. X dostatečně spolehlivě usuzovat na délku klíčového slova. V našem případě pak zjistí:

Posloupnost	Odstup	Rozklad na součin prvočinitelů odstupu
JTD	50	$2 \cdot 5 \cdot 5$
VIQM	265	$5 \cdot 53$
TDMHZGNMWK	75	$2 \cdot 3 \cdot 3 \cdot 5$
MWK	75	$3 \cdot 5 \cdot 5$

Největší společný faktor je 5. Optimistický kryptoanalytik by mohl říci, že „že délka klíčového slova je 5“ (ve skutečnosti funguje Kasiského test v praxi velmi dobře). Pokud je ale kryptoanalytik opatrný, mluví pouze o silné indicii pro délku klíčového slova 5. Jsou dva důvody pro jeho opatrnost:

1. Mohl by nastat případ, že se vyskytují dvě posloupnosti kryptogramu ze tří nebo více stejných písmen, které mají vzdálenost nedělitelnou pěti. Pak

bychom získali jako největší společný dělitel jedničku! (V našem případě tento případ skutečně nastane: posloupnost **KAH** se vyskytne dvakrát a sice s odstupem $128 = 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2$.) To znamená, že nesmíme počítat největší společný dělitel „slepě“ pomocí počítače, ale musíme jej určit „citem“. Musíme tedy zřejmé chyby vypustit.

2. Právě proto bychom mohli přijít na myšlenku, že délka klíče by nemusela být pouze 5, nýbrž 10, 15 nebo 30 (neboť faktory 2 a 3 se vyskytují také dostatečně často). Jinak řečeno: Kasiského test nám poskytuje délku klíčového slova až na *násobek*.

Kryptogram

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

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

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

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

Z výše uvedeného důvodu budeme prezentovat druhou metodu; tato určuje *řádový odhad* délky klíčového slova. Kombinace těchto obou metod je prakticky úplně spolehlivá.

2.3.2 Friedmanův test

Tento postup byl vyvinut Williamem Friedmanem v roce 1925. V tomto testu se ptáme na to,*s jakou šancí se náhodně vybraný pár písmen ze zprávy sestává ze stejných písmen.* Odpověď je pak dána indexem koincidence.

Představme si nejprve libovolnou posloupnost písmen délky n. Buď n_1 počet písmen **a**, n_2 počet písmen **b**, $\ldots$, n_{26} počet písmen **z**.

Zajímáme se o počet dvojic, kdy jsou obě písmena rovna **aa**. (Nepožadujeme, aby se uvažované dvojice skládaly za sebou následujích písmen.) Pro počet prvního **a** máme práve n_1 možností, pro výběr druhého **a** zbývá $n_1 - 1$ možností. Protože nezáleží na pořadí písmen, je počet hledaných dvojic roven $\frac{n_1 \cdot (n_1 - 1)}{2}$.

Je tedy počet dvojic, kdy jsou obě písmena stejná, roven

$$\frac{n_1 \cdot (n_1 - 1)}{2} + \frac{n_2 \cdot (n_2 - 1)}{2} + \ldots + \frac{n_{26} \cdot (n_{26} - 1)}{2} = \sum_{i=1}^{26} \frac{n_i \cdot (n_i - 1)}{2}.$$

Šance obdržení dvojice složené ze stejných písmen je určena následujícím výrazem:

$$\mathbf{I} = \frac{\sum_{i=1}^{26} n_i \cdot (n_i - 1)}{n \cdot (n - 1)}$$

a nazývá se Friedmanův **index koincidence**. Friedman sám značil toto číslo jako κ, proto se občas pro metodu, kterou v dalším předvedeme, používá název **kappa-test**.

*

Přibližme se nyní tomuto indexu koincidence z jiné strany. Předpokládejme, že bychom věděli, že se v našem textu vyskytuje písmeno **a** s pravděpodobnpstí p_1, písmeno **b** s pravděpodobnpstí p_2, $\ldots$, písmeno **z** s pravděpodobnpstí p_{26}. (Konkrétní hodnoty pro pravděpodobnosti p_i můžeme udat, jestliže víme, ze kterého jazyka text pochází.)

Představme si nyní dvě libovolně vybraná písmena našeho textu. Pravděpodobnost, že první písmeno je rovno **a** je p_1; tedy přibližná pravděpodobnost, že obě písmena jsou rovna **a** je ${p_1}^2$ (pokud n je dostatečně velké, lze takto vzniklou chybu zanedbat). Odpovídající vztahy platí i pro ostatní písmena. Tedy pravděpodobnost toho, že obě písmena jsou si rovna je

$${p_1}^2 + {p_2}^2 + \ldots + {p_{26}}^2 = \sum_{i=1}^{26} {p_i}^2.$$

Toto číslo závisí přirozeným způsobem na pravděpodobnostech $p_1, p_2, \ldots, p_{26}$. Spočtěme si dva příklady.

- Pro text v německém jazyce máme

  $$\sum_{i=1}^{26} {p_i}^2 = 0.0762.$$

  To znamená, že náhodně zvolená dvojice písmen se skládá ze dvou stejných písmen s šancí 7,62%.

- Představme si obráceně zcela náhodný text, tj. text, ve kterém jsou písmena divoce promíchána. Pak každé písmeno se zde vyskytne se stejnou pravděpodobností
$$p_i = \frac{1}{26}.$$
V tomto případě pak
$$\sum_{i=1}^{26} {p_i}^2 = \sum_{i=1}^{26} \frac{1}{26^2} = \frac{1}{26} = 0.0385.$$
Šance, že v takovémto textu najdeme dvě stejná písmena, se nám zmenšila na polovinu.

1. Pokud známe pravděpodobnosti $p_1, p_2, \ldots, p_{26}$ (jako je tomu např. s němčinou či angličtinou), pak víme, že součet čtverců pracděpodobností je přibližně roven indexu koincidence:
$$I \approx \sum_{i=1}^{26} {p_i}^2.$$
Obecně lze pak dokázat, že index koincidence (nebo stejně platně i $\sum_{i=1}^{26} {p_i}^2$) je tím větší, čím je text *nepravidelnější*, a menší, čím je text *pravidelnější*. Hodnota 0.0385 je absolutní minimum proi index koincidence. Totiž
$$0 \leq \sum_{i=1}^{26} (p_i - \frac{1}{26})^2 = \sum_{i=1}^{26} {p_i}^2 - \frac{1}{26}.$$

2. Vraťme se na okamžik k *mono*abecedním šifrováním. Protože monoabecední šifrování je pouze permutace písmen, zůstává rozdělení četností zachováno. (Četnosti jednotlivých písmen jsou permutovány zároveň s písmeny). Např. četnost 0.17 už nepatří písmenu **e**, nýbrž jeho ekvivalentu v kryptogramu.

   Máme tedy, že při monoabecedním šifrování index koincidence zůstává zachován, zatímco při polyabecedním šifrování klesá, vzhledem k tomu, že polyabecední šifrování bylo vytvořeno za tím účelem, aby se navzájem vyrovnaly četnosti jednotlivých písmen.

   Z toho lze odvodit **test**, který nám ukáže, zda byl kryptogram vytvořen monoabecedním šifrováním nebo ne: Nejprve vypočteme index koincidence kryptogramu. Je-li tento index přibliženě 0.0762, je šifrování pravděpodobně monoabecední. Je-li index koincidence zřetelně menší, můžeme vycházet z toho, že text byl šifrovan polyabecedním šifrováním.

*

Nyní použijeme index koincidence k výpočtu délky klíčového slova pro text zašifrovaný Vigenerovou šifrou. Cílem je určit index koincidence textu bez jeho znalosti. Protože bylo použito polyabecedního algoritmu, je index koincidence menší než 0.0762. Ale o kolik menší? Odpověď je, že to závisí na délce klíčového slova.

Předpokládejme, že klíčové slovo má délku l a skládá se z navzájem různých písmen.

Rozepišme náš kryptogram do l sloupců. Pak se v prvním sloupci nacházejí písmena číslo $1, l+1, 2l+1, \ldots$, tedy všechna ta písmena, která byla zašifrována pomocí prvního písmene klíčového slova. Podobně se v druhém, $\ldots$, l-tém sloupci nacházejí všechna ta písmena, která byla zašifrována pomocí druhého, $\ldots$, l-tého písmene klíčového slova.

Písmeno S_i klíč. slova	S_1	S_2	S_3	S_l
	1	2	3	l
	$l+1$	$l+2$	$l+3$	$2l$
	$2l+1$	$2l+2$	$\ldots$	$3l$
	$3l+1$	$\ldots$		
	$\ldots$			

Podrobnějším studiem výše uvedeného schématu lze vypočítat index koincidence.

První pozorování: Každý sloupec byl získán pomocí monoabecedního šifrování (dokonce pomocí posouvací šifry). Šance, že zde vybereme dvojici stejných písmen, je tedy rovna 0.0762. Uvažme nyní dvojice písmen, která stojí v různých sloupcích. Protože příslušné šifrovací abecedy byly vybrány „náhodně“, může se takováto dvojice skládat ze stejných písmen pouze náhodným způsobem.

Pravděpodobnost pro tento jev je podstatně nižší než 0.0762, tj. přibližně 0.0385. (Je to přesně $\frac{1}{26}$, jestliže je klíčové slovo náhodná posloupnost písmen. Pokud ne, je tato pravděpodobnost o něco vyšší.)

Druhé pozorování: Vypočtěme nyní počet dvojic písmen ze stejných sloupců a z různách sloupců. Má-li náš kryptogram celkem n písmen, pak v každém sloupci stojí přesně n/l písmen (Vzdáme se uvažování zaokrouhlovacích chyb; budeme předpokládat, že text je tak dostatečně dlouhý, že zaokrouhlovací chyby se neprojeví.)

Pro výběr jednoho písmene máme přesně n možností. Je-li toto písmeno zvoleno, pak je pevně určen i sloupec, ve kterém leží. V tomto sloupci máme k dispozici ještě zbývajících $n/l-1$ písmen, tedy právě tolik možností pro výběr druhého písmene. Je tedy počet dvojic písmen, která se nacházejí *v tom samém sloupci* roven

$$n \cdot (\frac{n}{l} - 1)/2 = \frac{n \cdot (n-l)}{2l}.$$

Protože máme k dispozici právě $n - n/l$ písmen mimo určený sloupec, je počet dvojic písmen *z různých sloupců* rovna

$$n \cdot (n - \frac{n}{l})/2 = \frac{n^2 \cdot (l-1)}{2l}.$$

Na základě výše zmíněného pak máme, že očekávaný počet A dvojic stejných písmen je roven

$$A = \frac{n \cdot (n-l)}{2l} \cdot 0.0762 + \frac{n^2 \cdot (l-1)}{2l} \cdot 0.0385.$$

Pravděpodobnost, že získáme dvojici složenou ze stejných písmen, je rovna

$$\frac{A}{n \cdot (n-1)/2)} = \frac{(n-l)}{l \cdot (n-l)} \cdot 0.0762 + \frac{n \cdot (l-1)}{l \cdot (n-1)} \cdot 0.0385,$$

tj. po úpravě

$$\frac{A}{n \cdot (n-1)/2)} = \frac{1}{l \cdot (n-1)} \cdot [0.0377 \cdot n + l \cdot (0.0385 \cdot n - 0.0762)].$$

Zároveň víme, že index koincidence **I** je aproximací tohoto čísla; proto platí

$$\mathbf{I} = \frac{0.0377 \cdot n}{l \cdot (n-1)} + \frac{0.0385 \cdot n - 0.0762}{n-1}.$$

Vyjádříme-li si z výše uvedeného vztahu l, získáme důležitou Friedmanovu formuli pro délku klíčového slova:

$$\boxed{l \approx \frac{0.0377 \cdot n}{(n-1) \cdot \mathbf{I} - 0.0385 \cdot n + 0.0762}.}$$

*

Použijme nyní tuto formuli na náš příklad. Najdeme-li všechna n_i, obdržíme $n = 368, \sum_{i=1} 26 {n_i}^2 = 5924$.

Máme tedy

$$\mathbf{I} = \frac{5924}{135056} = 0.0439.$$

Jedná se tedy s velkou pravděpodobností o polyabecední šifrování. Spočtěme nyní délku klíčového slova l :

$$l \approx 6.5.$$

To ukazuje současně s výsledkem testu Kasiského na to, že délka klíčového slova je skutečně 5 (a ne 10, 15 nebo 20).

2.3.3 Určení klíčového slova

Jakmile je zjištěna délka klíčovího slova, jde o to poznat klíčové slovo samotné. Ale to už není tak těžké.

Pokud kryptoanalytik Mr. X zná délku klíčového slova, ví, že písmena č. $1, l+1, 2l+1, \ldots$ příp. č. $2, l+2, 2l+2, \ldots$ atd. byla získána pomocí monoabecedního šifrování (dokonce pomocí posouvací šifry). Zpravidla tedy stačí nalézt ekvivalent písmene **e**.

V našem příkladu je $l = 5$. Ze 74 písmen první „monoabecední" části je 14 rovno **V**. Proto odpovídá **e** písmenu **V**. Z Vigenerova čtverce pak obdržíme , že první písmeno klíčového slova je „R". Analogicky, Ze 74 písmen druhé, třetí, čtvté a páté „monoabecední" částí je 11 rovno **E**, 8 rovno **H**, 21 rovno **M** a 13 rovno S. Proto odpovídá **e** písmenu **E**, **H**, **M** a **S**. Opětovným nahlédnutím do Vigenerova čtverce pak obdržíme , že další písmena klíčového slova jsou po řadě „A", „D", „I" a „O". Rozšifrování textu je již standartní záležitostí. Snadným porovnáním získáme

Zpráva

```
denhoechsteno rganisa
tionsstanderf uhrdiek
ryptologieinv enedigw
osieinformein erstaat
lichenbuerota etigkei

tausgeuebtwur deesgab
schluesselsek retaere
dieihrbueroim dogenpa
lasthattenund fuerihr
etaetigkeitru ndzehnd

ukatenimmonat bekamen
eswurdedafuer gesorgt
dasssiewaehre ndihrer
arbeitnichtge stoertw
urdensiedurft enihreb

uerosaberauch nichtve
rlassenbevors ieeineg
estellteaufga begeloe
sthatten
```

2.3.4 Závěrečné poznámky

Viděli jsme, že každé Vigenerovo šifrování s dostatečně krátkým klíčem (aby se mohla ke slovu dostat pravděpodobnost ve sloupcích) lze jednoduchým způsobem rozšifrovat.

Uvažujme nyní Vigenerovo šifrování *s dlouhým klíčovým slovem*. Budeme předpokládat, že klíčové slovo je dlouhé právě tak, jak je délka zprávy. Ukážeme dva triky, které Mr. X znemožní účinně využít výše uvedených testů.

Trik č.1 Mohli bychom se pokusit, použít jako klíč text knihy. Takový klíč má zcela určitě tu výhodu, že ho lze přenést bez velkých problémů. Např. stačí podat příjemci informaci *Eugen Eichhorn: Felix Hausdorff - Paul Mongré*, aby mohl začít dešifrovat kryptogram pomocí následujícího slova:

> *As you have already heard, Hausdorff was born in the Silesian metropolis Breslau, today called Wroclaw. In the last days of the Second World War, the German Wehrmacht declared Breslau a fortress; the result was its complete destruction. That happened ...*

V případě použití takovéhoto klíče se všechny metody na určení délky klíče minou účinkem. Protože však klíč tvoří souvislý text nějaké řeči (angličtina, němčina, apod.), působí na kryptogram statisticky signifikantní data, takže nemůžeme takovouto šifru označit za zcela bezpečnou. První, kdo odhalil tuto slabinu, byl opět Friedman. Proto půjdeme ještě o kus dál.

Trik č.2 V případě triku č.1 mohl Mr. X ještě použít nějakou statistiku v důsledku tvaru klíčového slova. Proto nyní zvolíme za klíčové slovo prakticky nekonečnou, náhodnou posloupnost písmen, na kterou si se statistickými testy nepřijdeme. Např. lze za ni zvolit výsledky opakování vrhu ideální 26-hranou kostkou. Lze pak ukázat, že takovýto způsob šifrování je **dokonce teoreticky bezpečný!** Jinak řečeno: nabízí nám **perfektní bezpečnost**.

Takovýmito perfektními systémy se budeme zabývat v následující kapitole.

Kapitola 3

Dopřejme si jistoty neboli trochu teorie

Mnozí lidé používají svoji inteligenci k zjednodušení, mnozí k zesložitění.
(Erich Kästner)

V této kapitole se pokusíme vytvořit teoretický základ pro naše dřívější úvahy. Zejména vymezíme pojem „perfektní bezpečnosti“ šifrovacího systému.

3.1 Šifrovací systémy

Podle našich předchozích představ si dohodnou odesílatel a příjemce *nějaký* klíč a zašifrují s ním zprávu. Správný pohled na věc se od této představy jemně odlišuje. Budeme nyní uvažovat *systémy*, které sestávají z nějaké *množiny* zpráv, příslušných kryptogramů a klíčů. V případě, že bychom následující myšlenky chtěli provést zcela precizně, museli bychom se držet axiomatiky; pro naše účely však bude lepší vysvětlení pojmů pomocí typických příkladů.

Takovýmto typickým příkladem množiny zpráv je sbírka matematických knih v knihovně sekce matematika týkajících se kódování. Získáme pak šifrovací systém, pokud budeme navíc uvažovat všech 312 afinních šifer s příslušnými kryptogramy. Pro jiný případ stačí vzít všechna slova cizího původu vyskytující se v tomto textu, všech 26 aditivních posouvacích šifrování a výsledné kryptogramy.

Zaveďme následující označení. Pomocí písmene $\mathbf{M}$ (messages) označíme množinu všech zpráv, $\mathbf{C}$ (cryptogram) množinu všech kryptogramů (obvykle se jedná o řetězce nad konečnými abecedami Σ_1 a Σ_2) a $\mathbf{K}$ (key) množinu všech klíčů. Šifrovacím systémem (kryptosystémem) pak nazýváme trojici $\langle \mathbf{M}, \mathbf{K}, \mathbf{C} \rangle$ spolu s dodatečným předpokladem, že existují funkce (neboli algoritmy) e a d takové, že

$$e : \mathbf{M} \times \mathbf{K} \to \mathbf{C} \qquad \text{a} \qquad d : \mathbf{C} \times \mathbf{K} \to \mathbf{M}$$

a že pro všechna $(M, K) \in \mathbf{M} \times \mathbf{K}$ platí:

$$d(e(M, K), K) = M.$$

Zejména tedy pro každý klíč K máme invertibilní funkci (transformaci) $f_K : \mathbf{M} \to \mathbf{C}$ tak, že $f_K(M) = e(M, K)$ a ${f_K}^{-1}(f_K(M)) = M$. Systém $(f_K)_{K \in \mathbf{K}}$ je nazýván šifrovací algoritmus.

Výše uvedená definice byla formulována praotcem moderní kryptografie Claudem E. **Shannonem**.

Uveďme dvě triviální pozorování:

1. Je možné, že dvě různé transformace převádí tutéž zprávu na jednu transformaci.

2. Skutečnost, že transformace je invertibilní, implikuje $|\mathbf{M}| \leq |\mathbf{C}|$.

3.2 Perfektní bezpečnost

Nyní víme, co je šifrovací systém. Těžištěm tohoto odstavce je podání definice a popisu bezpečného šifrovacího systému.

Intuivně řečeno znamená „perfektní bezpečnost“, že Mr. X nemá žádnou šanci zvětšit své znalosti o systému, i kdyby měl k dispozici všechno vědění a všechnu počítačovou kapacitu světa.

Předpokládejme nyní, že máme šifrovací systém $\langle \mathbf{M}, \mathbf{K}, \mathbf{C} \rangle$ a že
(a) p_i je pravděpodobnost, že je odeslána zpráva M_i, $1 \leq i \leq n = |M|$; tyto pravděpodobnosti se nazývají **a priori** (nebo **teoretické**) **pravděpodobnosti** a jsou přirozeně každému dobrému kryptoanalytikovi známy.
(b) pravděpodobnost, že je použit klíč K_j je k_j a výběr klíče nezávisí na zprávě, která je přenášena.

Tyto dvě rozdělení pravděpodobností indukují rozdělení pravděpodobností na množině možných kryptogramů, kde pro jistý kryptogram C, řekněme C_u, je pravděpodobnost toho, že „náhodný“ kryptogram C je roven kryptogramu C_u, je určena vztahem

$$P(C = C_u) = \sum p_i k_j,$$

kde v sumě na pravé straně sčítáme přes všechny dvojice zpráva-klíč (M_i, K_j) takové, že $e(M_i, K_j) = C_u$.

Výše uvedené lze přeformulovat následovně pomocí pojmu náhodné veličiny. Mějme tři náhodné veličiny M, K, C tak, že jev $M = M_i$ znamená, že byla odeslány zpráva $M_i \in \mathbf{M}$, jev $K = K_j$ znamená, že byl pro šifrování vybrán klíč $K_j \in \mathbf{K}$ a jev $C = C_u$ znamená, že byl zachycen kryptogram $C_u \in \mathbf{C}$. Zejména tedy $P(M = M_i) = p_i$ a $P(K = K_j) = P(K = K_j | M = M_i)$ pro všechan i a všechna k.

Připomeňme, že výše uvedená situace je analogická situaci v teorii kódování pro případ zdroje bez paměti. Přitom považujeme zdroj za proud symbolů jisté konečné abecedy. Zdroj má obvykle nějaký náhodný mechanismus, který je založen na statistice situace, která je modelovaná. Tento náhodný mechanismus může být poměrně dost komplikovaný, ale my se budeme pro okamžik soustředit na následující opravdu speciální a jednoduchý příklad. Značí-li X_i i-tý symbol vytvořený zdrojem, dohodneme se pak, že, pro každý symbol a_j, pravděpodobnost

$$P(X_i = a_j) = p_j$$

je nezávislá na i a tedy je nezávislá na všech minulých nebo v budoucnosti vyslaných symbolech. Jinak řečeno, $X_1, X_2, \ldots$ je právě posloupnost identicky distribuovaných, nezávislých náhodných veličin. Takovýto zdroj nazveme *zdrojem s nulovou pamětí* nebo *zdrojem bez paměti* a jeho entropie H je definována jako

$$H = -\sum p_j \log p_j,$$

kde sčítáme přes množinu j takových, že $p_j > 0$.

Připomeňme, že jsou-li $X_1, \ldots, X_m$ náhodné proměnné takové, že každá z nich nabývá pouze konečně mnoha hodnot, lze pak považovat $\mathbf{X} = (X_1, \ldots, X_m)$ za náhodný vektor a definovat souhrnou entropii $X_1, \ldots, X_m$ jako

$$H(X_1, \ldots, X_m) = H(\mathbf{X}) = -\sum_k p(x_1, \ldots, x_m) \cdot \log_2 p(x_1, \ldots, x_m), \qquad (3.1)$$

kde $p(x_1, \ldots, x_m) = P(X_1 = x_1, X_2 = x_2, \ldots, X_m = x_m)$.

Předpokládejme dále, že X je náhodná proměnná na pravděpodobnostním prostoru Ω a A je událost z Ω. Nabývá-li X konečné množiny hodnot $\{a_i : 1 \leq i \leq m\}$, je přirozené definovat *podmíněnou entropii* náhodné proměnné X určenou událostí A jako

$$H(X|A) = -\sum_{k=1}^{m} P(X = a_k|A) \log P(X = a_k|A).$$

Úplně stejně, je-li Y jiná náhodná proměnná nabývající hodnot b_k $(1 \leq k \leq m)$, definujeme *podmíněnou entropii* náhodné proměnné X určenou náhodnou proměnnou Y jako

$$H(X|Y) = \sum_j H(X|Y = b_j) P(Y = b_j).$$

Považujeme $H(X|Y)$ za entropii náhodné proměnné X určenou jistou hodnotou Y zprůměrovanou přes všechny hodnoty, jichž může Y nabývat.

Diskrétní kanál bez paměti je charakterizován vstupní abecedou $\Sigma_1 = \{a_1, \ldots, a_m\}$ vstupních znaků, výstupní abecedou $\Sigma_2 = \{b_1, \ldots, b_n\}$ výstupních znaků a *maticí*

P kanálu

$$P = \begin{pmatrix} p_{11} & p_{12} & \cdots & \cdots & p_{1n-1} & p_{1n} \\ p_{21} & p_{22} & \cdots & \cdots & p_{2n-1} & p_{2n} \\ \vdots & \vdots & \ldots & \ldots & \vdots & \vdots \\ p_{m-11} & p_{m-12} & \cdots & \cdots & p_{m-1n-1} & p_{m-1n} \\ p_{m1} & p_{m2} & \cdots & \cdots & p_{mn-1} & p_{mn} \end{pmatrix}.$$

Způsob používání kanálu je následující: každá posloupnost $(u_1, u_2, \ldots, u_N)$ symbolů ze vstupní abecedy Σ_1 na vstupu se převede na posloupnost $(v_1, v_2, \ldots, v_N)$ téže délky symbolů z výstupní abecedy Σ_2 na výstup tak, že

$$P(v_k = b_j | u_k = a_i) = p_{ij} \quad (1 \leq i \leq m, 1 \leq j \leq n),$$

a to nezávisle pro každé k.

Implicitně je ve výše uvedeném obsaženo, že pro každé i, $1 \leq i \leq m$ platí

$$\sum_j p_{ij} = 1.$$

Matice P s nezápornými hodnotami taková, že součet prvků v každém řádku je roven 1, se nazývá *stochastická matice*; v teorii náhodných procesů mluvíme o matici přechodu markovského řetězce.

Kapacita komunikačního kanálu je míra jeho schopnosti přenášet informaci. Formální definice je motivována níže uvedeným:

Předpokládejme, že máme diskrétní kanál bez paměti se vstupní abecedou $\Sigma_1 = \{a_1, \ldots, a_m\}$, výstupní abecedou $\Sigma_2 = \{b_1, \ldots, b_n\}$ a maticí P kanálu

$$P = [p_{ij}] = P(\mathbf{b}_j \text{ obdrženo} | \mathbf{a}_i \text{ odesláno}).$$

Přidáme-li k tomuto kanálu zdroj $\mathcal{S}$ bez paměti, který vysílá symboly $a_1, \ldots, a_m$ s pravděpodobnostmi $p_1, \ldots, p_m$, pak výstup kanálu můžeme považovat za zdroj $\mathcal{J}$ bez paměti, který vysílá symboly $b_1, \ldots, b_n$ s pravděpodobnostmi $q_1, \ldots, q_n$, kde

$$\begin{aligned} q_j &= \textstyle\sum_{i=1}^m P(\mathbf{b}_j \text{ obdrženo} | \mathbf{a}_i \text{ odesláno}) P(\mathbf{a}_i \text{ odesláno}) \\ &= \textstyle\sum_{i=1}^m p_i p_{ij}. \end{aligned}$$

Jsou-li $\mathbf{U}$ a $\mathbf{V}$ dva náhodné vektory, definujeme *informaci o* $\mathbf{U}$ *poskytnutou* $\mathbf{V}$ jako číslo

$$I(\mathbf{U}|\mathbf{V}) = H(\mathbf{U}) - H(\mathbf{U}|\mathbf{V}).$$

Jinak řečeno, $I(\mathbf{U}|\mathbf{V})$ vyjadřuje množství nejistoty o $\mathbf{U}$ odstraněné $\mathbf{V}$. Totiž, množství informace, průměrně obsažené v jednom znaku zprávy, je entropie vstupního rozdělení $H(\mathcal{S}) = -\sum_i p_i \log p_i$. Při přenosu diskrétním kanálem se ztratí informace

$$H(\mathcal{S}/\mathcal{J}) = -\sum_{i,j} p_{i,j} \log p_{i,j}$$

průměrně na jeden znak. Zbývá pak $H(\mathcal{S})-H(\mathcal{S}/\mathcal{J})$ přenesené informace. Jestliže entropii počítáme v bitech a známe průměrnou dobu τ, kterou kanál spotřebuje na přenos jednoho znaku, je rychlost přenosu

$$I(\mathcal{S}|\mathcal{J}) = \frac{H(\mathcal{S}) - H(\mathcal{S}/\mathcal{J})}{\tau} \text{ bitů za sekundu.}$$

Často se za jednotku času volí jeden přenos znaku a potom

$$I(\mathcal{S}|\mathcal{J}) = H(\mathcal{S}) - H(\mathcal{S}/\mathcal{J}) \text{ bitů za jednotku času.}$$

Informace o $\mathcal{S}$ podaná pomocí $\mathcal{J}$ je pak rovna

$$I(\mathcal{S}|\mathcal{J}) = H(\mathcal{S}) - H(\mathcal{S}|\mathcal{J}) = H(\mathcal{S}) + H(\mathcal{J}) - H(\mathcal{S},\mathcal{J})$$

a je to funkce, která závisí pouze na pravděpodobnostním rozdělení $q_1,\dots,q_n$, a maticí kanálu P. Proto je přirozené definovat *kapacitu* C kanálu jako maximální rychlost přenosu, tedy

$$C = \sup I(\mathcal{S}|\mathcal{J}), \tag{3.2}$$

kde supremum je bráno přes všechny zdroje bez paměti $\mathcal{S}$, nebo, ještě přesněji, nad všemi možnými rozděleními pravděpodobností $(p_1,\dots,p_n)$.

V dalším tedy můžeme považovat $\mathbf{M}$ za zdroj bez paměti s šifrovací funkcí e, přičemž klíče slouží jako komunikační kanál.

Základním pojmem je pojem *klíčové ekvivokace* zavedený Shannonem $H(K|C)$. ten nám měří průměrnou nejistotu, která nám zůstává po zachycení kryptogramu C. Podobně budeme definovat *ekvivokaci zpráv* jakožto $H(M|C)$. Občas budeme psát $\mathsf{S} = \langle \mathbf{M}, \mathbf{K}, \mathbf{C} \rangle$ a budeme značit $H(\mathsf{S})$ klíčovou ekvivokaci $H(K|C)$.

Náhodná proměnná zpráv má pak *entropii zpráv*

$$H(M) = -\sum p_i \log p_i,$$

a, jsou-li po řadě K a C náhodné proměnné klíčů a kryptogramů, jsou pak *klíčová entropie* $H(K)$ a *entropie kryptogramů* definovány jakožto

$$\begin{aligned} H(K) &= -\sum P(K=K_i)\log P(K=K_i), \\ H(C) &= -\sum P(C=C_j)\log P(C=C_j), \end{aligned}$$

kde sumace je prováděna přes všechny možné klíče K_i a všechny možné kryptogramy C_j.

Následující vlastnost ekvivokace vyjadřuje tu skutečnost, že daleko více nejistoty je spjato s klíčem než se zprávou.

Věta 3.2.1 *Klíčová ekvivokace je určena ekvivokací zprávy vztahem*

$$H(K|C) = H(M|C) + H(K|M,C).$$

Důkaz. Připomeňme základní identitu pro entropii

$$H(X|Y) = H(X,Y) - H(Y).$$

Můžeme tedy psát

$$\begin{aligned} H(M|C) &= H(M,C) - H(C) \\ &= H(M,K,C) - H(K|M,C) - H(C). \end{aligned}$$

Nyní tedy i

$$\begin{aligned} H(K|C) &= H(K,C) - H(C) \\ &= H(M,K,C) - H(M|K,C) - H(C). \end{aligned}$$

Ale

$$H(M|K,C) = 0,$$

protože jakmile je znám kryptogram C a klíč K, je jednoznačně určena i zpráva M a tedy míra neurčitosti je nulová. Tedy

$$H(K|C) = H(M,K,C) - H(C),$$

což, porovnáno s výše uvedeným, nám dává dokazovanou identitu.

Důsledek 3.2.2 *Klíčová ekvivokace je alespoň tak velká jako ekvivokace zprávy.*

Lemma 3.2.3 *Pro každý kryptosystém* $\langle \mathbf{M}, \mathbf{K}, \mathbf{C} \rangle$ *platí*

$$H(K,C) = H(M) + H(K).$$

Důkaz. Protože náhodné veličiny K a M jsou nezávislé, víme, že $H(M) + H(K) = H(M,K)$. Stačí tedy ověřit, že $H(C,K) = H(M,K)$ tj. $H(C|K) = H(M|K)$. Poznamenejme, že stačí pro pevný klíč K_j ověřit, že $H(C|K = K_j) = H(M|K = K_j)$. Položme **C'** $= \{C_u \in \mathbf{C}$: existuje $M_i \in \mathbf{M}$ tak, že $C_u = e(M_i, K_j)\}$. Pak evidentně $H(C|K = K_j) = H(C'|K = K_j)$, protože pro kryptogramy neležící v **C'** je odpovídající podmíněná pravděpodobnost nulová. Připomeňme známý fakt z teorie informace

$$H(\mathbf{U}|\mathbf{V}) = 0 \text{ právě tehdy, když } \mathbf{U} = g(\mathbf{V}) \text{ pro nějakou funkci } g. \tag{3.3}$$

Zřejmě, protože $f_{K_j} : \mathbf{M} \to$ **C'** tak, že $f_{K_j}(M_i) = e(M_i, K_j)$ a ${f_{K_j}}^{-1}(f_{K_j}(M_i)) = M_i$, je f_{K_j} bijekce. Speciálně, $C'|K = K_j$ je funkcí $M|K = K_j$ a obráceně $M|K = K_j$ je funkcí $C'|K = K_j$. Lehkou úpravou pak obdržíme, že $H(M|K = K_j) = H(C'|K = K_j)$, tj. $H(C|K = K_j) = H(M|K = K_j)$.

Důsledek 3.2.4 *Pro každý kryptosystém* $\langle \mathbf{M}, \mathbf{K}, \mathbf{C} \rangle$ *platí*

$$H(C|K) = H(M|K),\ \ H(C,K) = H(M,K) \text{ a } H(M) \leq H(C).$$

Důkaz. Stačí ověřit poslední nerovnost. $H(M)+H(K) = H(M,K) = H(C,K) \leq H(C) + H(K)$ tj. $H(M) \leq H(C)$.

Příklady:

(a) Předpokládejme, že „zprávy" v $\mathbf{M}$ jsou *písmena* slov nějaké (německé) knihy; pravděpodobnost zprávy je pak četnost odpovídajícího písmene; v případě písmene **e** je pak $p(\mathbf{e}) \approx 0.174$.

(b) Nyní předpokládejme, že „zprávy" v $\mathbf{M}$ jsou *dvojice za sebou následujících písmen* slov nějaké (německé) knihy; pravděpodobnost zprávy je pak četnost odpovídajícího bigramu.

Představme si nyní, že Mr. X zachytil kryptogram C. Aby ho byl schopen analyzovat, může (alespoň teoreticky) vyzkoušet všechny zprávy a vždy určit pravděpodobnost toho, že kryptogram C vznikl zašifrováním zprávy M. Označme pak tyto pravděpodobnosti $p_C(M) = P(M|C)$; mluvíme pak o **a posteriori** (nebo **pozorovaných**) **pravděpodobnostech**.

Příklady:

(c) Buď M stejné jako ve výše uvedeném příkladu (a); jako algoritmus budeme uvažovat posouvací šifry se všemi 26 možnými klíči. Uvažme, že každé písmeno kryptogramu C má tutéž šanci, že odpovídá určitému písmenu zprávy; např. v 17.4% případů vznikne C z **e**, v 9.8% případů vznikne z **n**, atd. Jinak řečeno, pro každý kryptogram C platí $p_C(M) = p(M)$ pro každou zprávu M.

(b) Nyní předpokládejme, že každá zpráva v $\mathbf{M}$ sestává z prvních 100 písmen každé strany prvního dílu slovníku *das große Brockhaus*. Algoritmus nechť opět sestává z posouvacích šifer. Pro každou zprávu M je její pravděpodobnost $\frac{1}{|M|}$, malé, ale stále ještě kladné číslo. Protože je relativně snadné prověřit, zda určitý kryptogram pochází z určité zprávy (rozdělení písmen v kryptogramu musí přesně odpovídat rozdělení písmen ve zprávě), je $p_C(M)$ rovno buď jedné nebo nule. To znamená obzvlášť, že pro každý kryptogram je $p_C(M) \neq p(M)$.

Proberme tuto skutečnost podrobněji: Předpokládejme, že kryptoanalytik Mr. X zjistí. že pro jistou zprávu M je $p_C(M) > p(M)$. Pak by věděl, že kryptogram C vznikl s vysokou pravděpodobností ze zprávy M. Tzn., že by se analýzou něco nového naučil. To ale nesmí při perfektním systému nastat. V případě, že by bylo $p_C(M) < p(M)$, pak by Mr. X věděl, že kryptogram C vznikne s velmi malou pravděpodobností ze zprávy M. I v tomto případě by si Mr. X rozšířil svoje znalosti.

Můžeme tedy definovat:
Šifrovací systém $\langle \mathbf{M}, \mathbf{K}, \mathbf{C} \rangle$ *poskytuje* **perfektní bezpečnost**, *jestliže pro každý kryptogram* C *platí*

$$p_C(M) = p(M)$$

pro každou zprávu M.

Jinak řečeno, $\langle \mathbf{M}, \mathbf{K}, \mathbf{C} \rangle$ je perfektní, pokud jsou a priori pravděpodobnosti rovny pravděpodobnostem a posteriori (viz příklad (c)). *Posouvací šifry jsou perfektní, jestliže operují nad jednotlivými písmeny.* Mr. X se pak může namáhat jak chce; písmena kryptogramu jsou totiž zcela náhodně rozdělena.

Chceme-li perfektnost šifrovacího systému vyjádřit pomocí náhodných proměnných M a C, je systém perfektní právě tehdy, když M a C jsou nezávislé. Z toho bezprostředně plyne, že šifrovací systém $\langle \mathbf{M}, \mathbf{K}, \mathbf{C} \rangle$ poskytuje **perfektní bezpečnost**, jestliže pro každou zprávu M platí

$$p_M(C) = p(C)$$

pro každý kryptogram C.

*

Věta 3.2.5 *Kryptosystém* $\langle \mathbf{M}, \mathbf{K}, \mathbf{C} \rangle$ *je perfektní právě tehdy, když*

$$H(M|C) = H(M).$$

Důkaz. Z teorie informace je známo, že $H(M|C) = H(M)$ právě tehdy, když M a C jsou nezávislé náhodné proměnné tj. to je právě tehdy, když se jedná o perfektní kryptosystém.

Ptejme se nyní, jak můžeme rozpoznat, kdy je šifrovací systém perfektní či nikoliv. K tomu si dokážeme několik jednoduchých kritérií.

1. Kritérium *Je-li šifrovací systém* $\langle \mathbf{M}, \mathbf{K}, \mathbf{C} \rangle$ *perfektní, pak každá zpráva s odpovídajícím klíčem může být zobrazena na libovolný kryptogram.*

Proč platí toto kritérium? Uvažme zprávu M a kryptogram C. Protože $\langle \mathbf{M}, \mathbf{K}, \mathbf{C} \rangle$ je perfektní, platí $p_C(M) = p(M)$. V každém šifrovacím sytému je $p(M) > 0$, protože každá zpráva se vyskytuje s kladnou pravděpodobností. Dohromady obdržíme $p_C(M) > 0$. To znamená, že existuje klíč, pomocí kterého se zašifruje M do C.Tím je dokázáno první kritérium.

Toto kritérium je velmi užitečné - v „negativním smyslu“: Umožní nám rozhodnout, že jisté systémy *nejsou* perfektní.

2. Kritérium *Je-li šifrovací systém* $\langle \mathbf{M}, \mathbf{K}, \mathbf{C} \rangle$ *perfektní, pak platí:*

$$|\mathbf{M}| \leq |\mathbf{C}| \leq |\mathbf{K}|.$$

Důkaz. Zřejmě $|\mathbf{M}| \leq |\mathbf{C}|$. Proč platí $|\mathbf{C}| \leq |\mathbf{K}|$? Uvažme libovolnou, pevně zvolenou zprávu M a zašifrujme ji pomocí všech možných klíčů z $\langle \mathbf{M}, \mathbf{K}, \mathbf{C} \rangle$. Podle prvního kritéria lze M převést do každého možného kryptogramu. Pro každý kryptogram C potřebujeme alespoň jeden klíč (totiž pomocí jednoho klíče nemůžeme M zobrazit na dva různé kryptogramy). Potřebujeme tedy alespoň tolik klíčů, kolik je kryptogramů. Máme tedy $|\mathbf{C}| \leq |\mathbf{K}|$.

3. Kritérium *Buď $\langle \mathbf{M}, \mathbf{K}, \mathbf{C} \rangle$ šifrovací systém tak, že*

$$|\mathbf{M}| = |\mathbf{C}| = |\mathbf{K}|,$$

ve kterém se všechny klíče vyskytují s toutéž pravděpodobností. Dále předpokládejme, že ke každé zprávě M a ke každému kryptogramu C existuje právě jeden klíč K z $\langle \mathbf{M}, \mathbf{K}, \mathbf{C} \rangle$ tak, že $e(M, K) = C$. Pak je $\langle \mathbf{M}, \mathbf{K}, \mathbf{C} \rangle$ perfektní.

Důkaz. Stačí zřejmě ověřit, že pro každou zprávu M_i platí $P(M = M_i | C = C_j) = P(M)$ pro každý kryptogram C_j. Z Bayesova vzorce máme

$$\begin{aligned} P(M = M_i | C = C_j) &= \frac{P(C = C_j | M = M_i) \cdot P(M = M_i)}{\sum_{k=1}^{|K|} P(C = C_j | M = M_k) \cdot P(M = M_k)} \\ &= \frac{P(M = M_i)}{\sum_{k=1}^{|K|} P(M = M_k)} = P(M = M_i), \end{aligned}$$

neboť $P(C = C_j | M = M_k) = \dfrac{1}{|K|}$ nezávisle na j a k.

Přirozeným způsobem jak zvýšit bezpečnost šifrování je vzít různé systémy a kombinovat je. Dvě takovéto metody navržené Shannonem jsou stále základem mnoha praktických kryptosystémů. Jedná se o

- *Vážený součet*: Jsou-li $\mathbf{S}_1$ a $\mathbf{S}_2$ dva kryptosystémy se stejným prostorem zpráv $M = M_1 = M_2$ a $0 < p < 1$, je pak jejich *vážený součet* $p\mathbf{S}_1 + (1-p)\mathbf{S}_2$ kryptosystém určený následným výběrem: použijeme $\mathbf{S}_1$ s pravděpodobností p a $\mathbf{S}_2$ s pravděpodobností $1 - p$.

  Má-li tedy $\mathbf{S}_1$ klíče $K_1, \ldots, K_m$ s pravděpodobnostmi použití p_i pro klíč K_i a $\mathbf{S}_2$ má klíče $K'_1, \ldots, K'_n$ s pravděpodobnostmi použití p'_j pro klíč K'_j, má pak kryptosystém $p\mathbf{S}_1 + (1-p)\mathbf{S}_2$ $m + n$ klíčů $K_1, \ldots, K_m, K'_1, \ldots, K'_n$ s pravděpodobnostmi použití pp_i pro klíč K_i a s pravděpodobnostmi použití $(1-p)p'_j$ pro klíč K'_j. Tento postup lze přirozeně rpzšířit na více než dva systémy.

- *Součin*: Druhý způsob kombinování kryptosystémů $\mathbf{S}_1$ a $\mathbf{S}_2$ je to, že nejprve použijeme na naši zprávu kryptosystém $\mathbf{S}_1$ a potom aplikujeme $\mathbf{S}_2$ na výsledný kryptogram. Abychom toto mohli provést, musí být nutně $\mathbf{C}_1 \subseteq \mathbf{M}_2$. Pak můžeme definovat součin jako $\mathbf{S}_1 * \mathbf{S}_2$.

Jsou-li klíče $K_1, \ldots, K_m$ s pravděpodobnostmi použití p_i pro klíč K_i v kryptosystému $\mathbf{S}_1$ a $\mathbf{S}_2$ má klíče $K'_1, \ldots, K'_n$ s pravděpodobnostmi použití p'_j pro klíč K'_j, má pak kryptosystém $\mathbf{S}_1 * \mathbf{S}_2$ $m.n$ klíčů (K_i, K'_j) s pravděpodobnostmi použití $p_i p'_j$. Poznamenejme, že skutečně efektivních klíčů může být méně, protože některé se složených transformací mohou splývat.

Poznamenejme, že evidentně platí následující:

Jsou-li $\mathbf{S}_1$, $\mathbf{S}_2$ a $\mathbf{S}_3$ kryptosystémy tak, že níže uvedené operace jsou definovány, $0 < p < 1$, $q = 1 - p$, pak

$$\mathbf{S}_3 * (\mathrm{p}\mathbf{S}_1 + \mathrm{q}\mathbf{S}_2) = p\mathbf{S}_3 * \mathbf{S}_1 + q\mathbf{S}_3 * \mathbf{S}_2,$$
$$(\mathrm{p}\mathbf{S}_1 + \mathrm{q}\mathbf{S}_2) * \mathbf{S}_3 = p\mathbf{S}_1 * \mathbf{S}_3 + q\mathbf{S}_2 * \mathbf{S}_3,$$
$$\mathbf{S}_1 * (\mathbf{S}_2 * \mathbf{S}_3) = (\mathbf{S}_1 * \mathbf{S}_2) * \mathbf{S}_3$$
$$\mathbf{S}_1 * \mathbf{S}_2 \text{ není obecně rovno } \mathbf{S}_2 * \mathbf{S}_1.$$

3.3 Redundance přirozeného jazyka a bod unicity

Věnujme se nyní chvíli zkoumání přirozeného jazyka jakým je například angličtina. Budeme v dalším považovat angličtinu za jazyk skládající se z abecedy o 27 písmenech, z toho je 26 římských písmen a 1 mezera. První, a velmi špatná aproximace angličtiny je, že vezmeme *aproximaci 0. řádu.* V tomto případě mají všechny symboly stejnou pravděpodobnost: každý se tedy vyskytne s pravděpodobností $\frac{1}{27}$ a následující text nám ukáže typickou sekvenci symbolů vytvořenou takovýmto zdrojem:

DM QASCJDGFOZYNX ZSDZLXIKUD.

Tato aproximace vůbec nevyužívá relativní četnosti symbolů použitých v anglickém jazyce. Použijeme-li odhady těchto četností, můžeme vytvořit *aproximaci 1. řádu*, jejímž typickým příkladem je

OR L RW NILI E NNSBATEI.

Ačkoliv je tento přístup zřejmější než aproximace 0. řádu, stále zde není žádná informace o vzájemné závislosti sousedních písmen. Tomuto lze vyhovět například *Markovovým zdrojem 1. řádu*, kde můžeme použít podmíněné pravděpodobnosti založené na četnostech dvojic písmen tj. *digramů*:

$$P(i|j) = p(i,j)/p(j),$$

kde $p(i,j)$ je pravděpodobnost výskytu digramu (i,j) a $p(i|j)$ je podmíněná pravděpodobnost výskytu písmene i za předpokladu, že předcházející písmeno je j.

To je však velmi časově náročné a Shannon místo toho navrhl použít metodu Monte Carlo, která má stejný efekt.

Vyberme náhodně text či texty. Náhodně z textu vyberme první písmeno jakožto první symbol X_1. Předpokládejme bez újmy na obecnosti, že je to např. B. Opět náhodně nalistujme nějakou stránku textu a pokračujme na ní dále, až narazíme na první výskyt B. Vezměme za X_2 písmeno textu bezprostředně za B. Použijeme-li výše uvedené metody, lze obdržet *Markovovu aproximaci 1. řádu pro angličtinu*:

OUCTIE IN ARE AMYST TE TUSE SOBE CTUSE.

Shannonovu metodu lze použít na to, abychom získali lepší aproximaci tak, že vybereme písmena z textu vzhledem k dvěma předchozím písmenům. Např., *Markovovou aproximací druhého řádu* je posloupnost

HE AREAT BEIS THAT WISHBOUT SEED DAY OFTE, AND HE IS FOR THAT MINUMB LOOTS WILL AND GIIRLS, A DOLL WILL IS FRIECE ABOARICE STRED SAYS.

Použijeme-li Shannonovu metodu s Cicerovým dílem *de Senectute*, obdrží velmi zřetelnou Markovovu aproximaci latiny:

IENEC FES VIMONILLITUM M ST ER PEM ENIM PTAUL

(Markovova aproximace 1. řádu)

SENECTOR VCI QUAEMODOMIS SE NON
FRATURDIGNAVIT SINE VELIUS

(Markovova aproximace 2. řádu).

Teoreticky může být tato metoda použita pro aproximace libovolně vysokého řádu. Je však více než namáhavé provádět už aproximace třetího řádu. Lze však akceptovat to, že už aproximace druhého řádu už je přijatelná.

Alternativní přístup navržený Shannonem bylo modelování angličtiny nikoliv jako zdroje písmen, nýbrž jako zdroje s množinou anglických *slov*, jakožto základní abecedou. Shannon dává přednost náhodnému výběru z textů před metodou četnosti anglických slov. Uveďme následující aproximace:

REPRESENTING AND SPEEDILY IS AN GOOD APT OR COME CAN DIFFERENT NATURAL HERE HE THE A IN CAME THE TO OF THE EXPERT GRAY COME TO FURNISHES THE LINE MESSAGE HAD BE THESE

(slovní aproximace 1. řádu)

THE HEAD AND IN FRONTAL ATTACK ON AN ENGLISH WRITER THAT THE CHARACTER OF THIS POINT IS THEREFORE ANOTHER METHOD FOR THE LETTERS THAT THE TIME OF WHOEVER TOLD THE PROBLEM FOR AN UNEXPECTED.

(Markovova aproximace 2. řádu).

Budeme tedy v dalším považovat přirozené jazyky za zdroje s entropií. Pokusíme se podat jisté odhady a interpretace této entropie, kterou budeme v případě angličtiny značit jako H_E. Je známo, že lze H_E interpretovat pomocí přibližné formule

$$2^{nH_E} \simeq T(n) \qquad (n \text{ dostatečně velké}),$$

kde $T(n)$ označuje počet typických (=smysluplných) posloupností délky n anglického jazyka. Tento vztah nám však bezprostředně nepomůže s odhadem H_E, protože není znám žádný způsob odhadu $T(n)$. Víme však, že existuje 27^n možných posloupností délky n z anglické abecedy a protože $\log 27 = 4.76$, máme

$$H_E \leq 4.76 \text{ bitů na symbol.}$$

Lepší odhad pro H_E můžeme obdržet z aproximace 1. řádu, ve které můžeme použít informaci o rozdílných pravděpodobnostech výskytu písmen. Například nejpravděpodobnějším symbolem je mezera s pravěpodobností $P(\text{mezera}) = 0.18\ldots$, $P(E) = 0.13\ldots$ atd.

Použijeme-li základní identitu

$$H(X,Y) \leq H(X) + H(Y),$$

dostaneme horní závoru

$$H_E \leq H_E^1 \leq -\sum_{p_i} \log p_i,$$

kde p_i je pravděpodobnost výskytu i-tého symbolu . Podobně obdržíme

$$H_E \leq H_E^2 = -\frac{1}{2}\sum_i \sum_j p(i,j) \log p(i,j),$$

kde $p(i,j)$ jsou odhadnuté výskyty symbolů (i,j) a my ignorujeme dvojice symbolů s nulovou pravděpodobností (Qq).

Následující tabulka nám ukazuje přehled odhadů entropií pro 26- a 27-písmenou anglickou abecedu.

	26-ti písmenná abeceda	27-ti písmenná abeceda
H_E^0	4.70	4.76
H_E^1	4.14	4.03
H_E^2	3.56	3.32
H_E^3	3.30	3.10

Jiný přístup nalezení odhadu entropie je založený na četnosti slov. Považujme angličtinu za konečný jazyk skládající se ze slov $w_1, \ldots, w_N$, jež se vyskytují nezávisle s pravděpodobnostmi $p_1, \ldots, p_N$. Pak *slovní entropie* H_W je určena vztahem

$$H_W = -\sum_{i=1}^{N} p_i \log p_i.$$

Shannon navrhl, že pak entropii symbolů H_E lze aproximovat jakožto

$$H_E = H_W / \overline{w},$$

kde $\overline{w}$ je průměrná délka slova v angličtině. K tomuto přístupu lze mít následující výhrady:

- Slova použitá v angličtině nejsou nezávislá a slovní entropie je spíše odhad slovní entropie prvního řádu.
- Podíl slovní entropie a průměrné délky slova je velmi hrubá aproximace a je nejlépe ji nahradit vhodnou nerovností.

Pokračujme však dále ve výše uvedeném přístupu. Abychom vyčíslili slovní entropii, použijme pravidlo navržené linguistou G. K. Zipfem (1935). To tvrdí, že, pokud jsou slova přirozeného jazyka uspořádáná v klesajícím uspořádání podle jejich pravděpodobností výskytu (p_n pak označuje pravděpodobnost $n-$tého nejvýše pravděpodobného slova), dobrá aproximace těchto pravděpodobností je určena formulí

$$p_n = A/n,$$

kde A je nějaká konstanta závisející na daném jazyce.

Ačkoliv Zipfův přístup byl kritizován, jeho pravidlo dobře funguje pro tak různé jazyky jako je hebrejština, starogermánština, křováčtina a norština. Shannon použil Zipfovo pravidlo jakožto aproximaci pro angličtinu s konstantou $A = 0.1$ a počtem slov $M = 12366$. Pak

$$\sum_{n=1}^{12366} p_n = 0.1 \sum_{n=1}^{12366} \frac{1}{n} = 1,$$

pak je slovní entropie $H_w = 9.72$ bitů na slovo. Protože střední délka $\overline{w}$ anglických slov je 4.5, obdržíme odhad pro $H_{4.5} \simeq 9.72/4.5 = 2.16$ bitů na písmeno.

Proveďme nyní následující výpočet:

$$H_W = \sum_{k=1}^{\infty} H(W : |W| = k) P(|W| = k),$$

kde W je „náhodný“ slovní výstup a $|W|$ označuje délku (nebo počet písmen) W. Tedy

$$\begin{aligned} H_W &= \textstyle\sum_{k=1}^{\infty} H(X_1X_2\dots X_k)P(|W|=k) \\ &\le \textstyle\sum_{k=1}^{\infty} kH(X)P(|W|=k), \end{aligned}$$

kde $H(X)$ je entropie symbolů H_E a nerovnost bezprostředně vyplývá ze základní identity

$$H(U,V) \le H(U) + H(V).$$

Máme pak $H_W \le H_{4.5} \sum_{k=1}^{\infty} kH(X)P(|W|=k)$, tj.

$$H_W \le H_{4.5}\overline{w}.$$

Je známo, že zdroj s entropií H má v abecedě $|\Sigma|$ jednoznačně dešifrovatelné zakódování s minimální průměrnou délkou slova $l(n)$ typického řetězce o n symbolech, přičemž

$$l(n) = nH/\log|\Sigma|.$$

Představíme-li si redundanci jako míru zbytečných symbolů (v procentech), je přirozené ji definovat následujícím přirozeným způsobem:

$$l(n) \simeq n(1 - R/100).$$

Z výše uvedeného pak obdržíme

$$R = 1 - H/\log_2|\Sigma|,$$

uvažujeme-li redundanci jako číslo mezi 0 a 1. Poslední vztah bude pro nás oficiální definicí.

Přesný odhad redundance je obtížný; odhady zřejmě závisí na vybraném textu. Uveďme následující příklad

	Bible	Měsíčník
H_1	4.086	4.152
H_{12}	2.397	2.824
R	0.413	0.285
$\overline{w}$	4.060	4.653

Zajímavější je studium identických pasáží Bible přeložených do různých jazyků. Zatímco samojština je jazyk s pouze 16 písmeny, z nichž 60% tvoří samohlásky, ruština před rokem 1917 používala abecedu o 35 písmenech. Srovnání viz v následující tabulce:

	Angličtina	Ruština	Samojština
H_1	4.114	4.612	3.370
H_{12}	2.397	2.395	2.136
R	0.413	0.474	0.372
$\overline{w}$	4.060	5.296	3.174

Shannon odhadl, že entropii angličtiny lze redukovat na jeden bit na písmeno, což by nám dávalo řádově redundanci asi 75%. Tento odhad je však nutno interpretovat s jistou opatrností. Například výše uvedené neznamená, že můžeme rekonstruovat zprávu, ve které jsou písmena smazána s pravděpodobností $\frac{3}{4}$. Přesný způsob mazání je také důležitý. Jsou-li písmena smazána s pravděpodobností 0.5, pak například zpráva

MATHEMATICS IS BEAUTIFUL

může být obdržena ve tvaru

MTMASSBUFL;

a tedy by bylo opravdu obtížné získat zprávu pouze z narušeného textu. Bylo dokázáno, že kritická hodnota je $p \simeq 0.25$ a pro vyšší hodnotu je obdržení původní zprávy nemožné.

Jinak řečeno, ačkoliv je teoreticky možné zkrátit vytištěný text na čtvrtinu jeho současné délky, náhodné zkrácení není vhodný způsob, jak toho dosáhnout. Je nám ale jasné, že velkou redukci lze obdržet smysluplným zakódováním. Např., lze bez obtíží akceptovat pravdivost následujících tvrzení:

- Vynecháme-li nějaké písmeno z textu, lze ho zpětně zrekonstruovat.
- Vynecháme-li všechny samohlásky z textu, lze text zpětně zrekonstruovat.

Oba tyto případy jsou příklady suboptimálního zakódování a vezmeme-li redundanci angličtiny mezi 75% a 50%, dostaneme, že entropie H_E splňuje

$$1.19 \leq H_E \leq 2.38.$$

Dále buď dán kryptosystém $\langle \mathbf{M}, \mathbf{K}, \mathbf{C} \rangle$, položíme M_N a C_N pro „náhodné“ části zdrojového textu a odpovídajícího kryptogramu délky N. Nyní pak zřejmě

$$\begin{aligned} H(K|C_N) &= H(K, C_N) - H(C_N) \\ &= H(M_N, K, C_N) - H(C_N) \\ &= H(M_N, K) - H(C_N) \\ &= H(M_N) + H(K) - H(C_N). \end{aligned}$$

Definujeme pak *bod unicity* U jakožto

$$U := \min\{N > 0 : H(K|C_N) = 0\}, \text{ tj.}$$

$$H(M_U) + H(K) - H(C_U) = 0.$$

Předpokládejme, že platí následující:

1. Přirozený jazyk, ve kterém šifrujeme, má tu vlastnost, že je dán vhodný odhad $H(M_N)$ jako
$$H(M_N) \simeq NH,$$
kde H je entropie jednoho symbolu jazyka;

2. kryptosystém má tu vlastnost, že všechny sekvence délky N symbolů mají stejnou pravděpodobnost jakožto kryptogram; jinak řečeno
$$H(C_N) \simeq N\log|\Sigma|.$$

To není nevhodný požadavek: každý dobrý kryptosystém by měl mít tuto vlastnost. Z výše uvedeného obdržíme

$$UH + H(K) - U\log|\Sigma| = 0,$$

tj.
$$U = \frac{H(K)}{\log|\Sigma| - H}.$$

Obvykle se rovněž předpokládá, že každý klíč můžeme vybrat se stejnou pravděpodobností a to znamená, že

$$U = \frac{\log|\mathbf{K}|}{\log|\Sigma| - H},$$

kde H je entropie symbolu zdroje.

Připomeňme, že existuje těsný vztah mezi bodem unicity kryptosystému a redundancí jazyka, ve kterám se přenáší zpráva. Přitom redundance R jazyka s entropií H je určena vztahem

$$R = 1 - \frac{H}{\log|\Sigma|},$$

a tedy
$$U = \frac{\log|\mathbf{K}|}{\log|\Sigma| - H} == \frac{\log|\mathbf{K}|}{R\log|\Sigma|}.$$

Zejména pak má-li jazyk nulovou redundanci, je pro každý kryptosystém splňující 1 a 2 bod unicity nekonečno.

Příklad: Předpokládejme, že šifrujeme pomocí jednoduché substituce tak, že máme právě 26! klíčů. Uvažujeme-li $\log 26 = 4.7$ a entropii anglického jazyka H_E rovnu 2 bitům na symbol, obdržíme

$$U = \frac{\log 26!}{4.7 - 2} = \frac{88.4}{2.7} \simeq 32.$$

Jinak řečeno, při výše uvedené entropii anglického jazyka jsme obdrželi hodnotu bodu unicity rovnu 32 symbolům. To pak celkem souhlasí s empirickým pozorováním Shannona (1949), který tvrdí, že pro bod unicity

"lze ukázat, že leží mezi krajními body 20 a 30. S 30 písmeny existuje téměř vždy jediné řešení pro kryptogram tohoto typu a s 20 můžeme najít nějaký počet řešení. "

Podobným způsobem Friedman (1973) tvrdí, že

"Prakticky každý příklad 25 nebo více charakterů reprezentujících monoabecední zašifrování smysluplné zprávy v angličtině lze snadno vyřešit."

V roce 1977 M.E. Hellman navrh alternativní a přítažlivé rozšíření výše zkoumaného přístupu. V tomto modelu je prostor zpráv rozdělen do dvou disjunktních podmnožin. První podmnožina obsahuje 2^{HN} smysluplných či typických zpráv, přičemž každá z těchto zpráv má a priori pravděpodobnost 2^{-HN}. Zbývající zprávy nemají v našem jazyku smysl a mají pravděpodobnost 0. Zároveň budeme předpokládat, že klíče jsou použity nezávisle na zprávě a se stejnou pravděpodobností. Je-li C kryptogram, označme $Z(C)$ počet dvojic (M, K) takových, že zpráva M je smysluplná a

$$e(M_i, K_j) = C,$$

pak nepřítel, který zachytí C bude v pochybách o použitém klíči. Je-li $|Z(C)| > 1$, je kryptogram C zašifrován pomocí šifrování s *falešným klíčem*. Klademe-li

$$s(C) = \max\{[Z(C) - 1], 0\},$$

očekávaná hodnota $s(C)$, totiž

$$\overline{s} = \sum_{C \in \mathbf{C}} s(C)P(C),$$

nám odhaduje očekávaný počet šifrování s falešným klíčem. Ale je okamžitě vidět, že

$$\overline{s} = \overline{z} - 1,$$

kde

$$\overline{z} = \sum_{C \in \mathbf{C}} Z(C)P(C) = \sum_{C \in \mathbf{C}} Z^2(C)/2^{NH}|\mathbf{K}|,$$

a tedy z definice modelu obdržíme pro každý kryptogram C

$$P(C) = Z(C)/2^{NH}|\mathbf{K}|.$$

Přitom evidentně

$$\sum_{C \in \mathbf{C}} Z(C) = 2^{NH}|\mathbf{K}|.$$

Aplikujeme-li na výše uvedené jednoduché lemma tvrdící, že pro všechna x_i splňující

$$\sum_{i=1}^{n} x_i = a,$$

máme

$$\sum_{i=1}^{n} x_i^2 \geq a^2/n,$$

obdržíme

$$\overline{Z} \geq (2^{NH}|\mathbf{K}|)^2/|\mathbf{C}|2^{NH}|\mathbf{K}| = 2^{NH}|\mathbf{K}|/|\mathbf{C}|.$$

Můžeme pak vyslovit následující

Věta 3.3.1 *Za předpokladu platnosti výše uvedeného je očekávaný počet šifrování s falešným klíčem odhadnut jako*

$$\overline{s} \geq (2^{NH}|\mathbf{K}|/|\mathbf{C}|) - 1.$$

Píšeme-li nyní $\mathbf{K} = 2^{H(K)}$, $\mathbf{C} = 2^{NH_0} = |\Sigma|^N$, kde H_0 je entropie jazyka, lze výše uvedenou větu přepsat jakožto

$$\overline{s} \geq 2^{NH-H(K)-NH_0} - 1,$$

přičemž pravá strana je rovna nula přesně v bodu unicity.

Příklad: Předpokládejme, že šifrujeme pomocí Vigenerova šifrování tak, že máme anglickou abecedu s 26 písmeny. Víme, že $H_0 = 4.7 = \log 26$ a $H = H_E \simeq 1.5$ bitů, $H(K) = 80 \log 26 = 376$. Obdržíme pak průměrně alespoň $2^{376-320} \simeq 2^{56}$ různých šifrování s falešným klíčem pro kryptogram o 100 písmenech.

Kapitola 4

One-time Pad

Nyní budeme hovořit o následujícím perfektním systému: Předpokládejme, že abeceda Σ je obvyklá 26-ti písmenná anglická abeceda a že tečky, mezery atd. jsou vypuštěny a že odesílaná zpráva M sestává z N písmen. Abychom zašifrovali zprávu, vygenerujeme náhodnou posloupnost o N písmenech z abecedy Σ, přičemž výběr každého písmene je nezávislý a každé písmeno má pravděpodobnost $\frac{1}{26}$, že bude vybráno. Tato náhodná posloupnost $(Z_1, \ldots, Z_N)$ bude klíč K a, abychom zašifrovali $M = (x_1, \ldots, x_N)$ pomocí K budeme definovat

$$C = \mathrm{e}(M, K)$$

$$y_i = x_i \oplus Z_i \bmod 26,$$

kde jako v obvyklém substitučním číslicovém systému jsme písmenům po řadě přiřadili číselnou hodnotu z množiny $\{0, \ldots, 25\}$. Tedy jako klíče vybereme rovněž všech 26^N posloupností délky N; každou z těchto posloupností lze zvolit se stejnou pravděpodobností tj.

$$H(K) = N\log 26.$$

Protože klíč $K = (Z_1, Z_2, \ldots, Z_N)$ je posloupnost náhodných písmen z 26-ti prvkové abecedy Σ, je zřejmé, že existuje 26^N stejně pravděpodobných kryptogramů. Zároveň platí

$$P(K|C) = \frac{1}{26^N}$$

pro všechna $K \in \mathbf{K}$. Z kritéria 3 vidíme, že šifrovací systém $\langle \mathbf{M}, \mathbf{K}, \mathbf{C} \rangle$ je perfektní, neboť klíč je jednoznačně určen zprávou a kryptogramem, množina zpráv je je zároveň množinou klíčů, resp. kryptogramů, zejména tedy mají stejnou mohutnost. Zejména je tedy tento tzv. one-time pad systém *perfektní.*

Tento šifrovací systém byl objeven v roce 1926 americkým inženýrem společnosti AT&T Gilbertem S. **Vernamem** (v dřívějších dobách byla písmena klíče napsána na listech trhacího bloku a jakmile bylo klíčové písmeno použito, byl odpovídající list vytrhnut a zničen).

Dnes se one-time pad neprovozuje s písmeny nýbrž s bity. Pak $\mathbf{a}_i$, $\mathbf{k}_i \in \{0,1\}$ a kryptogram $\mathbf{a}_1 \oplus \mathbf{k}_1 \mathbf{a}_2 \oplus \mathbf{k}_2 \dots \mathbf{a}_n \oplus \mathbf{k}_n$ získáme pomocí binárního sčítání.

Pro bezpečnost tohoto systému je podstatné, že všechny posloupnosti délky n se vyskytují s toutéž pravděpodobností. Jinak řečeno: Bity klíčového slova musí být voleny náhodně. Nejlépe si to představíme tím způsobem, že vrháme ideální minci. V praxi používáme fyzikální náhodný zdroj a tím automaticky vytvoříme bity.

Za tuto formu perfektní bezpečnosti musíme – nikoliv neočekávaně – platit vysokou cenu. Pro tradiční one-time pad potřebujeme velké množství papíru, který musí být před útočníkem absolutně bezpečně ukryt. Proto se takovéto systémy používají jen zřídka. V druhé světové válce se one-time pad používal anglickou dešifrovací skupinou, aby zprostředkovala zprávy premiérovi, které byly Němci zašifrovány pomocí Enigmy a které Angličané rozšifrovali. Tímto způsobem si spojenci zajistili, že Němci až do konce války nevěděli, že Enigma byla rozluštěna.

Jednou z dalších nevýhod tohoto systému je neexistence matematického způsobu generování nezávislých náhodných proměnných, které slouží jako klíč. Tedy je nutné použít pseudonáhodných posloupností generovaných jednou z mnoha standardních metod. Neexistuje pak žádná záruka, že takováto pseudonáhodné posloupnosti nám budou stejnou úroveň bezpečnosti. Jedná se o hluboký matematický problém.

*

Proč se tento nepochybně perfektní systém používá jen velmi zřídka? Abychom byli schopni zodpovědět tuto otázku, představme si sebe v roli příjemce. Ten může přirozeně kryptogram pohodlně rozšifrovat: dešifrování je v podstatě stejný postup jako zašifrování (používáme-li bity, jedná se dokonce o přesně totéž). To al může příjemce provést jen v případě, že má klíč.

Kde je vlastně problém? Problém spočívá v tom, že musíme přenést (doručit) dlouhý tajný klíč. Kdybychom toto prováděli pomocí stejné cesty jako zprávu, je vzhledem k délce klíče šance přečtení klíče stejná jako při předání nezašifrovaného textu zprávy. Člověk by si mohl myslet, že u takovéhoto systému by mohl odesilatel zprávu nepříteli předat přímo do domu. To ale není zcela správné; totiž pro přenos klíče může odesilatel určit druh, způsob a dobu předání, což samozřejmě u přenosu zprávy neplatí. Jiný způsob přenosu klíče je použití kurýra.

4.1 Posouvací registry

Při přenosu klíče se nejedná pouze o teoretický problém, nýbrž o to, že obtížnost výměny klíče silně ovlivňuje nasazení šifrovacích systémů.

Důležitý postup pro vyřešení tohoto problému spočívá v tom, že namísto skutečně náhodných klíčových posloupností použijeme pouze **pseudonáhodné** posloupnosti. Takováto posloupnost vypadá na první pohled jako skutečná náhodná posloupnost. A co je ještě důležitější: náhodnou posloupnost lze určit pomocí několika málo dat; tyto data pak představují skutečný klíč. Oba komunikující partneři pak mohou z těchto dat spočítat náhodné posloupnosti a zašifrovat zprávu resp. dešifrovat kryptogram. Problém přenosu klíče tak není zcela vyřešen, ale podstatně ulehčen.

Samozřejmě musíme za tuto výhodu zaplatit: takovéto systémy neposkytují žádnou perfektní bezpečnost. Budeme tedy hledat kompromis mezi docílenou bezpečností a množinou tajně přenositelných dat.

*

Posouvací registr je poslopnost v řadě za sebou následujících registrů, přičemž každý registr může obsahovat pouze číslici 1 (on) nebo 0 (off). Hodinový strojek reguluje chování systému, který pracuje v souladu s následujícími podmínkami:

Předpokládejme, že systém má n registrů $R_1, R_2, \ldots, R_n$ a že $X_i(t)$ označuje obsah registru R_i v čase t. Nechť je dále na začátku systém ve stavu

$$\mathbf{x}(0) = (\mathrm{X}_1(0), \ldots, \mathrm{X}_\mathrm{n}(0)).$$

Pokud

$$\mathbf{x}(\mathrm{t}) = (\mathrm{X}_1(\mathrm{t}), \ldots, \mathrm{X}_\mathrm{n}(\mathrm{t})).$$

označuje stav systému v době t, stav v čase $t+1$ je určen vztahy

$$X_i(t+1) = X_{i-1}(t) \qquad (2 \leq i \leq n), \tag{4.1}$$

$$X_1(t+1) = f(\mathbf{x}(\mathrm{t})), \tag{4.2}$$

kde f je nějaká binární funkce n proměnných. Pokud je f tvaru

$$f = \sum_{i=1}^{n} c_i X_i(t),$$

mluvíme o lineárním posouvacím registru. Přitom chování systému je jednoznačně určeno (a) počátečním stavem $\mathbf{x}(0)$ a (b) množinou konstant $c_1, \ldots, c_n$. Budeme vždy předpokládat, že $c_n \neq 0$; jinak bychom mohli pracovat bez registru R_n.

Způsob, jakým lineární systém pracuje, je, že po obdržení signálu každý registr provede dvě věci:

(i) Přenese svůj obsah do svého pravého souseda ($n-$tý registr toto provést nemůže).

(ii) Takové registry R_i, pro které je $c_i = 1$ přenesou svůj obsah do čítače, ten je sečte a výsledek přenese do registru R_1.

Jakmile je jednou nastaven počáteční vektor, lze posouvací registr považovat za zdroj nekonečné posloupnosti binárních číslic

$$X_1(0), X_1(1), X_1(2), \ldots .$$

Ačkoliv takto vytvořená posloupnost není náhodná, lze ukázat, že má jisté rysy nahodilosti. Navíc ji lze snadno a rychle generovat. Bohužel je však velmi nejistá.

Vlastnosti posloupností vytvořených lineárními posouvacími registry

Nejprve uvažujme periodicitu. Nekonečná posloupnost $(y_i : 0 \leq i < \infty)$ se nazývá *periodická* s *periodou* p, jestliže je p kladné přirozené číslo takové, že $y_{i+p} = y_i$ pro všechna i a navíc je p nejmenší kladné přirozené číslo s touto vlastností. Má-li tedy posloupnost $(y_i : 0 \leq i < \infty)$ periodu p, můžeme ji psát ve tvaru

$$y_0, y_1, y_2, \ldots, y_{p-1}, y_0, y_1, y_2, \ldots, y_{p-1}, y_0, y_1, \ldots .$$

Jinak řečeno, posloupnost s periodou p je přesně posloupnost opakování konečného bloku délky p.

Vraťme se nyní k posloupnosti určené lineárním posouvacím registrem: předpokládejme, že počáteční vektor $\mathbf{x}(0)$ není nulový vektor a že rovnice 4.1 a 4.2 lze přepsat ve tvaru

$$\mathbf{x}(\mathrm{t}+1) = \mathbf{C}\mathbf{x}(\mathrm{t}), \tag{4.3}$$

kde $\mathbf{C}$ je matice tvaru

$$\mathbf{C} = \begin{bmatrix} c_1 & c_2 & c_3 & \ldots & c_{n-1} & c_n \\ 1 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 1 & 0 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 1 & 0 \end{bmatrix}.$$

Protože jsme předpokládali, že $c_n = 1$ a protože

$$\det \mathbf{C} = c_n = 1,$$

vidíme, že $\mathbf{C}$ je regulární matice. Iterováním rovnice 4.3 dostaneme $\mathbf{x}(\mathrm{t}) = \mathbf{C}^{\mathrm{t}}\mathbf{x}(0)$. Přitom platí

Věta 4.1.1 *Posloupnost vytvořená pomocí lineárního posouvacího registru je periodická a pokud je vytvořena z n registrů, je její maximální perioda $2^n - 1$.*

Důkaz. Protože je $\mathbf{C}$ regulární, je i regulární matice $\mathbf{C}^{\mathrm{i}}$ $(i = 0, 1, \ldots)$; přitom je $\mathbf{x}(0)$ nenulový vektor a existuje právě $2^n - 1$ nenulových vektorů délky n. Je-li $m = 2^n - 1$, pak jsou

$$\mathbf{x}(0), \mathbf{C}\mathbf{x}(0), \mathbf{C}^2\mathbf{x}(0), \ldots, \mathbf{C}^{\mathrm{m}}\mathbf{x}(0)$$

nenulové vektory délky n a tudíž nemohou být všechny různé: řekněme, že

$$\mathbf{C}^{\mathrm{s}}\mathbf{x}(0) = \mathbf{C}^{\mathrm{s+t}}\mathbf{x}(0),$$

kde $0 \leq s < s + t \leq 2^n - 1$. Protože existuje $\mathbf{C}^{-\mathrm{s}}$, máme

$$\mathbf{x}(\mathrm{t}) = \mathbf{C}^{\mathrm{t}}\mathbf{x}(0) = \mathbf{C}^{-\mathrm{s}}\mathbf{C}^{\mathrm{s+t}}\mathbf{x}(0) = \mathbf{x}(0).$$

Tedy

$$\mathbf{x}(\mathrm{r} + \mathrm{t}) = \mathbf{C}^{\mathrm{r+t}}\mathbf{x}(0) = \mathbf{C}^{\mathrm{r}}\mathbf{C}^{\mathrm{t}}\mathbf{x}(0) = \mathbf{C}^{\mathrm{r}}\mathbf{x}(\mathrm{t}) = \mathbf{C}^{\mathrm{r}}\mathbf{x}(0) = \mathbf{x}(\mathrm{r})$$

pro všechna $r \geq 0$, a $\mathbf{C}$ je periodická s periodou nejvýše $t \leq 2^n - 1$. Posloupnost vytvořená pomocí lineárního posouvacího registru je tedy periodická.

Definujme *charakteristický polynom* lineárního posouvacího registru jako polynom

$$P_n(x) = 1 + \sum_{i=1}^{n} c_i x^i,$$

s $c_n \neq 0, c_i \in \{0, 1\}$. Charakteristický polynom je *primitivní*, jestliže

(a) nemá vlastní netriviální dělitele,

(b) $P_n(x)$ nedělí polynom $x^d + 1$ pro všechna $d < 2^n - 1$.

Následující tvrzení uvedeme bez důkazu.

Věta 4.1.2 *Posloupnost vytvořená pomocí lineárního posouvacího registru z nenulového vstupu má maximální periodu právě tehdy, je-li její charakteristický polynom primitivní.*

Nalezení primitivních polynomů je netriviální úloha moderní algebry. Poznamenejme pouze, že primitivní polynomy existují pro každé n a že jsou tabelovány.

4.2 Kryptoanalýza lineárních posouvacích registrů

Můžeme tedy použít lineární posouvací registry k vytvoření pseudonáhodných posloupností pro kryptografické účely. To je laciné, lineární posouvací registry provádí výpočty velmi rychle – co můžeme ještě víc chtít?

Nelze popřít, že posloupnosti vytvořené pomocí lineárních posouvacích registrů mají vynikající statistické vlastnosti; a to platí dokonce pro posloupnosti, které vzniknou z relativně krátkých lineárních posouvacích registrů. Ale z kryptologického pohledu mají tyto posloupnosti mimořádně pochybný charakter. To je důsledkem toho, že v případě known-plaintext útoku mu nejsou schopny odolat.

Definujme *blok délky* t jako posloupnost tvaru $011\ldots10$ obsahující právě t jedniček. *Dírou délky* t je posloupnost tvaru $100\ldots01$ obsahující právě t nul.

Platí následující výsledek.

Věta 4.2.1 *Má-li lineární posouvací registr s* n *registry maximální periodu* $2^n - 1$*, mají pak výsledné posloupnosti délky* $2^n - 1$ *následující vlastnosti:*

(a) *obsahuje právě* $2^{n-1} - 1$ *nul a* 2^{n-1} *jedniček;*

(b) *obsahuje pro všechna* t *taková, že* $1 \leq t \leq n-2$*,* 2^{n-t-2} *bloků délky* t *a stejný počet děr délky* t*.*

Důkaz. Stav lineárního posouvacího registru lze v každém okamžiku jednoznačně popsat přirozeným číslem z intervalu $[1..2^n-1]$: stačí vzít příslušnou část výstupní posloupnosti.

Protože se všechna nenulová čísla z intervalu $[1..2^n-1]$ musí vyskytnout jako stavy v cyklu maximální délky, výsledek okamžitě dostaneme výsledek (a) spočtením sudých a lichých čísel v této množině.

Abychom dokázali (b), poznamenejme, že běh typu $011\ldots10$ obsahující právě t jedniček se může vyskytnout jako součást výstupu právě tehdy, když v nějaké části výpočtu je stav lineárního posouvacího registru $011\ldots10x_1x_2\ldots x_{n-t-2}$, kde $x_i \in \{0,1\}$. Protože máme právě 2^{n-t-2} stavů tohoto tvaru a protože každý stav je realizován v nějakém okamžiku výpočtu vzhledem k tomu, že lineární posouvací registr má maximální periodu, výsledek (b) pro bloky platí. Zaměníme-li 0 a 1, dostáváme výsledek (b) pro díry.

Vraťme se nyní k dešifrování. Je-li

$$M = m_1m_2\ldots$$

zpráva složená z binárních číslic, a je-li

$$X = x_1x_2\ldots$$

posloupnost vyprodukovaná lineárním posouvacím registrem, pak kryptogram C je posloupnost

$$C = c_1 c_2 \dots,$$

kde

$$c_i = m_i + x_i \pmod 2 \qquad (1 \leq i < \infty). \tag{4.4}$$

Jsou-li tedy m_i a c_i známy, lze x_i získat triviálně jako

$$x_i = m_i + c_i \pmod 2 \qquad (1 \leq i < \infty). \tag{4.5}$$

Uvažme nyní lineární posouvací registr s n registry a koeficienty $d_1, d_2, \dots, d_n$. Jakmile zná nepřítel *nějakých 2 n* za sebou následujících členů x_i výsledné posloupnosti, je schopen najít tyto koeficienty $d_1, d_2, \dots, d_n$. Totiž odpovídající systém lineárních rovnic má tvar

$$\begin{bmatrix} 1 & 0 & 0 & \dots & & & 1 & 0 & 0 & \dots & & \\ 0 & 1 & 0 & \dots & & & 0 & 1 & 0 & \dots & & \\ 0 & 0 & 1 & \dots & & & 0 & 0 & 1 & \dots & & \\ \vdots & \vdots & \vdots & \ddots & & & \vdots & \vdots & \vdots & \ddots & & \\ 0 & 0 & 0 & 0 & 0 & 0 & x_1 & x_2 & x_3 & \dots & x_{n-1} & x_n \\ 0 & 0 & 0 & 0 & 0 & 0 & x_2 & x_3 & x_4 & \dots & x_n & x_{n+1} \\ 0 & 0 & 0 & 0 & 0 & 0 & x_3 & x_4 & x_5 & \dots & x_{n+1} & x_{n+2} \\ \vdots & \vdots & \vdots & \ddots & & & \vdots & \vdots & \vdots & \ddots & & \\ 0 & 0 & 0 & 0 & 0 & 0 & x_n & x_{n+1} & x_{n+3} & \dots & x_{2n-2} & x_{2n-1} \end{bmatrix} \cdot \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ \vdots \\ c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n+1} \\ x_{n+2} \\ x_{n+3} \\ \vdots \\ x_{2n} \end{bmatrix}.$$

To pak plně určuje šifrovací systém a tudíž platí následující věta.

Věta 4.2.2 *Užití posloupností vytvořených pomocí lineárního posouvacího registru není bezpečné proti known-plaintext útoku.*

Chceme-li zachovat hezké vlastnosti lineárních registrů a zároveň zajistit větší stupeň bezpečnosti, použijeme v rovnici 4.2 nelineární funkci. Skutečně je tomu tak, že většina dnes používaných algoritmů je založena na nelineárních posouvacích registrech, ačkoliv bychom neměli zapomenout na DES.

Kapitola 5

Výpočetní složitost

5.1 Obtížnost problému

Důkaz toho, že existují nerozhodnutelné problémy a nevyčíslitelné funkce, je jedním z největších výdobytků v této oblasti.

Šifrovací systém, jehož dešifrování je založeno na výpočtu nevyčíslitelných funkcí, by měl výhodnou pozici. Můžeme však snadno ověřit, že takovéto zbožné přání nemůže být splněno: všechny takovéto systémy jsou konečné a tedy mohou být narušeny prověřením všech možností.

Teorie výpočetní složitosti se týká třídy problémů, které lze v principu vyřešit: ale vzhledem k této třídě se teorie pokouší klasifikovat problémy podle jejich výpočetní obtížnosti v závislosti na množství času nebo paměti potřebných pro toto řešení,

Porozumění základním pojmům teorie složitosti je podstatné pro kryptografii a v této kapitole se budeme snažit pokrýt podstatu problémů teorie složitosti. Nejprve začněme informálně s několika příklady.

Příklad. 1 *Násobení přirozených čísel* Uvažme problém násobení dvou binárních n-bitových čísel x a y. Je-li $x = x_1 \dots x_n$ a $y = y_1 \dots y_n$, můžeme provést standardní metodu „dlouhého násobení“, která je učena na základní škole následovným způsobem:

Postupně násobme x čísly y_1, y_2 atd., posuňme a pak přičtěme výsledek. Každé násobení x číslem y_i nás stojí n jednoduchých *bitových operací*. Podobně sečtení n součinů nám zabere $O(n^2)$ bitových operací. Je tedy celkový počet operací $O(n^2)$. Můžeme výše uvedené ještě zlepšit? Tj., existuje rychlejší algoritmus v tom smyslu, že provede podstatně méně bitových informací. Přesněji, jsou-li dána 2 n-bitová čísla, n sudé, píšeme

$$x = a2^{\frac{1}{2}} + b, \qquad y = c2^{\frac{1}{2}} + d,$$

pak součin z lze získat pomocí tří multiplikací $\frac{1}{2}$ n-bitových čísel použitím repre-

zentace
$$z = xy = (ac)2^n + [ac + bd - (a-b)(c-d)]2^{\frac{1}{2}n} + bd.$$

Označíme-li $T(n)$ čas, který nám zabere násobení podle této metody, máme, prože násobení 2^n je rychlé, že

$$T(n) \leq 3T(\frac{1}{2}n) + O(n).$$

Po vyřešení výše uvedené rekurentní nerovnosti obdržíme, že

$$T(n) \leq An^{\log 3} + Bn,$$

kde A a B jsou konstanty. Protože $\log 3 \simeq 1.59$, máme k dispozici algoritmus o časové složitosti $O(n^{1.59})$ v porovnání se standardním $O(n^2)$ algoritmem. V současnosti má jeden z nejlepších známých algoritmů (Schönhagen, Strassen) složitost $O(n \log n \log\log n)$.

Příklad. 2 *Determinanty a permanenty.* Uvažujme následující dva výpočetní problémy. Pro každý vstup složený z binární matice A typu $n \times n$ chceme vypočítat

1. determinant matice A, píšeme pak $\det A$,

2. permanent matice A, píšeme pak $\mathrm{per} A$, permanenent je definován jako

$$\mathrm{per} A = \sum_{\pi} a_{1\pi(1)} a_{2\pi(2)} \dots a_{n\pi(n)},$$

kde sčítáme přes všechny permutace π na množině $\{1, 2, \dots, n\}$ a a_{ij} značí $(i,j)-$tou komponentu matice A.

Zdánlivě je permanent mnohem jednodušší funkce matice A než její determinant; jedná se o součet toho samého systému termů, ale bez toho, že bychom dávali pozor na $\pm$ znaménka jako u determinantu.

Avšak z hlediska výpočetní složitosti platí opak: zatímco determinant je relativně snadná funkce k výpočtu, u permanentu se prokázalo, že se jedná o vždy téměř jistě o neobvykle obtížnou záležitost.

Upřesněme výše uvedené: standardní Gaussova eliminační metoda výpočtu determinantu matice $n \times n$ potřebuje $O(n^3)$ bitových operací, přičemž V. Strassen zkonstruoval algoritmus, který potřebuje

$$O(n \log_2 7) = O(n^{2.81\cdots})$$

bitových operací. Redukce exponentu pod hranici $\log_2 7$ se prokázalo být velmi obtížné a jeden z nejrychlejších současných algoritmů (Coppersmith, Winograd) potřebuje $O(n^{2.3976\cdots})$ operací. Snadn je vidět, že všechny vstupy matice musí být načteny a tedy
$$n^2 \leq t_{\det}(n) \leq Cn^{2.3976\dots}$$

kde $t_{\mathrm{det}}(n)$ označuje časovou složitost problému výpočtu determinantu a C je nějaká konstanta. Pro permanent však oproti výše uvedenému žádný takový algoritmus není znám. Nejrychlejší doposud známý algoritmus je pouze o něco lepší než sečtení všech $n!$ termů naší sumy.

Příklad. 3 *Třídění.* Předpokládejme, že chceme sestrojit algoritmus, který, obdrží-li na vstupu n celých čísel $a_1, \ldots, a_n$, setřídí tyto v rostoucím pořadí. Snadný, téměř instinktivní přístup je následující algoritmus, známý jako *Bubblesort.*

Postupně porovnávejme a_1 s každým s prvků $a_2, \ldots, a_n$. Pro maximální index i takový, že $a_i < a_1$ umístěme a_1 za a_i a obdržíme pak nové uspořádání. Po $n-1$ porovnáních obdržíme uspořádání $b_1, \ldots, b_n$; je okamžitě vidět, že se jedná o vzestupně uspořádaný seznam. Jednoduchý výpočet ukazuje, že k výše uvedenému je třeba $O(n^2)$ porovnání.

Poznamenejme, že máme několik rekurzivních algoritmů založených na principu rozděl a panuj tím, že třídíme 2 polovice množiny a pak spojíme setříděné seznamy k sobě, což nám zabere pouze $O(n \log n)$ srovnání. Pro názornost uveďme následující tabulku

n	$n \log_2 n$	n^2
50	$\simeq 300$	2500
500	$\simeq 4500$	250000

.

Ovšem, výraz $O(n \log n)$ může skrýt velké konstantní výrazy, ale tak joko tak se jedná o velký rozdíl, obzvláště proto, že třídění je často používaný algoritmus a velikost seznamů je často velmi velká.

Jiný fascinující pohled na třídění je ten, že existuje spodní mez stejného řádu pro každý algoritmus založený na třídění. Často mluvíme o informačně-teoretické spodní mezi, ale nejedná se o nic jiného, než o přímý důsledek pozorování, že každý algoritmus založený na srovnání lze reprezentovat pomocí binání stromové struktury a protože musíme pokrýt všech $n!$ možných uspořádání, každý takovýto strom musí mít alespoň $n!$ listů.

Příklad. 4 *Test prvočíselnosti.* Bezprostředně se zdá, že testovat, zda přirozené číslo N je prvočíslo, lze vyřešit velmi rychlým a snadným algoritmem: testujeme, zda je N dělitelené 2 nebo nějakým lichým číslem z intervalu $[3, N^{\frac{1}{2}}]$. Protože se jedná pouze o $\frac{1}{2}N^{\frac{1}{2}}$ dělení, jedná se o polynomiální algoritmus v proměnné N a tudíž rychlý algoritmus.

Avšak další úvahy ukazují, že reprezentace čísla N v počítači by byl binární řetězec délky $\lceil \log N \rceil$ a tudíž, abychom mohli algoritmus na testování prvočíselnosti považovat za rychlý, jeho výpočetní složitost musí být polynomiální v $n = \lceil \log N \rceil$. Doposud však žádný takovýto algoritmus není znám. V současnosti jsou k dispozici algoritmy se složitostí

$$t(N) = O(\ln N)^{c \ln \ln \ \ln N},$$

kde c je kladná konstanta.

Příklad. 5 *Největší společný dělitel a Euklidův algoritmus.* Uvažujme problém nalezení největšího společného dělitele (nsd) dvou přirozených čísel u a v. Zřejmá metoda je faktorizace obou čísel na prvočísla

$$u = 2^{u_1}3^{u_2}5^{u_3}\dots, \qquad v = 2^{v_1}3^{v_2}5^{v_3}\dots,$$

pak lze zjistit jejich největší společný dělitel následovně

$$w = \mathrm{nsd}(u, v) = 2^{w_1}3^{w_2}5^{w_3}\dots,$$

kde $w_i = \min\{u_i, v_i\}$. Jedná se však o velmi neefektivní postup. Potřebujeme totiž faktorizovat obě čísla a tento postup nelze rychle provést pro velká přirozená čísla. Metoda, jíž tento postup můžeme obejít, je známá jako *Euklidův algoritmus.*

Předpokládejme, že $u > v > 0$. Pak obdržíme posloupnost dělení:

$$\begin{aligned}
u&=a_1v + b_1, & 0 &\le b_1 < v,\\
v&=a_2b_1 + b_2, & 0 &\le b_2 < b_1,\\
b_1&=a_3b_2 + b_3, & 0 &\le b_3 < b_2,\\
&\vdots\\
b_{k-2}&=a_kb_{k-1} + b_k, & 0 &\le b_3 < b_2,
\end{aligned}$$

která skončí buď $b_k = 0$ nebo $b_k = 1$. Pokud $b_k = 1$, jsou čísla u a v nesoudělná; pokud $b_k = 0$, je $\mathrm{nsd}(u, v) = b_{k-1}$.

O tomto algoritmu lze snadno dokázat, že je korektní a detailní analýza jeho účinnosti ukáže, že nejhorší případ (měřeno počtem dělení) nastane, jsou-li u a v za sebou následující Fibonacciho čísla F_{n+2} a F_{n+1}. Pak v tomto případě

$$F_{k+2} = F_{k+1} + F_k,$$

a to vede k následujícímu Lamého výsledku (1845)

Věta 5.1.1 *Je-li $0 \le u, v < N$, pak počet dělení při použití Euklidova algoritmu na u a v je nejvýše*

$$\lceil log_\varphi(\sqrt{5}N)\rceil - 2,$$

kde φ je zlatý řez $\frac{1}{2}(1 + \sqrt{5})$.

5.2 P=polynomiální čas

Základní mírou obtížnosti výpočtu je množství doby, které nám výpočet zabere. Zformulování přesné definice, co to je čas, je netriviální záležitost; přesná formulace požaduje velmi precizní definici strojového modelu, jednotky času atd.

V příkladech z předchozího paragrafu jsme měřili *složitost* výpočtu v pojmech počtu základních operací, které byly prováděny. Mohlo se jednat o součet bitů, srvnání nebo cokoliv jiného.

Klíčové pojmy jsou následující:

1. složitost je funkce velikosti vstupu (obvykle ji značíme jako n),
2. pro danou velikost vstupu n je složitost doba *nejhoršího možného případu* běhu algoritmu.

Připomeňme si, že při testování prvočíselnosti přirozeného čísla N jsme obdrželi jinou složitost v případě, že jsme považovali vstup velikosti N nebo vhodněji pomocí reprezentace $n = \lceil \log N \rceil$ binárních číslic. Na základě tohoto důsledku bude *velikost vstupu* vždy považována za „přirozenou" délku ekonomického vstupu. Co se týče definice složitosti jako nejhoršího možného případu, jiná možnost – „průměrný případ" – se potýká s obtížemi, a to jak teoretickými tak praktickými. Ne poslední obtížnost je rozhodnout citlivě o pravdivostním rozdělení na vstupu.

Nejprve budeme postupovat *neformálně*. Řekneme, že algoritmus $\mathcal{A}$ má *polynomiální složitost*, jestliže existuje polynom $p(x)$ tak, že

$$t_{\mathcal{A}}(n) \leq p(n),$$

pro všechna přirozená čísla n, přičemž $t_{\mathcal{A}}(n)$ je maximální doba potřebná algoritmem k výpočtu přes všechny vstupy velikosti n.

Problém lze provést v *polynomiálním čase*, pokud existuje nějaký algoritmus, který ho řeší a má polynomiální složitost, v tomto případě tvrdíme, že *problém leží v třídě P*.

Porovnejme naši definici s příklady 1-5 z předchozího paragrafu.

Příklad. 1 *Násobení přirozených čísel* je operace prováděná v polynomiální době,

Příklad. 2 *Determinanty a permanenty.* Výpočet determinantu je problém, který určitě leží v P. U výpočtu permanentu nevíme, zda tento problém leží v P, předpokládá se, že zde neleží, a důkaz jakékoliv implikace by měl velkou důležitost v teorii složitosti.

Příklad. 3 *Třídění* lze provést pomocí $O(n \log n)$ srovnání a protože srovnání lze provést v polynomiálním čase, leží třídění v P.

Příklad. 4 O *testu prvočíselnosti* nevíme, zda leží v P. Zatím bylo dokázáno, že je „velmi blízko" P.

Příklad. 5 Nalezení *největšího společného dělitele* dvou přirozených čísel velikosti $\leq N$ a proto velikosti vstupu $\log N$ bitů lze provést v čase $O(\log N)$ a proto tento problém leží v P.

Třída P je v současnosti nejdůležitější třídou v matematice a computer science, to, že nějaký problém leží v P, lze obvykle považovat za to, že se jedná o výpočetně dobrý problém. Ačkoliv poslední uvedené obecně zcela neplatí (problém nalezení klik v grafu), následující tvrzení nám ukazují, že se jedná o atraktivní a efektivní pojem.

1. Třída P je robustní vzhledem k různým reprezentacím vstupních hodnot za předpokladu, že tyto změny jsou vůči sobě polynomiálně korelovány.[‡] Například, zda uvažujeme vstup matice typu $n \times n$ velikosti n nebo n^2, nedělá žádný rozdíl.

2. Třída P je robustní vzhledem k použitému modelu výpočetního stroje. Jinak řečeno, zda použijeme Pentium nebo stroj s náhodným přístupem nebo Turingův stroj, naše třída zůstane nezměněna. Toto lze celkem snadno dokázat. Je pouze nutno ověřit, že doby pro simulaci základních operací na obou strojích, jsou polynomiálně korelovány.

Připomeňme stručně formální definici třídy P.

Turingovy stroje – formální definice třídy P

Turingův stroj sestává z 2-směrné nekonečné pásky rozdělené do čtverců. Každý čtverec může obsahovat symbol z konečné abecedy Σ. Páska je snímána rychlostí 1 čtverec za jednotku času tzv. *čtecí i zapisovací* hlavicí. Stroj může být v jednom z konečné množiny Q stavů, $Q = \{q_0, q_1, \ldots, q_m\}$ a příslušná akce stroje v danám čase je jednoznačně určena jeho vnitřním stavem a symbolem v současně snímaném čtverci. Akce provede libovolnou z následujících operací:

1. změní (přepíše) snímaný symbol na jiný symbol ze Σ,

2. posune čtecí hlavici o jeden čtverec doprava či doleva,

3. změní svůj současný stav z q_i na q_j.

Výpočet Turingova stroje sestává z následujících kroků:

1. reprezentace výpočtu konečným řetězcem $x \in \Sigma^*$, který je umístěn ve čtvercích $1 - n$, kde n je počet symbolů obsažených v x,

2. odstartování činnosti Turingova stroje z jeho počátečního stavu (obvykle q_0) s přepisovací hlavou na čtverci 1 a jeho pokračování se základními operacemi (četní, zapsání a změny stavu) až do doby, kdy stroj skončí v koncovém stavu (q_f).

Výstup stroje M po jeho aplikování na stav x je obsah pásky dosažený v koncovém stavu. Jeden *krok* výpočtu stroje sestává z jedné akce 1-3 uvedených výše a *délka* neboli *čas použitý* při výpočtu je počet takovýchto kroků. Pokud M označuje nějaký Turingův stroj , označíme tuto dobu $t_M(x)$.

Funkce $f : \Sigma^* \to \Sigma^*$ je vyčíslitelná pomocí Turingova stroje M, jestliže pro všechna $x \in \Sigma^*$, x je vstup pro M, v případě ukončení výpočtu stroj zastaví s

[‡]Dvě funkce f a g jsou vůči sobě polynomiálně korelovány, pokud existují polynomy p_1 a p_2 tak, že $f(n) \leq p_1(g(n))$ a $g(n) \leq p_2(f(n))$ pro všechna dostatečně velká n.

hodnotou $f(x)$ na jeho výstupní pásce. Tedy Turingův stroj je přesná analogie počítače – každý výpočet, který lze vykonat moderním počítačem, lze provést i Turingovým strojem. Ovšem v praxi je konstrukce Turingova stroje schopného i pouze jednoduchých výpočtů velmi časově náročná. Proto byl vyvinut soubor základních Turingových strojů, které provádí odpovídající úlohy. Konstruujeme-li pak složitý Turingův stroj, používáne strojů již dříve zkonstruovaných, podobně jako když používáme podrutiny v obvyklých počítačových programech.

Můžeme pak formálně definovat časovou složitost. Je-li M Turingův stroj, který zastaví pro všechny vstupy $x \in \Sigma^*$, *časová složitost* Turingova stroje M je funkce $t_M : \mathbf{Z}^+ \to \mathbf{Z}^+$ určená vztahem

$$\begin{aligned} t_M(n) \quad &= \max\{t : \text{existuje } x \in \Sigma^* \text{ tak, že } |x| = n \\ &\qquad \text{a čas provedený strojem } M \text{ na vstupu } x \text{ je } t\}. \end{aligned}$$

Funkce f je vyčíslitelná v *polynomiálním čase* nebo má *polynomiální složitost*, pokud existuje nějaký Turingův stroj, který vypočte f a jistý polynom p tak, že $t_M(n) \leq p(n)$ pro všechna n. V praxi většinou neuvažujeme s Turingovým modelem, ale pracujeme na mnohem vyšší úrovni.

5.3 NP = nedeterministický polynomiální čas

Popišme si neformální ideu třídy NP. Předpokládejme, že máte za úkol prodat velká složená čísla opravdu hodně zaměstnaným nákupčím. S pomocí otroků pracujících neomezený počet hodin můžete sestavit seznam složených čísel c_1, c_2, ... a abychom byli schopni prodávat tato čísla rychle, budete mít k dispozici odpovídající seznam faktorů y_1, y_2, ... tak, že pokud má dojít k prodeji, vše, co musíte udělat, je dát číslo c_i dohromady s faktorem y_i a ověřit, že složené číslo c_i je dělitelné číslem y_i. Pak y_i nazýváme *certifikátem* neprvočíselnosti čísla c_i, protože pak existuje při jeho použití algoritmus pracující v polynomiální době pro ověření, že c_i je složené.

Neformálně můžeme tedy říci, že vlastnost *náleží* do NP, jestliže splnění této vlastnosti lze ověřit v polynomiální době s pomocí vhodného certifikátu.

Podejme nyní formální definici:

Buď Σ nějaká konečná abeceda. Libovolnou podmnožinu π množiny Σ^* nazveme *vlastností* (jazykem). Řekneme pak, že $\pi \in NP$, jestliže existuje funkce $f : \Sigma^* \times \Sigma^* \to \{0, 1\}$ tak, že

1. $x \in \pi$ právě tehdy, když existuje $y \in \Sigma^*$ tak, že $f(x, y) = 1$,

2. výpočtová časová náročnost f je omezena polynomem v proměnné x.

V pojmech teorie Turingových strojů, považujem y sdružené se vstupem x za *certifikát* relace patřit prvku x v π, a necháváme Turingův stroj pracovat v

polynomiální době na vstupu sestávajícímu z vstupu x a certikátu y. Speciálně pak
$$P \subseteq NP,$$
považujeme-li P za soubor vlastností. Otázka, zda $P = NP$ je pracděpodobně nejdůležitější otázkou v teoretické computer science.

5.4 NP-úplné a NP-těžké problémy

Důležitou vlastností třídy NP je, že obsahuje „nejtěžší vlastnosti“ tak, že kdybychom byli schopni vyřešit některou z těchto vlastností, byli bychom schopni rozhodnout každou z vlastností v NP. Precizněji, řekneme, že vlastnost π_1 je *polynomiálně redukovatelná* na vlastnost π_2, pokud existuje funkce f z P tak, že x má vlastnost π_1 právě tehdy, když $f(x)$ má vlastnost π_2. Píšeme pak
$${\pi_1}^{\alpha}\pi_2.$$
Je snadno vidět, že pokud ${\pi_1}^{\alpha}\pi_2$, implikuje existence polynomiálního algoritmu pro π_2 existenci polynomiálního algoritmu pro π_1. Totiž, nechť $\mathcal{A}$ je algoritmus pro π_2, který pracuje v čase $t(n)$. Je-li x nějaký vstup pro π_1 tak, že $|x| = n$, pak transformujme x na $f(x)$ a aplikujme algoritmus $\mathcal{A}$. Protože transformace pracuje v polynomiálním čase, je $|f(x)|$ ohraničené nějakým polynomem $g(n)$. Tedy transformace f a algoritmus $\mathcal{A}$ pracují v čase ohraničeny nějakým polynomem.

Připomeňme následující tvrzení:

Věta 5.4.1 *Existuje vlastnost $\pi \in NP$ tak, že libovolná jiná vlastnost π' je polynomiálně redukovatelná na π.*

Takovéto π se nazývá *polynomiálně úplný problém.*

Připomeňme, že máme k dispozici seznam více než 3000 NP-úplných problémů. Přitom téměř každá vlastnost z NP, o které se neví, zda leží v P, je NP-úplná, ačkoliv máme k dispozici tvrzení, které říká, že pokud $NP \neq P$, pak $NP - P$ obsahuje problémy, jež nejsou NP-úplné.

Řekneme, že problém π je NP-těžký, pokud existuje nějaká NP-úplná vlastnost π_0 tak, že pokud existuje polynomiální algoritmus pro π, existuje i polynomiální algorotmus pro π_0. Triviální důsledky této definice jsou následující tvrzení:

1. Pokud existuje polynomiální algoritmus pro každý NP-problém, je pak $NP = P$.

2. Je-li π_1 NP-těžký a ${\pi_1}^{\alpha}\pi_2$, je i π_2 NP-těžký.

3. Každý NP-úplný problém je NP-těžký.

Obrácené tvrzení k 3 není pravdivé.

5.5 Složitost kombinačních obvodů

Věnujme se na chvíli zcela odlišnému modelu výpočtu, který zhruba odpovídá realizaci čipu nebo VLSI obvodu (very large scale integration – velmi vysoká integrace). *Kombinační obvod* (*logický obvod*, *hradlo*) je zařízení pro výpočet *boolovských funkcí*. Obvykle je reprezentován jakožto konečný orientovaný acyklický graf, jehož množina vrcholů se dělí na *vstupní vrcholy*, *vnitřní vrcholy* neboli *brány* a jediný *výstupní vrchol*. Každý vstupní vrchol patří k právě jednomu z argumentů počítané boolovské funkce. Každý z vnitřních vrcholů odpovídá vzájemně jednoznačně příslušné boolovské funkci dvou proměnných použité během výpočtu. Výstupní vrchol pak obsahuje výsledek výpočtu boolovské funkce na vstupních vrcholech.

Řekneme, že *kombinační obvod* C s n vstupními vrcholy *vypočte* boolovskou funkci $f(x_1, \ldots, x_n)$, pokud, v případě, že vstupním vrcholům je přiřazena po řadě posloupnost $x_1, x_2, \ldots, x_n$ a tyto vstupní hodnoty jsou zpracovány vnitřními branami dle zřejmého pořadí indukovaného acyklickým uspořádáním grafu C, je hodnota výstupního vrcholu rovna $f(x_1, \ldots, x_n)$.

Velikost kombinačního obvodu C je počet vnitřních vrcholů a značí se $\mathrm{c}(X)$. Je-li f boolovská funkce, je pak *obvodová složitost* funkce f definována jakožto minimální velikost kombinačního obvodu, který vypočtu (realizuje) funkci f a označuje se jakožto $\mathrm{c}(f)$.

Příklad 5.5.1 (**Hamiltonovská kružnice**) *Uvažme následující otázku: Určete, zda graf G obsahuje Hamiltonovskou kružmici. Pro každou hodnostu n můžeme najít kombinační obvod C_n, jenž má $O(n^2)$ vstupních vrcholů (jeden pro každou možnou hranu), která dává na výstupu výsledek* TRUE *tehdy a jen tehdy, když vstupní graf G má hamiltonovskou kružnici. V současné době, bohužel, všechny známé takovéto kombinační obvody C_n mají exponenciální počet vrcholů.*

Řekneme tedy, že vlastnost π má *polynomiální obvodovou velikost* neboli obsahuje pouze *malé obvody*, pokud existuje posloupnost *kombinačních obvodů* $(C_n : 1 \leq n < \infty)$ a polynom p tak, že C_n rozhodne π na všech možných vstupech velikosti n a velikost $\mathrm{c}(C_n)$ splňuje

$$\mathrm{c}(C_n) \leq p(n) \qquad (1 \leq < \infty).$$

Poznamenejme, že platí:

- Je-li výpočet turingovsky vypočítatelný v polynomiálním čase, pak má malé kružnice.
- Obrácené tvrzení není pravdivé: existují nerekurzivní funkce, které nejsou vypočítatelné žádným turingovským strojem, ale mají malé kružnice.

- Má-li každý NP-těžký problém malé kružnice, má i každý NP-problém malé kružnice.

V dalším budeme každý problém s malými kružnicemi považovat za „snadný“ a šifrovací systém na něm založený nebude považován za bezpečný.

5.6 Náhodné algoritmy

Algoritmus, který má pravděpodobnost chyby méně, než 2^{-100} a který během minuty je schopen identifikovat číslo $2^{400}-593$ jakožto největší prvočíslo menší než 2^{400}, má bezprostřední praktický a estetický důsledek. Takovými jsou například algoritmy pro testování prvočíselnosti od Rabina a dále od Solovaye a Strassena.

Nejprve ilustrujme ideu náhodného algoritmu na příkladě: Předpokládejme, že máme polynomiální výraz v n proměnných, řekněme $f(x_1, \ldots, x_n)$ a že si přejeme ověřit, zda je polynom f identicky nulový. Ověřit to analyticky je neskutečné počítání. Předpokládejme místo toho, že jsme vygenerovali náhodný vektor $(r_1, \ldots, r_n)$ a vyčíslili $f(r_1, \ldots, r_n)$. Pokud $f(r_1, \ldots, r_n) \neq 0$, víme že f je nenulový. Pokud $f(r_1, \ldots, r_n) = 0$, je buď f identicky nulové nebo jsme měli obrovské štěstí s výběrem $(r_1, \ldots, r_n)$. Proveďme tyto kroky několikrát a pokud vždy obdržíme nulu, můžeme tvrdit, že f je identicky nula. Pravděpodobnost toho, že jsme udělali chybu je zanedbatelná.

Je-li π výpočetní problém a x je vstup nebo instance π, řekneme, že *náhodný algoritmus* pro řešení π postupuje následovně. V jistých okamžicích pro řešení instance x problému π algoritmus provede náhodné rozhodnutí. S vyjímkou těchto náhodných rozhodnutí je algoritmus čistě deterministický. Jazyk L je *náhodně rozhodnutelný* v polynomiálním čase neboli leží ve třídě RP, pokud existuje polynom f a polynomiální algoritmus, který pro každý vstup x a každý možný certifikát y délky $f(|y|)$ vypočte hodnotu $\nu(x, y) \in \{0, 1\}$ tak, že

1. $x \notin L$ implikuje $\nu(x, y) = 0$ pro všechna y.
2. $x \in L$ implikuje $\nu(x, y) = 1$ pro alespoň polovici všech možných certifikátů y.

Evidentně
$$RP \subseteq NP$$
a
$$P \subseteq RP.$$

Kapitola 6

Autentičnost a digitální podpisy

6.1 Motivace

V předchozích 3 kapitolách jsme předvedli metody, které mohou pomoci proti **pasívnímu útoku**; pomocí zašifrování lze data pro nepovolané osoby udělat nečitelná. Téma této kapitoly je věnováno metodám proti **aktivnímu útoku**.

Kryptografický protokol spedifikuje, jakým způsobem každá strana začíná a odpovídá na zprávy a to včetně chybných nebo ilegálních zpráv. Protokol může rovněž specifikovat požadavky na nastavení jako je např. nastavení knihovny veřejných klíčů. Strana, která se řídí protokolem, bude ochráněna proti jistým specifikovaným nebezpečím i v tom případě, že ostatní strany se protokolem neřídí.

Je zřejné, že Mr. X může způsobit podstatně větší škodu, jestliže neumí pouze pasívně číst data, nýbrž je dokonce aktivně změnit. Skutečně se u většiny dnešních aplikací v kryptologii požaduje autentičnost dat a ne jejich utajení. Je zvykem rozlišovat 3 základní typy problémů, které vzniknou z různých variant aktivního útoku. Odpovídající „bezpečnostní architektura" byla vytvořena institucí *International Standards Organisation (***ISO***)* v „Security Addendum k referenčnímu modelu ISO".

Nejdříve je třeba ptát se, zda byla zpráva přenesena bez změny nebo zfalšování; jedná se o požadavek **integrity zprávy**. Je-li Mr. X v pozici, že může měnit zprávy, nezbývá než doufat, že příjemce *zpozoruje* případnou změnu. Musí být schopen rozhodnout, zda byla zpráva změněna či nikoliv.

Druhý typ útoku je prvnímu podobný; zde je položen důraz na otázku, zda si příjemce může být jistý, že zpráva skutečně pochází od údajného odesilatele. Mluvíme pak o **autentičnosti zprávy**.

Poslední varianta je **autentičnost uživatele**: Může osoba dokázat svoji identitu? Příjemce potřebuje prostředek, aby se mohl přesvědčit o tom, že skutečně komunikuje s tou osobou, o které si myslí, že je s ní spojen.

6.2 Integrita a autentičnost

6.2.1 Symetrická autentičnost

Ochrana autentičnosti informace sestává z dvou následujících aspektů:

- ochrana původce informace neboli dle terminologie ISO autenticita původu dat,
- skutečnost, že informace nebyla změněna neboli dle terminologie ISO integrita informace.

První aspekt lze prezentovat tak, že je informace načítána např. z harddisku osobního počítače a my implicitně důvěřujeme zdroji informace. Jiným aspektem je časování, umístění do fronty vzhledem k jiným zprávám a určení zprávy. Až donedávna se obecně předpokládalo, že zašifrování informace je dostatečné k tomu, aby se prokázala její autenticita. Použitý argument byl ten, že pokud šifrový text dal po dešifrování smysluplnou informaci, tato by měla vzniknout od někoho, kdo zná tajný klíč, což garantuje autenticitu zprávy a odesílatele. Ve dvou příkladech ukážeme, že tato víra není správná: ochrana integrity závisí na šifrovacím algoritmu a na módu, ve kterém je algoritmus použit.

Vernamova šifra, kde je náhodný klíč přičítán modulo 2 k šifrovému textu nám poskytuje perfektní bezpečnost, ale aktivní útočník může změnit libovolný bit zdrojového textu tím, že změní odpovídající bit šifrového textu. Tato informace analogicky platí pro libovolnou přičítací proudovou šifru a pro OFB mód (Output FeedBack) každé blokové šifry. Částečně toto platí i pro případ, že šifra je použita CFB módu (Cipher FeedBack) nebo CBC módu (Cipher Block Chaining).

Je-li zdrojový text delší než jeden blok zašifrován pomocí blokové šifry v ECB módu, aktivní útočník může snadno přeuspořádat bloky. Jiným příkladem zranitelnosti aktivním útočníkem je zdrojový text zašifrovaný pomocí CFB módu. Vzhledem k synchronizačním vlastnostem každá modifikace šifrového textu způsobí odpovídající modifikaci zdrojového textu a následně zkomolí následující části zdrojového textu. Poté co chyba opustí FB registr, bude šifrový text opět správně dešifrován. Je-li ale modifikována poslední část šifrového textu, je zcela nemožné najít tuto modifikaci. Pokud se zkomolení vyskytne v prostředku zdrojového textu, lze chybu detekovat pomocí redundance.

V jiných módech (jako např. CBC mód) je každý šifrový text složitou funkcí předchozích bitů zdrojového textu a nějaké počáteční hodnoty. Pokud modifikace jednoho bitu šifrového textu způsobí zkomolení t bitů zdrojového textu, pravděpodobnost, že že nový zdrojový text bude akceptován jako smysluplný je rovna 2^{-tD}, kde D je redundance informace. V případě přirozeného jazyka zakódovaného pomocí 5 bitů na charakter je redundance na bit $D \simeq 0.74$ a tato pravděpodobnostje rovna $2^{-22.2}$ pro $t = 30$. Avšak, je-li $D = 0$ a zašifrování neposkytuje žádnou autenticitu, jsou všechny zprávy smysluplné a to nezávisle na šifrovacím algoritmu nebo na módu šifrování. To pak znamená, že útočník může modifikovat zprávy nebo padělet zprávy dle svého výběru. Omezení je pak to, že útočník neví dopředu, co bude obsahem odpovídajícího zdrojového textu, ale pro mnohé aplikace lze takovýto útok považovat za zdroj vážných problémů. Poznamenejme, že i v případě existence redundance se požaduje kontrola lidským faktorem nebo vhodným počítačovým programem.

Abychom zajistili integritu zprávy, je nutno přidat speciální redundanci, a je-li informace spojena s původcem zprávy, musí být použít v tomto procesu tajný klíč (to předpokládá spojení osoby a jejího klíče) nebo zvláštního kanálu pro zajištění integrity. Můžeme pak identifikovat dvě základní metody.

- ◇ První metoda je analogická metodě symetrické šifry, kde utajení velkého množství dat je založeno na utajení a autenticitě krátkého klíče. V tomto případě autenticita informace závisí na utajení a autenticitě klíče. Abychom dosáhli tohoto účelu, informace se zkomprimuje na kvantitu pevné délky, kterou nazýváme *hešovacím kódem*. Poté se hešovací kód připojí k informaci. Funkce, která provede tuto operaci komprese, se nazývá *hešovací funkce*. Základní myšlenkou zabezpečení integrity je *přidat redundanci* k informaci. Přítomnost redundance dovoluje příjemci provést rozlišení autentické informace a podvodné informace.

  Abychom garantovali původ dat, je nutno v procesu použít tajný klíč. Tajný klíč může být obsažen v procesu komprese nebo může být použit, aby ochránil hešovací kód a/nebo informaci. V prvním případě mluvíme o MACu (Message Authentication Code), zatímco v druhém případě se hešovací kód nazývá MDC (Manipulation Detection Code).

- ◇ Druhá metoda sestává na zajištění autenticity (jak integrity a autenticity původu) informace o autenticitě MDC. Typickým příkladem této metody je uživatel počítače, který počítá MDC pro všechny své důležité soubory. Může si pak uložit soubor všech MDC na disketě, kterou si bezpečně uschová. Pokud tyto soubory zašle vzdálenému příteli, může jednoduše poslat soubory a sdělit příteli po telefonu jejich MDC. Autenticita telefonního kanálu je zajištěna hlasovou identifikací.

Přidání redundance není jistě dostatečné. Speciální důraz musíme klást na obranu proti útokům na vysoké úrovni, jako je například opakování autentifikovanézprávy.

Oba případy nefungují, pokud si odesílatel a příjemce navzájem nedůvěřují. V prvním případě sdílejí stejný tajný klíč. Pokud jedna ze stran tvrdí, že informace byla změněna druhou stranou, nemůže soudce rozhodnout, kdo má pravdu, i když obě strany vydají společný tajný klíč. Druhý přístup může pouze zajistit nepřevzetí, pokud obě strany věří autenticitě MDC: v praxi je to však obtížné realizovat, protože obě strany mají podobný přístup ke kanálu.

6.2.2 Asymetrická autenticita

Jestliže chceme být ochráněni proti vnitřnímu napadnutí, potřebujeme elektronický ekvivalent podpisu. V tomto případě třetí strana bude schopna rozlišit dvě strany a to na základě skutečnosti, že způsobilosti obou stran jsou různé. Pojem digitálního podpisu byl zaveden W. Diffiem a M. Hellmanem (blíže viz 7.2). Požadavky na elektronický podpis jsou, že podpis závisí na podepisované informaci (protože není fyzicky spjat s dokumentem) a že podepsaný je jediná osoba, která je schopna vytvořit podpis (to znamená, že nikdo jiný nemůže zfalšovat podpis tj. podepsaný nemůže zapřít, že informaci podepsal právě on). Digitální podpisové schéma sestává z následujících prvků:

- ◇ inicializační fáze (např. generování klíče a obecné nastavení),
- ◇ procesu podpisu, kdy je vytvořen podpis,
- ◇ procesu verifikace, kdy příjemce (nebo soudce) ověří, zda je podpis správný.

Digitální podpis v tomto smyslu lze vytvořit pomocí zařízení bezpečných proti falšování, konvenčních jednosměrných funkcí nebo technik veřejného klíče.

Poznamenejme dále, že bylo definováno několik zobecnění – např. s různými stupni bezpečnosti a více hráči ve hře. Příklady takovýchto jsou následující: *libovolné podpisy*, kde proces podpis a verifikace zahrnuje interakci s třetí stranou, *skupinové podpisy*, kde podpisující a/nebo kontroloři jsou členy skupiny, *slepé podpisy*, kde podpisující podepíše „slepou“ nebo „ maskovanou“ zprávu a *neviditelné* nebo *nepopiratelné* zprávy, kde lze podpis verifikovat pouze ve spolupráci s podpisujícím.

6.2.3 Message-Authentication-Code

Připomeňme si, že při integritě a autentičnosti zprávy jde o to, abychom vyvinuli metody, které příjemci umožní rozhodnout, zda zpráva došla neporušená a autentická. K tomu potřebuje příjemce něco, s čím může být zpráva ověřena: Potřebuje dodatečnou informaci od odesilatele. Takový informační blok se nazývá **kryptografický zkušební součet** neboli **Message-Authentication-Code**, zkráceně **MAC**.

Protokol k vytvoření a verifikaci kryptografického zkušebního součtu je založen na použití tajného klíče k, který je znám jak odesilateli tak příjemci, a kryptografickém algoritmu A, který budeme v dalším diskutovat.

Odesílatel neposílá pouze holou zprávu M, nýbrž dodatečně příslušný MAC; ten se vypočte pomocí klíče k algoritmem A ze zprávy M následovně:

$$\mathrm{MAC} = A_k(M).$$

Poznamenejme, že M je odesíláno nezašifrované, protože cílem odesílatele není utajit obsah zprávy, nýbrž zprávu zabezpečit. Pokud chceme navíc důvěrnost, musí být m a MAC zašifrovány.

Nyní přijde na řadu příjemce. Jeho zájem je zjistit, zda přijatá zpráva souhlasí se zprávou odeslanou a zda skutečně pochází od uvedeného odesilatele. Aby to provedl, simuluje proceduru odesilatele: Použije algoritmus A s klíčem k na přijatou zprávu M' a prověří, zda výsledek souhlasí s obrženým MACem.

Je-li $A_k(M') \neq MAC'$, ví příjemce, že se „něco" stalo: proto neakceptuje zprávu jako autentckoi a odmítne ji. Je-li ale $A_k(M') = MAC'$, může si být dostatečně jistý, že zpráva nebyla změněna. Přirozeně tato jistota závisí ve velké míře na kvalitě algoritmu A a velikosti množiny možných klíčů. Představy o MAC–mechanismu jsou následující:

- Podvodu ze strany Mr. X bude zamezeno, protože nezná klíč k. Musel by totiž spočítat odpovídající MAC pro svou zprávu.
- Příjemce může pouze rozpoznat, či je zpráva neporušená a autentická; v záporném případě nemá žádnou možnost zrekonstruovat původní zprávu. To znamená, že v tomto případě je nutný nový přenos zprávy.
- MAC–mechanismus je metoda k dosažení integrity a autentičnosti. Už jsme viděli, že lze poznat integritu. Poku proběhne verifikace kladně, je příjemce rovněž přesvědčen o autentičnosti zprávy, protože odesilatel je jedinou jinou instancí, která zná tajný klíč.

*

Jaké algoritmy A můžeme použít k výpočtu MACu? Okamžitá odpověď je jednoduchá: použijme jednoduše šifrovací algoritmus, přičemž MAC je kryptogram, který odpovídá zprávě M. Odhlédneme-li od toho, že takovýto slabý algoritmus není vhodné doporučit, má tento návrh tu nevýhodu, že přenášená data jsou dvojnásobně delší než „vlastní zpráva". Přirozeně každý MAC prodlouží zprávu, ale chtěli bychom délku tohoto dodatečného bloku držet v nějakých rozumných mezích.

V praxi používáme k výpočtu MACu také šifrovací algoritmus, ale ne přímo, nýbrž v tzv. **Cipher-Block-Chaining módu**. Představme si šifrovací algoritmus f (v praxi se většinou používá algoritmus DES), který zobrazuje bloky zprávy složených z n znaků pomocí nějakého klíče K na bloky kryptogramu, rovněž složených z n znaků (typická hodnota je $n = 64$). Abychom mohli vypočítat MAC, rozdělíme zprávu M do bloků M_1, M_2, $\ldots$, M_s délky n. Pak aplikujeme f na blok M_1 a obdržíme první blok kryptogramu $C_1 = f_K(M_1)$. Potom přičteme C_1 k M_2 a položíme $C_2 = f_K(C_1 \oplus M_2)$. Tento postup opakujeme až skončíme výstupem $C_s = f_K(C_{s-1} \oplus M_s)$, který vybereme za MAC. Takto vypočtený MAC má následující přednosti:

– MAC má pevnou délku n nezávislou na délce zprávy.

– MAC závisí na všech blocích zprávy.

Protože všechny možné zprávy jsou zkomprimovány na MACy pevné délky, má mnoho zpráv tentýž MAC. To nepředstavuje žádný problém pro příjemce, protože ten nemusí rekonstruovat původní zprávu z MACu.

Ne všechny algoritmy jsou vhodné k tomu, aby byl Mr. X postaven před nepřekonatelné problémy. Algoritmus pro výpočet MACu by měl mít následující vlastnosti.

1. Mělo by být prakticky nemožné najít pro daný MAC odpovídající zprávu (pokud tato vlastnost platí, nazýváme algoritmus **jednosměrnou** (one-way) **funkcí**). „Prakticky nemožné" znamená, že s dnešními metodami a počítači by vyřešení problému trvalo příliš dloho (několik století).

2. Mělo by být prakticky nemožné najít dvě zprávy, které mají tentýž MAC (jednosměrná funkce splňující tuto podmínku se nazývá **bezkolizní**).

Je velmi obtížné podat precizní matematickou definici jednosměrné funkce. Neformálně je *jednosměrná funkce* funkce $f : S \to T$, kde S a T jsou množiny takové, že
(1) pro všechna $x \in S$ je $f(x)$ „snadno" vypočitatelné,
(2) máme-li k dispozici informaci, že $f(x) = y$, neexistuje žádný „přiměřený" způsob
jak získat (výpočtem) x pro „dostatečně velké" množství prvků y z T.

Pracovní slova zde jsou „snadno", „přiměřeně" a „dostatečně velké".

Je zřejmé, že je-li dáno $f(x)$, jeden způsob, jak získat x je prohledávat všechny možné hodnoty $x \in S$. Nepovažujeme to za přiměřené, protože S sestává obvykle z posloupnosti binárních řetězců délky $n \sim 200$.

Požadujeme, že výpočet pro nalezení x ze znalosti y je příliš dlouhotrvající nebo nákladný, kdykoliv y leží v „dosti velké" podmnožině množiny T.

Příklad. Elementárním příkladem kandidáta na jednosměrnou funkci je, pro dostatečně velké prvočíslo p, funkce $f(x)$, kde $f(x)$ je polynom nad tělesem $\mathbf{Z}_p$. Pak je relativně snadné vypočíst $f(x)$ $(1 \leq x \leq p-1)$, ale obvykle je těžké nalézt řešení rovnice

$$f(x) = y.$$

Výše uvedená nepřesná definice znamená, že co je jednosměrná funkce se mění s dobou. Například, výpočet požadující milión instrukcí a 1O OOO slov paměti nemohl být v roce 1950 považován za snadný, ale nyní by trval několik sekund na osobním počítači. Tedy funkce považovaná v roce 1950 za jednosměrnou nemusí být za ni považovaná nyní.

Jedna metoda podání formální definice by mohlo být užitím fyzikálního přístupu. Např. 10^{60}-bitová paměť vždy zůstane nedosažitelnou, protože i kdybychom potřebovali pouze jednu mulekulu na bit paměti, její konstrukce by vyžadovala více hmoty než existuje v slunečním systému. Podobně, termodynamika nám dává omezení maximálně 10^{70} operací, které lze provést s využitím celkové energie slunce.

Nižší úroveň nedosažitelnosti je použití myšlenek výpočetní složitosti. Nejdříve uvažujme některé z vlastností, které bychom rádi požadovali po jednosměrné funkci.

(I) Výpočet $f(x)$ z x musí být přiměřený: vyjádříme to tím, že f je vypočitatelná v polynomiálně omezené době (říkáme, že $f \in \mathrm{P}$.

(II) Výpočet f^{-1} nesmí být snadné; budeme tudíž požadovat, že není znám žádný algoritmus pro výpočet f^{-1} v polynomiálně omezené době.

(III) Třetí podmínka bude tzv. *upřímnost* funkce tj., že existuje polynom p splňující $|x| \leq p(|f(x)|)$.

Poslední podmínka je technická podmínka pro vyloučení funkcí jako

$$f(x) = \lceil \log \log x \rceil,$$

která zcela jistě splňuje (I) a (II), ale kterou bychom nemohli obvykle považovat za jednosměrnou funkci. Funkce f splňující (I),(II) a (III) se nazývá slabě jednosměrná funkce.

Příklad. Nechť $\mathbf{I}_k$ značí množinu všech k-bitových přirozených čísel tj.

$$\mathbf{I}_k = \{2^{k-1}, \dots, 2^k - 1\} \qquad (k = 1, 2, \dots).$$

Nechť $\mathbf{S}_k = \mathbf{I}_k \times \mathbf{I}_k$ a nechť $f : \mathbf{S}_k \to \mathbf{Z}^+$ je definována jako

$$f(m, n) = m \cdot n.$$

Položíme-li $\mathbf{S} = \bigcup\{\mathbf{S}_k : 1 \leq k < \infty\}$ a rozšíříme-li f na $\mathbf{S}$, získáme slabě jednosměrnou funkci. Přitom v současné době není známo, že by inverzní funkce ležela v $\mathbf{P}$.

Není lehké najít matematickou explicitní definici jednosměrné funkce. Uveďme následující příklad.

Příklad. Buď F šifrovací algoritmus, který zobrazuje zprávu M pomocí klíče K na kryptogram C, $F_K(M) = e(M, K) = C$ a předpokládejme, že $\mathbf{M} \subseteq \mathbf{K}$. Změňme trochu tuto funkci. Zafixujme za tímto účelem (ne tajnou) *zprávu* M_0 (např. $M_O = 00\dots0$); tato „zpráva" se objeví v algoritmu místo obvyklé zprávy, nehraje ale roli variabilní zprávy. Variabilní zpráva se vloží do algoritmu F na místě klíče. Krátce: uvažujeme funkci

$$f = e(M_0, -) : \mathbf{M} \to \mathbf{C}.$$

Tvrdíme pak, že f je jednosměrná funkce. Představme si, že Mr. X zná jak M_0 tak i C a chtěl by nají M. V řeči šifrovacího algoritmu F to lze vyjádřit následovně: Mr. X zná sobě odpovídající dvojici zpráva-kryptogram (M_0, C) a chtěl by vypočíst klíč. Je-li algoritmus F kryptologicky bezpečný, je rezistentní proti tomuto known-plaintext útoku a proto je f jednosměrná funkce.

Položme si otázku, zda lze *matematicky dokázat*, že tato funkce f je jednosměrná. Odpověď zní: Ne! Matematici nemohli ještě o žádné funkci dokázat, že je jednosměrná. To znamená, že neznáme žádnou funkci, jejíž funkční hodnoty lze spočítat v polynomiálně omezené době, ale která při výpočtu funkce inverzní potřebuje exponenciální dobu. Nevíme tedy, zda teoreticky jednosměrné funkce existují. Pro praktické účely mají ale výše popsané funkce dostatečně dobré vlastnosti.

6.3 Autentičnost uživatele

Základní úloha bezpečnostní techniky je spolehlivá identifikace osob tj. autentifikace. Dřívo to byl výlučně proces mezi dvěma lidmi, který je dnes rozšířen na proces mezi člověkem a počítačem. Přitom vznikají přirozeně problémy; ale, jak uvidíme, lze i s počítačem provádět velmi dobrou autenticitu uživatele. Nejprve si připomeňme, jak je tento problém řešen mezi lidmi. Zásadně lze lidi poznávat podle následujících znaků:

– Osobu lze identifikovat podle *charakteristických vlastností.*

– Osobu lze identifikovat podle *vlastnictví.*

– Osobu lze identifikovat podle *znalosti.*

První mechanismus je v běžném životě používán tak často, že si ho ani nejsme vědomi: poznáváme naše známé podle jejich obličeje, hlasu, chůze atd. V jistých situacích se používají za účelem obzvlášť spolehlivé identifikace otisky prstů. Následující metody se používají jen ve speciálních situacích. V druhém kole kupónové privatizace je nutno předložit občanský průkaz, kde je uvedeno české občanství; při přechodu hranic se musíme prokázat pasem; platíme-li kreditní kartou, lze ověřit identitu jejím vlastnictvím atd. Identifikace na základě znalosti je prováděna velmi zřídka – ačkoliv tento mechanismus je znám již z nejstarších dob (vojáci musí znát současné heslo, aby se mohli identifikovat).

Při procesu autenticity mezi člověkem a počítačem je situace zcela jiná. Autenticita na základě znalosti je mechanismus, který lze nejjednodušeji realizovat;

autenticita na základě vlastnictví je rovněž možná. Naproti tomu je autenticita podle charakterostických vlastností velmi komplexní a používá se pouze při aplikacích, které vyžadují extrémně vysokou bezpečnost. Proto se budeme výhradně zabývat s metodami založenými na znalosti resp. vlastnictví. Při použití těchto metod má uživatel nějaké tajemství a počítač se chce přesvědčit, že osoba má skutečně jí odpovídající tajemství.

Hesla

Klasická metoda, se kterou se osoba prokáže stroji, je založena na heslech. *Heslo* je libovolná posloupnost znaků (písmena, číslice, speciální znaky), která je výlučným tajemstvím osoby a počítače. To znamená, že v ideálním případě znají jen uživatel a počítač příslušné heslo.

Základní myšlenka je jednoduchá: počítač uložil „referenční heslo“ P_0, které je známo uživateli. Když chce uživatel prokázat svou antenticitu (např. přístup k danému počítači), napíše své heslo P a počítač porovná, zda jsou P a P_0 totožné. Uživatel je připuštěn, jestliže $P = P_0$.

Je mnoho problémů s hesly, obzvláště takové, které souvisí se špatnám zacházením s hesly. My se budeme věnovat jen těm problémům, které lze řešit pomocí kryptologických prostředků. To jsou problémy, které se týkají ukládání hesel.

V případě, že má Mr. X čtecí práva na soubor obsahující hesla (nebo si ho opatří), může číst všechna hesla a tím se stát libovolným uživatelem. Výše popsaná procedura je pouze tehdy bezpečná, jestliže je soubor obsahující hesla uložen jako nečitelný – a to je požadavek, který každý systémový programátor odmítne jako nerealizovatelný. Přirozeně nám napadne *zašifrovat hesla*. Když to provedeme „naivně“, navrhneme následující protokol: počítač uloží zašifrovaná hesla $P^* = e(P_0, K)$,; při ověření autenticity uživatele se P^* rozšifruje a srovná se s heslem zadaným uživatelem.

Tato procedura má jasnou výhodu. Dokonce i když Mr. X může číst soubor obsahující hesla, může je číst pouze zašifrovaná. Protože ale není schopen je dešifrovat, nemůže simulovat jednotlivého uživatele vložením správného hesla.

Je tento argument správný? Přirozeně jsme podstatně redukovali množinu tajných dat; namísto neohraničeného počtu hesel, která mají být tajně uložena, musíme v šifrovacím modelu jen uložit tajně klíč K. To je nepochybně velká výhoda, ale není to žádné principiální řešení problému: Pokud si Mr. X opatří klíč K, může dešifrovat všechna zašifrovaná hesla a má tak stejně jako předtím možnost stát se libovolným uživatelem.

Po takovéto analýze je pro tvůrce přístupového systému založeného na heslech těžké vyjádřit své přání: chtěl by mít k dispozici funkci, která „zašifruje“ hesla bez toho, že by použila tajný klíč – to není nic jiného, než naše známá jednosměrná funkce.

Nyní použijme takovouto jednosměrnou funkci f. V paměti počítače uložme hodnotu $P^{\#} = f(P_0)$. Během procedury ověření autenticity se aplikuje f na posloupnost znaků P, kterou vloží uživatel. Pak počítač prověří, zda platí $P^{\#} = f(P)$.

Přednost této metody je zřejmá: soubor obsahující hesla je nedešifrovatelný a není žádné tajmeství. Je pozoruhodné, že jednosměrné funkce byly vyvinuty proto, aby ukládaly hesla v nečitelné formě; tato metoda byla v šedesátých letech navržena a realizována Rogerem **Needhamem** z University Cambridge.

Přirozeně nechrání takováto procedura proti špatně zvoleným heslům. Mr. X se může totiž pokusit o „chosen password"-útok: Má-li seznam nejčastěji používaných hesel (dívčí jména, telefonní čísla, data narození,...), může aplikovat f na tato populární hesla a výsledky srovnat s obsahem souboru obsahující pozměněná hesla. Empirické výsledky ukazují, že Mr. X může tímto způsobem získat podstatné procento všech hesel.

Autenticita pomocí čipových karet

Žádný systém založený na heslech nesplňuje extrémně vysoké bezpečnostní požadavky. To spočívá zcela jednoduše na tom, že lidé nemají schopnost si pamatovat hesla; aby si člověk zapamatoval heslo (bez toho, že by si je bokem zapsal), musí se jednat o krátké a přehledné slovo. Mimo to je ochrana pomocí hesel statická metoda pro určení autenticity. Přiroozeně může počítač nutit uživatele k pravidelné obměně hesel, např. každý měsíc. To je ale jen malý pokrok: Stejné heslo se vždy použije vícekrát. Proč je to problém? Mr. X jednou odposlechne přenos hesla a od té chvíle se může vydávat za uživatele.

Abychom vyřešili tento problém, musí se data, která jsou vyměňována mezi počítačem a uživatelem neustále měnit – pokaždé nová otázka a nová odpověď. Je jasné, že uživatelova odpověď se musí formulovat tak, že nikdo jiný nemůže tuto odpověď poskytnout. To znamená, že reakce uživatele na otázku musí záviset na nějakém tajemství, jinak by totiž na ni mohl odpovědět i Mr. X.

Odpovídající protokol podle metody **Challenge-and-Response** probíhá následovně. Jak počítač tak uživatel mají k dispozici jednosměrnou funkci f a společný tajný klíč K. Počítač obdrží od uživatele data k identifikaci; v našem případě třeba jméno uživatele A. Nato si počítač opatří pro dané jméno příslušný klíč K. (Klíč K může být uložen v seznamu, může být odvozen systémovým globálním klíčem nebo dán k dispozici pomocí podobné procedury.)

Počítač se musí přesvědčit o tom, že uživatel, který se chce identifikovat, je skutečně uživatel A a ne zákeřný Mr. X. Pokud může uživatel dokázat, že má správný klíč K, je automaticky uznán za autentického. Tj. považujeme za dokázané, že se skutečně jedná o uživatele A. Cílem počítače v následující popsané proceduře autenticity je „nepřímo" se přesvědčit, že uživatel je ve vlastnictví klíče K. V popisu protokolu označme klíč, který je ve vlastnictví uživatele (to by mohl být i Mr. X) jako K'.

Za tímto účelem klade počítač uživateli otázku tím, že mu zašle náhodně zvolené číslo $RAND$. Pak uživatel vypočte **parametr autenticity** $AP' = f_{K'}(RAND)$

a zašle jej počítači. Současně počítač vypočte hodnotu $AP = f_K(RAND)$ a porovná zda $AP = AP'$. Pokud ne, pak něco nesouhlasí, v opačném případě má počítač s vysokou pravděpodobností stejný klíč jako uživatel. Tedy uživatel bude akceptován.

Tato procedura má dvě výhody a dvě nevýhody. Protože je číslo $RAND$ náhodně voleno, mění se případ od případu a Mr. X nemá žádnou šanci předpovědět následující $RAND$. Také AP má charakter náhodného čísla, takže Mr. X nikdy nemůže na otázku dát správnou odpověď. Právě tak důležitá je skutečnost, že pomocí protokolu se prokáže, že oba klíče jsou identické, ale klíče samotné se uvnitř protokolu nikdy neobjeví.

Nevýhody Challenge-and-Response protokolu jsou sice „malé“, ale nesmějí být jak prakticky tak teoreticky zanedbány. Uvědomme si, že počítač vlastní přístup k tajným klíčům uživatelů. To počítači umožňuje, aby se vydával vzhledem k jiným instancím (nebo vůči sám sobě) za uživatele. Jinak řečeno: Předpoklad pro použití Challenge-and-Response protokolu je, že uživatelé důvěřují integritě počítače provádějícího autenticitu.

Jiná nevýhoda spočívá v tom, že odesilatel A musí provést algoritmus f. Provedení kryptografického algoritmu však přesahuje znalosti a možnosti každého lidského stvoření; tyto věci bychom měli přenechat elektronickému sluhovi. Takovýmto sluhou může být např. tzv. „čipová karta“. Problém je však v tom, aby čipová karta rozpoznala, že ji chce použít oprávněný uživatel A a ne Mr. X. Musí tedy od uživatele vyžadovat, aby vůči ní prokázal totožnost.

6.3.1 Zero-Knowledge protokol

Zero-Knowledge protokol je procedura, která jedné straně dovolí přesvědčit druhou stranu, že má určité tajemství, aniž by prozradila o tomto tajemství nejmenší informaci.

Zero-knowldege protokol je dvoustranný protokol; jedna strana se nazývá *dokazovatel*, druhá *prověřovatel*. Dokazovatel zná nějakou skutečnost a přeje si přesvědčit prověřovatele o této skutečnosti. Prověřovatel chce protokol, který mu dovolí se přsvědčit o platnosti této skutečnosti právě tehdy, když je tato skutečnost pravdivá. Přesněji, dokazovatel (jestliže jedná podle protokolu) bude schopen přesvědčit prověřovatele o platnosti skutečnosti, pokud je tato skutečnost pravdivá, ale dokazovatel (dokonce i když se pokusí podvádět) nebude mít žádnou podstatnou šanci v přesvědčení prověřovatele o platnosti skutečnosti, pokud tato neplatí. Přitom si dokazovatel nepřeje podat žádnou informaci o podstatě skutečnosti (např. důvody proč skutečnost platí).

Uveďme tři jednoduché příklady.

Historický příklad: Tartagliovo tajemství

Řešení rovnic byl v historii matematiky nanejvýš důležitý problém. Jednoduché bylo řešení lineárních a kvadratických rovnic. Kubická rovnice, tedy rovnice tvaru

$$ax^3 + bx^2 + cx + d = 0$$

nebyla tak snadno řešitelná a nalezení jejího řešení trvalo podstatně déle. Benátský matematik Niccolo **Tartaglia** (1499-1557) objevil podle svých údajů v roce 1535 metodu jejího řešení. Veřejně oznámil svůj objev, ale vzorec pečlivě tajil. Tartaglia snadno ukázal, že je schopen kubickou rovnici řešit a tím každého bez obtíží přesvědčil, že jeho řešení je správné.

Ačkoliv se Tartagliovi konkurenti mimořádně snažili, nebyli schopni odhalit jeho tajemství. Až Geronimo **Cardano** (1501-1576) přesbědčil Tartagliu, aby mu vzorec ukázal. Slíbil, že ho udrží v absolutní tajnosti. Stalo se. co se stát muselo: Ve své knize *Ars Magma* (1545) zveřejnil Cardano Tartagliův vzorec; čestně však popsal, že tato formule pochází od Tartaglii. Přesto zůstává ironií osudu, že se dnes vzorec pro řešení kubické rovnice nazývá **Cardanova formule**.

Pro nás je důležité, že Tartaglia měl tajemství (metodu řešení kubických rovnic), které byl schopen utajit a o jehož existenci mohl přesvědčit ostatní.

Odmocninový příklad

Uvažme hru probíhající následovně: Máma hráče A a hráče B. Hráč A tvrdí, že zná číslo s, které hráč B nezná, a s přitom splňuje, že $s^2 \equiv 34 \bmod 55$. Hráč A chce hráče B přesvědčit, že toto číslo opravdu zná. Proto náhodně zvolí číslo r, umocní je na druhou a spočítá zbytek po dělení 55, řekněme např. 26 a ukáže jej hráči B.

Nyní přichází na řadu hráč B. Hodí si mincí; pokud padne hlava, chce znát od prvního hráče $r \cdot s \bmod 55$, jinak pouze r.

Předpokládejme, že padla hlava. Pak řeknu, že $r \cdot s \bmod 55 = 53$, a to lze jednoduše prověřit, neboť skutečně platí

$$53^2 \bmod 55 = 4 = (26 \cdot 34) \bmod 55 = r^2 \cdot s^2 \bmod 55.$$

Uvědomme si, že je právě tak těžké najít s, pokud navíc známe $r^2 \bmod 55$ a $r \cdot s \bmod 55$, jako kdybychom je obě neznali. Připomeňme, že r nesmí být zcela náhodně zvoleno; například by r mělo být větší než $\sqrt{55}$.

Lze u této hry podvádět? Ano - v případě, že hráč B neobjeví podvod. Ale po čase se na *alespoň jeden podvod* přijde.

Představme si, že hráč A podváděl a číslo s nezná. Kdyby věděl, že mince hráče B neukáže hlavu, neměl by žádný problém ukázat hráči B číslo r. Avšak A by nevěděl, co udělat, kdyby hráč B chtěl znát $r \cdot s \bmod 55$!

Kdyby hráč A věděl, že mince hráče B ukáže hlavu, mohl by A také podvádět; nejprve by si vybral číslo, o kterém by věděl, že jeho zbytek po dělení 55 je čtverec, např $2^2 = 4$ a zvolil by pak

$$r^2 = (4/s^2) \bmod 55 = (4/34) \bmod 55 = 26.$$

V případě, že by hráč B chtěl znát $r \cdot s \bmod 55$, odpoví jednoduše 2 a hráč B se může přesvědčit, že platí

$$2^2 = (26 \cdot 34) \bmod 55 = (4/34) \cdot 34 \bmod 55.$$

Když ale hráč A musí říci číslo r, okamžitě se přijde ne jeho podvod.

Celkem tedy můžeme říci, že hráč A má v každém kole 50%-ní šanci úspěšného podvodu. Aby se tedy hráč B přesvědčil, že hráč A nelže, musela by se hra hrát více kol.

Tato hra ukazuje rozhodující vlastnosti skutečného Zero-Knowledge protokolu:

- Protokol je **interaktivní**; oba partneři provádí náhodné volby (hráč A volí náhodné číslo r, hráč B se rozhodne, zda chce znát r nebo $r \cdot s \bmod 55$).
- Pravděpodobnost úspěšného podvodu závisí na počtu odehrátých kol. V každém kole se pravděpodobnost zmenší na polovičku.

Příklad tříbarevného obarvení grafu

Uvažme následující hru dvou hráčů A a B. Hráč A chce přesvědčit hráče B, že jistý graf je obarvitelný třemi barvami, aniž by mu prozradil konkrétní obarvení. Přitom hráč A to může provést v posloupnosti $|E|^2$ stavů zadaných následovně:

- Hráč A náhodně přebarví tyto tři barvy (např. všechny červené uzly na modro, všechny žluté uzly na červeno a všechny modré uzly na žluto).
- Hráč A zašifruje barvu každého uzlu pomocí použití šifrovacího schématu s různou pravděpodobností pro každý vrchol. Potom ukáže hráči B všechna tato zašifrování spolu s předpisem přiřazujícím zašifrování s odpovídajícím vrcholem.
- Hráč B vybere hranu grafu.
- Hráč A provede dešifrování barev dvou uzlů této hrany odkrytím odpovídajících šifrovacích klíčů.
- Hráč B potvrdí, že dešifrování bylo provedeno správně a že dva koncové uzly hrany jsou obarveny dvěma různými, ale legálními hranami.

Je-li graf skutečně 3-barevný (a hráč A zná obarvení), pak hráč B nikdy nezjistí žádnou špatně obarvenou hranu. V případě, že graf není 3-barevný, pak existuje šance alespoň $|E|^{-1}$ v každém stavu, že hráč A se hráče B pokouší podvést. Šance, že by hráč A mohl hráče B podvést v $|E|^2$ krocích je exponenciálně malá.

Poznamenejme, že historie naší komunikace – v případě, že graf je 3-barevný – sestává se zřetězení zpráv odeslaných během každého stavu. Je možné dokázat, (za předpokladu, že je možné perfektní zašifrování), že pravděpodobnostní rozdělení definované nad těmito průběhy naší množinou možných interakcí je nerozeznatelné v polynomiálním čase od rozdělení, které můžete vytvořit nad těmito průběhy samotnými, bez účasti hráče A. To znamená, že hráč B nezíská jinou vědomost z protokolu, než že graf je skutečně 3-barevný.

Nyní se věnujme skutečnému protokolu.

Fiat-Shamirův protokol

V roce 1986 předložili izraelští matematici Adi **Shamir** a Amos **Fiat** protokol, který otevřel nové rozměry v určování autenticity uživatele. Přesněji, jedná se o určení autenticity typu počítač – počítač, přičemž jedním z počítačů je čipová karta uživatele. Fiat-Shamirův protokol je založen na výsledcích Shafi **Goldwassera** a Silvia **Micaliho** z Massachusettského technologického institutu a rovněž Charles **Rackoffa** z Univerzity Toronto. Verze, kterou nyní popíšeme, je zobecněním odmocninového příkladu.

Bezpečnost protokolu spočívá na tom, že je mimořádně obtížné najít druhou odmocninu nějakého čísla v modulo n. Přesněji, buď dáno přirozené číslo n, které není prvočíslo. *Jestliže neznáme rozklad n na součin prvočísel, je prakticky nemožné najít číslo s tak, že $s^2 \bmod n = v$.* Doporučeje se volit n tak veliké, aby bylo řádově asi 10^{200}; to znamená, že normálním písmen je n dlouhé asi 1 m.

Před vlastním procesem určení autenticity uživatele pomocí **Fiat-Shamirova protokolu** se zvolí v šifrovací centrále dvě prvočísla p a q a utvoří se jejich součin $n = p \cdot q$. Je rozhodující, že centrála drží p a q v tajnosti, zatímco n je veřejně známo. Proto musí být n tak veliké, že faktorizace n je odsouzena k neúspěchu. Důvod proto je, že můžeme relativně jednoduše určit druhé odmocniny z $\bmod p$ a $\bmod q$. Z těchto čísel lze pak snadno získat druhou odmocninu $\bmod n$.

Centrála spočítá pro každého uživatele číslo s (které je tajemství uživatele) a číslo v tak, že platí $v = s^2 \bmod n$. Číslo v slouží k veřejné identifikaci uživatele. Centrála by mohla zvolit pro v identifikační údaje uživatele (v binární podobě) a odtud vypočítat s. Číslo n je veřejná systémová konstanta.

Tímto jsou už všechny úlohy centrály popsány. Zejména už centrála nehraje při aktuálních procesech určení autenticity žádnou úlohu.

Popis Fiat-Shamirova protokolu

Uživatel		**Počítač**
Uživatel **náhodně zvolí** číslo r, které je nesoudělné s n a vypočte $x = r^2$ **mod** n**.**		
$\boxed{x}$	$\longrightarrow$	
		Počítač náhodně vybere bit b.
Uživatel v případě, že $b = 1$ položí $y := r \cdot s$ **mod** n; v případě, že $b = 0$ položí $y := r$ **mod** n.		
$\boxed{y}$	$\longrightarrow$	
		Počítač prověří, že y je nesoudělné s n;v případě, že $b = 1$ prověří, zda y^2 **mod** $n = x \cdot v$ **mod** n; v případě, že $b = 0$ prověří, zda y^2 **mod** $n = x$.

Přitom zřejmě má Fiat-Shamirův protokol následující vlastnosti:

- Oba počítače musí provést jen několik málo výpočtů **mod** n: na straně uživatele se musí pouze umocnit na druhou náhodné číslo r; v polovině všech případů se musí ještě spočítat $r \cdot s$. Počítač provádějící určení autenticity uživatele musí umocnit na druhou číslo y a v polovině všech případů vynásobit x s v.

- Počítač používá pouze veřejně dostupné informace, zatímco uživatel podstatným způsobem používá tajemství znalosti s.

- Počítač se za jistou dobu přesvědčí, že uživatel zná opravdu tajemství s. Pravděpodobnost, že by nepřítel, který nezná tajemství s pokaždé předpověděl, který bit b bude zvolen, je při t-násobném opakování protokolu jen $1/2^t$. V případě, že $t = 20$, je tato pravděpodobnost méně než 1 ku miliónu.

- I po provedení určení autenticity uživatele zůstane tajemství s absolutně utajeno.

- Bezpečnost protokolu je v rozhodné míře závislá na tom, že výpočet odmocnin $\mathbf{mod}\ n$ je tak obtížné, že ani sám Mr. X není schopen vypočítat druhou odmocninu z v – a to ani tehdy ne, když odposlechne tisíce transakcí mezi uživatelem a počítačem.

6.4 Čipové karty

Co je to čipová karta? **Čipová karta** je plastiková karta o velikosti obvyklé šekové nebo kreditní karty, do které je zabudován *čip*, který je schopen ukládat data a provádět výpočty tj. čipová karta je skutečný minipočítač. Dnešní čipové karty nemají žádné baterie, takže proud musí být přiváděn během provozu z venku. To je jedna z úloh zlatých kontaktů, podle kterých lze odlišit čipovou kartu od jejích zaostalých příbuzných. Jiná jejich důležitá funkce je, že zajišťují transport dat od a do karty.

Čipová karta vznikla v roce 1974, kdy francouzský novinář Roland **Moreno** nahlásil k patentování *systém pro ukládání dat na nezávislém přenosném předmětu*. Od této doby byly vyrobeny stovky miliónů čipových karet a to hlavně ve Francii, kde se stala vzorem čipová karta CP 8 firmy Bull. Také v Německu se čipové karty masově používají (telefonní karty).

Proč se tak zajímáme v kryptologii o čipové karty? Protože s čipovou kartou máme poprvé k dispozici médium, které zajišťuje bezpečnost na základě kryptologie a to pro každého a ne pouze pro experty. To spočívá v tom, že čipová karta má dvě vlastnosti, které doposud byly od sebe odděleny:

- Čipové karty jsou *ideální pro kryptologii*: mohou provádět kryptologické algoritmy a ukládat bezpečným způsobem tajné klíče.

- Čipové karty jsou *ideální pro lidi*: dají se velmi jednoduše používat. Uvidíme, že jediná úloha uživatele je, aby si zapamatoval svoje tajné číslo, aby se mohl vůči čipové kartě identifikovat.

Shrneme-li si předchozí argumenty, máme že čipové karty se neobsluhují obtížněji než obvyklé kreditní karty; poskytují však bezpečnostní služby na nejvyšší úrovni, totiž takové, které jsou založeny na kryptologických mechanismech.

6.4.1 Čipové karty na kontrolu vstupu

Základní idea je vložit čipovou kartu mezi uživatele a počítač, aby se proces určení autenticity uživatele stal bezpečnější. Tento proces je prováděn následovně:

- Uživatel se identifikuje vůči své čipové kartě pomocí tajného čísla, které se obvykle nazývá **PIN** (osobní identifikační číslo).

- Karta se identifikuje vůči počítači pomocí dynamického protokolu pro určení autenticity.

V případě, že Mr. X si chce zajistit neoprávněný přístup k systému, musí nejen zjistit u právoplatného uživatele tajné číslo, nýbrž i získat čipovou kartu.

Uvažme nyní oba kroky k určení autenticity samostatně.

- Když uživatel zadá svůj PIN přes klávesnici terminálu, přenese se tajné číslo přímo k čipové kartě a porovná se s uvnitř uloženou referenční hodnotou. Pokud obě tyto hodnoty nesouhlasí, nedovolí karta žádnou jinou službu. Protože se porovnává pouze tajné číslo uživatele uvnitř jeho karty, není třeba žádného centrálního PIN–souboru. Můžeme si snadno představit systémy, u kterých může uživatel měnit svůj PIN, pokud chce, a u kterých může mít PIN proměnnou délku.

- Proces určení autenticity karty vůči počítači se provádí pomocí Challenge-and-Response procedury, jak je popsáno v odstavci 6.3. Proto potřebují karta a počítač společný algoritmus f a společný tajný klíč K. Počítač vyzve kartu tak, že jí zadá náhodné číslo $RAND$. Karta aplikuje algoritmus f za klíče K na náhodné číslo $RAND$ a zašle výsledek AP jako odpověď počítači. Tento pak v mezidobí rovněž spočítal AP a může se tak přesvědčit, zda oba výsledky splývají.

Přednosti této procedury leží na dlani: Čísla RAND se mění případ od případu; z tohoto důvodu nemůže Mr. X předpovědět, které bude příští číslo $RAND$. Přitom tajná data (jako např. klíče) zůstanou utajené.

Dnešní čipové karty nají standardizované rozměry $53,98 \times 85,595 \times 0,76$ mm dle normy ISO 7810. Čipové karty nesou ve svém těle vlepený nebo zalaminovaný plochý mikroelektronický **paměťový modul**, jehož poloha, význam a počet kontaktů je normován. Všeobecně i u nás rozšířenou takovou kartou je telefonní karta SPT Telecom, s.p. pro placení ve veřejných telefonních automatech. Modul totiž nesmí být větší než 20 mm^2, jinak by se totiž čip pro ohnutí karty mohl zlomit. Proto jsou na čipových kartách prakticky všude implementovány symetrické algoritmy, tj. algoritmy, u kterých obě strany vlastní tentýž tajný klíč. Samozřejmě bychom si přáli implementaci Zero-Knowledge protokolu nebo asymetrických algoritmů, což ale doposud není zcela možné.

Karty osazené paměťovým čipem jsou buď **kontaktní** nebo **bezkontaktní**. Velikost a typ jejich paměti předurčují čipové kary pro určité typy aplikací. Z hlediska použití lze rozlišit tyto základní skupiny karet:

– **paměťové karty s volným přístupem** (Free Access Cards) obvykle s 2-4 kbity paměti typu EEPROM.

– **předplatní karty** (Prepaid Cards/Token Memorz Cards) obvykle v podobě PROM nebo kombinace ROM/PROM 104-160-208 bitů, při aplikaci „abacus“ čitačů až 20 kbitů jednotek,

– **víceúčelové paměťové karty** (Multi-Service Cards, Electronic Purse) s pamětmi ROM/EPROM do 1 kbit spolu se zabezpečovací logikou jakou je transportní kód, který slouží k otevření karty v procesu jejího vydávání,PIN na kartě, který je vydavatelem karty přidělen adresně uživateli karty, který slouží pro ověření konzistence karty a jejího předkladatele ap.

Aplikace pro identifikaci uživatele

Důkazy založené na Zero-Knowledge protokolu tvoří nový revoluční způsob pro realizaci hesel. Základní myšlenka je, že každý uživatel uloží tvrzení věty ve své veřejně přístupné knihovně a přitom důkaz tvrzení zná pouze on. Po loginu uživatel provede Zero-Knowledge důkaz správnosti věty. Je-li důkaz uspokojivý, je poskytnuto povolení práce na počítači. Toto garantuje, že dokonce protivník, který odposlechne Zero-Knowledge důkaz, se nemůže naučit tolik, aby získal neautorizovaný přístup k počítači. To je nová vlastnost, které nemůže být dosaženo tradičním heslovým mechanismem.

6.4.2 Nákupy s čipovou kartou

Idea **elektronických nákupů** (electronic cash) je následující: Zákazník A jde do obchodního domu, vybere si zboží a zaplatí pomocí čipové karty.

Situace, která vznikne, je problém určení autenticity mezi třemi různými skupinami: zákazník A, obchodník O a banka B (za předpokladu identity banky obchodníka a zákazníka).

Musíme rozlisšovat mezi dvěma různými službami – určení autenticity komunikujících partnerů a určení autenticity zprávy, což je nyní něco jako elektronický šek.

Určení autenticity zákazníka A vzhledem k terminálu obchodníka O se provede pomocí obvyklého protokolu pro určení autenticity uživatele. Obvykle se požaduje oboustranné určení autenticity; také terminál obchodníka se musí vykázat vůči čipové kartě jako autentický. Čipová karta má pak možnost rozlišit manipulované terminály od opravdových. Takovýto protokol opět probíhá podle metody „Challenge-and-Response“ s tím rozdílem, že nyní posílá karta náhodné číslo, na které musí dát počítač obchodníka správnou odpověď.

Jiná věc je určení autenticity elektronického šeku. Tento dokument se podepíše způsobem MAC a pak je odeslán od čipové karty k terminálu obchodníka. Je zřejmé, že obchodník O nemůže změnit šek; z druhé strany je vhodné, aby mohl ověřit platnost šeku. Ale problém při použití symetrických algoritmů spočívá v tom, že každý, kdo dovede ověřit MAC, je schopen ho také zfalšovat (změní jednoduše dokument a určí s pomocí tajného klíče odpovídající nový MAC). Zde se nabízí jako nejjednodušší řešení asymetrická podpisovací schémata.

Když je z praktických důvodů nutné použít symetrické algoritmy, lze postupovat následovně:

Zákazník A a jeho banka B mají společný tajný klíč K^*, který obchodník O nezná. Na elektronický šek D aplikujeme obvyklou MAC-proceduru. Potom, co jsou zákazník a karta akceptováni, lze uskutečnit nákup. Za tímto účelem musí potvrdit účetní data D (obnos, druh zboží, datum, jméno obchodního domu) a předat je takovým způsobem obchodníkovi O, že

– obchodník má jistotu, že obdrží peníze od banky, a

– na obchodmích datech obchodník O nemůže provést dodatečné úpravy (podvodný obchodník by mohl po dodání zboží zvýšit dohodnnutou částku a tuto zvýšenou částku požadovat od banky).

Aby bylo možno zabránit tomuto podvodu, má čipová karta uložený tajný klíč K^*. To je klíč, který byl dohodnut mezi zákazníkem A a jeho bankou B – a tento klíč je zcela neznám každému potenciálmímu podvodníkovi.

Čipová karta neposílá pouze účetní data D na terminál obchodníka. nýbrž také s pomocí klíče K^* podepsaná data. Takto konstruovaný MAC se nyzývá **certifikát** a označuje se CER. Takovýto CER zaručuje, že obchodník O nemůže zfalšovat získaný šek. Pokud se chce přesvědčit o spolehlivosti šeku, může pověřit určením pravosti banku.

Způsob certifikování není přirozeně ideální a to ze dvou důvodů. Nejdřív musí jak zákazník A, tak obchodník O důvěřovat bance B. Dále nemůže obchodník vyměnit šek okamžitě. Dohromady však nabízí tento systém podstatně větší bezpečnost pro všechny zúčastněné než jiné systémy založené na magnetických či jiných kartách.

Kapitola 7

Asymetrické šifrovací systémy neboli systémy s veřejným klíčem

Doposud jsme pracovali se šifrovacími systémy následujících vlastností:

1. Kdo může zašifrovat zprávu, může ji i dešifrovat.
2. Každá dvojice partnerů musí mít svůj společný tajný klíč.

Druhá vlastnost je nepochybně nevýhodná. Pokud by počítačová síť měla n navzájem propojených účastníků, museli by používat $\frac{n\cdot(n+1)}{2}$ různých šifrovacích klíčů, které by si účastníci museli mezi sebou vyměnit. Zvolíme-li např. $n = 500$, bylo by nutno mít v systému cca 125000 klíčů. Vzhledem k nutné obnově za nové klíče bychom se dostali do prakticky nerealizovatelné situace.

O šifrovacím systému s první vlastností pak obvykle mluvíme jako o **symetrickém** šifrovacím systému. První vlastnost můžeme považovat za výhodnou, protože lze k šifrování a dešifrování použít stejný stroj. Asymetrické algoritmy se vyznačují tím, že jsou od první vlastnosti co nejvíce možná vzdáleny. Ukážeme, že takovéto systémy nemají druhou vlastnost a tedy práce s klíči je jednoduchá.

Budeme hlavně používat pojem **asymetrického algoritmu**; občas budeme mluvit o **public-key algoritmu**. Takovéto postupy byly vyvinuty v roce 1976 Whitfieldem **Diffiem** a Martinem **Hellmanem** v jejich práci *New Directions in Cryptography*, za kterou tito američtí matematici obdrželi v témže roce výroční cenu M.I.T.

7.1 Asymetrické šifrovací systémy

Budeme předpokládat, že každý účastník $\mathbf{T}$ má *dvojici* klíčů, a to

- **veřejný klíč** $\mathbf{E} = \mathbf{E_T}$ k zašifrování;

- **soukromý (tajný) klíč** $\mathbf{D} = \mathbf{D_T}$ k dešifrování;

které se vyznačují následující vlastností: *Ze znalosti klíče* $\mathbf{E_T}$ *nelze zjistit soukromý klíč* $\mathbf{D_T}$. Kryptosystém s touto vlastností se nazývá **asymetrický kryptosystém**.

Pokud navíc předpokládáme, že pro každou zprávu M platí

$$\mathbf{D}(\mathbf{E}(M)) = M,$$

mluvíme o **asymetrickém šifrovacím systému**. Asymetrický kryptosystém se nazývá **asymetrické podpisovací schéma**, pokud pro každou zprávu M lze pomocí veřejného klíče $\mathbf{E}$ prověřit, zda se k sobě M a $\mathbf{D}(M)$ hodí.

Všechny veřejné klíče jsou uloženy ve veřejně dostupném souboru (podobnému telefonnímu seznamu), zatímco soukromé klíče jsou tajné tj. známé pouze jejich vlastníkům.

Šifrování a dešifrování pomocí asymetrického šifrovacího systému probíhá ve 3 krocích:

1. Chce-li **A** zaslat **B** zprávu M, pak
   - najde veřejný klíč $\mathbf{E_B}$ pro **B**,
   - zašifruje zprávu M pomocí klíče $\mathbf{E_B}$ a
   - odešle $\mathbf{E_B}(M)$ k **B.**
2. **B** může kryptogram $\mathbf{E_B}(M)$ dešifrovat, protože zná jako jediný tajný klíč $\mathbf{D_B}$:
   $$\mathbf{D_B}(\mathbf{E_B}(M)) = M.$$
3. Žádný jiný účastník nemůže $(\mathbf{E_B}(M)$ rozluštit, protože podle předpokladu ze znalosti $(\mathbf{E_B}$ a $(\mathbf{E_B}(M)$ nelze získat znalost o $\mathbf{D_B}$.

*

Výhody asymetrického šifrovacího systému jsou následující:

- *Není potřeba výměna klíčů.* Tímto je vyřešen hlavní problém symetrického algoritmu. Zejména je tedy s pomocí asymetrického algoritmu možná *bezprostřední komunikace.* Můžeme tedy navzájem komunikovat, aniž bychom se složitě dohadovali o klíči. Asymetrické algoritmy jsou ideální pro *otevřenou komunikaci.*

- *Není potřeba mnoho klíčů.* U symetrického algoritmu se zvyšuje počet klíčů *kvadraticky* s počtem uživatelů, u asymetrického algoritmu je počet klíčů roven dvojnásobku počtu uživatelů.

- *Lze přijmout bez problémů nové uživatele.* Je-li přijat nový účastník do symetrického systému, musí si s ním všichni staří účastníci vyměnit klíč. U asymetrického systému není naproti tomu nutno, aby staří účastníci aktualizovali svoje data.
- *Mnoho asymetrických systémů poskytuje skvělé možnosti pro elektronický podpis.*

Nevýhody asymetrického šifrovacího systému jsou pak:

- *Doposud není znám žádný asymetrický kryptosystém, který by byl zároveň rychlý a bezpečný.* Postup, kterým se budeme hlavně zabývat, je tzv. RSA-algoritmus. V poslední době se také pracuje s algoritmy, které spočívají na „diskrétních logaritmech".
- *Asymetrické algoritmy potřebují jistý dohled nad klíči.* Může se totiž stát, že Mr. X opatří svoji schránku s falešným jménem, ke kterému se však hodí jeho klíč. Pak může zachytit všechny zprávy, které byly adresovány původnímu adresátovi.

7.2 Elektronický podpis

Další důležitou myšlenkou **Diffieho** a **Hellmana** je *elektronický* neboli *digitální podpis*. Nejdříve si ujasněme vlastnosti obvyklého podpisu rukou.

Předpokládejme, že osoba A se podepsala rukou na nějaký dokument D. Pak má tento podpis v ideálním případu následující vlastnosti:

- Pouze osoba A může vytvořit tento podpis.
- Každý jiný účastník může prověřit, že se opravdu jedná o podpis osoby A.

Diskutujme nejprve digitální podpis s použitím symetrického šifrovacího systému (např. DES). Popíšeme dva možné přístupy.

Diffie-Lamportovo schéma

Odesílat A, který si přeje podepsat n-bitovou binární zprávu

$$M = M_1 \dots M_n,$$

si předem vybere $2n$ klíčů šifrovacího systému $\langle \mathbf{M}, \mathbf{K}, \mathbf{C} \rangle$. Označme je po řadě jako

$$a_1, \dots, a_n; \qquad b_1, \dots, b_n.$$

Tyto klíče jsou tajné.

Je-li šifrovací algoritmus e, osoba A vygeneruje $4n$ *parametrů* $\{(X_i, Y_i, U_i, V_i) : 1 \leq i \leq n\}$, kde X_i, Y_i leží v definičním oboru e a

$$U_i = e(X_i, a_i) \quad \text{a} \quad V_i = e(Y_i, b_i) \quad (1 \leq i \leq n). \tag{7.1}$$

Tyto parametry jsou dopředu zaslány příjemci B. Zároveň jsou odeslány nezávislému prověřovateli (veřejný registr).

Nyní předpokládejme, že osoba A chce odeslat podepsanou n-bitovou zprávu $M = M_1 \ldots M_n$. Bude postupovat podle následující procedury. Jejím podpisem bude řetězec

$$S = S_1 \ldots S_n,$$

kde pro všechna $i, 1 \leq i \leq n$ platí

$$S_i = \begin{cases} a_i & \text{pokud} \quad M_i = 0, \\ b_i & \text{pokud} \quad M_i = 1. \end{cases}$$

Ověřovací protokol osoby B probíhá následovně: pro všechna i $(1 \leq i \leq n)$ použije osoba B bit M_i a klíč S_i, aby ověřila, že

$$\begin{cases} \text{pokud} \quad M_i = 0 & \text{pak} \quad e(X_i, S_i) = U_i, \\ \text{pokud} \quad M_i = 1 & \text{pak} \quad e(Y_i, S_i) = V_i. \end{cases}$$

Osoba B pak akceptuje podepsanou zprávu pouze za předpokladu, že ověřovací protokol je splněn pro všechna i.

Ačkoliv tento systém je jednoduchý pro použití a snadno pochopitelný, má minimálně dvě nevýhody. První je nutná předběžná komunikace s parametry. Důležitější je však zvýšení rozměru zprávy – např. v případě DESu, kdy klíče mají délku 64 bitů, by se zpráva zvětšila 64-krát.

Rabinovo pravděpodobnostní podpisovací schéma

Rabin (1978) navrhl jiný přístup. Buď e šifrovací funkce nějakého šifrovacího systému $\langle \mathbf{M}, \mathbf{K}, \mathbf{C} \rangle$. Buď dále $(K_i : 1 \leq i \leq 2r)$ posloupnost náhodně vybraných klíčů, které odesílatel A uchová v tajnosti. Příjemce B obdrží seznam $2r$ parametrů (X_i, U_i), $(1 \leq i \leq 2r)$, kde

$$e(X_i, K_i) = U_i \quad (1 \leq i \leq 2r), \tag{7.2}$$

a tyto parametry jsou uloženy na nějakém veřejně přístupném místě.

Předpokládejme nyní, že osoba A si přeje podepsat zprávu M. Jejím podpisem pak bude řetězec

$$S = S_1 S_2 \ldots S_{2r},$$

kde pro všechna i, $(1 \leq i \leq 2r)$ platí

$$S_i = e(M, K_i).$$

Osoba B pak pokračuje následovně: nejprve vybere náhodně či jinak r klíčů, které si přeje uveřejnit. Nechť to jsou klíče

$$K_{i_1}, K_{i_2}, \ldots, K_{i_r}.$$

Pak po obdržení těchto klíčů osoba B prověří, že platí

$$e(M, K_{i_j}) = S_{i_j}, \qquad e(X_{i_j}, K_{i_j}) = U_{i_j} \quad (1 \leq j \leq r).$$

Osoba B akceptuje podpis jako podpis osoby A, pokud všechny tyto rovnosti platí. Je zřejmé, že bezpečnost příjemce závisí na jeho důvěře, že jedině osoba vlastnící tajné klíče mu mohla poslat tuto podepsanou zprávu.

Předpokládejme, že si osoba A chce podrobit kritice zprávu, o které se tvrdí, že ji podepsala a kterou B zkontroloval. Protokol A je jasný; musí vytvořit před kontrolorem svých $2r$ tajných klíčů

$$S = K_1, K_2 \ldots K_{2r},$$

a veřejně prověřit rovnosti

$$e(M, K_i) = S_i \qquad e(X_i, K_i) = U_i \quad (1 \leq i \leq 2r).$$

Protokol Rabinova systému se řídí pravidlem, že kritika je oprávněná, pouze v případě, že všechna U_i a S_i s vyjímkou nejméně r S_i souhlasí. Uvažme, co může nastat ve třech možných případech:

(a) Platí méně než r kontrol S_i. V tomto případě neměla osoba B akceptovat (M, S) jako správně podepsanou zprávu.

(b) Platí právě r kontrol S_i. V tomto případě, když si osoba B vybrala r klíčů k zveřejnění, musela si vybrat právě tyto klíče. Pravděpodobnost výběru takovéto množiny je určena předpisem

$$p_r = \frac{1}{\binom{2r}{r}}$$

a $p_r \sim 10^{-10}$ pro $r = 18$.

(c) Platí $r + 1$ kontrol S_i či více. V tomto případě je příjemce v právu.

Vraťme se nyní k asymetrickému šifrování. Pak můžeme digitální podpis realizovat následujícím způsobem:

1. Chce-li osoba A **podepsat** zprávu M, tak
   - „zašifruje“ M pomocí svého tajného soukromého klíče $\mathbf{D_A}$,
   - uveřejní podepsanou zprávu $\mathbf{D_A}(M)$.

2. Každý jiný účastník může tento **podpis** $\mathbf{D_A}(M)$ zkontrolovat tím, že pomocí veřejného klíče $\mathbf{E_A}$ prověří, zda se k sobě hodí M a $\mathbf{D_A}(M)$.

Např. u RSA algoritmu se prověří zda platí

$$\mathbf{E_A}(\mathbf{D_A}(M)) = M.$$

V takovém případě je někdy i jiná možnost kontroly digitálního podpisu: podepsaná osoba A zveřejní jen $\mathbf{D_A}(M)$, ale ne zprávu M. *Pokud osoba* B *obdrží při aplikaci* $\mathbf{E_A}$ *smysluplnou zprávu, považuje to za důkaz správnosti digitálního podpisu.* Všimněme si, že při vytvoření a kontrole správnosti digitálního podpisu byly použity pouze klíče patřící odesílateli A.

Každý účastník může prověřit digitální podpis. Rozhodující rozdíl mezi digitálním a obyčejným podpisem je, že digitální podpis $\mathbf{D_A}(M)$ je neoddělitelně spjat se zprávou M. Naproti tomu je obyčejný podpis připojen na konec zprávy. Důsledkem je pak, že při změně $\mathbf{D_A}(M)$ na $(\mathbf{D_A}(M)'$ se změní i $\mathbf{E}(\mathbf{D_A}(M))$ a tedy zpráva M a $\mathbf{E}(\mathbf{D_A}(M)')$ spolu nesouhlasí.

Předpokládejme tedy, že máme šifrovací systém s veřejným klíčem s vlastností, že pro každého uživatele I šifrovací a dešifrovací funkce komutují tj. že

$$e_I(d_I(M)) = M. \tag{7.3}$$

Uvažme následující algoritmus.

1. Odesílatel A vypočte podpis S zprávy M užitím svého vlastního soukromého klíče a obdrží
$$S = d_A(M).$$
2. Užitím veřejného klíče příjemce B vypočte A kryptogram
$$C = e_B(S).$$
3. Příjemce B vypočte podpis S z kryptogramu C užitím svého vlastního soukromého klíče a obdrží
$$S = d_B(C).$$
4. Užitím veřejného klíče odesílatele A vypočte B zprávu
$$M = e_A(S).$$

Nyní je příjemce B ve velmi výhodné pozici. Vlastní dvojici (M, S). V případě sporu, potřebuje-li přesvědčit soudce, že odesílatel A opravdu odeslal zprávu, požádá A o vytvoření jejího soukromého klíče K_A. Odesílatel A musí opravdu vytvořit svůj soukromý klíč K_A, protože ho lze otestovat na identitě $e_A(d_A(M)) = M$. Soudci pak pouze stačí prověřit, že $S = d_A(M)$.

Ze stejného důvodu musí příjemce B zůstat poctivý. Předpokládejme, že změnil zprávu M na zprávu M'. Pak by ale musel být schopen změnit podpis S na S', aby platilo, že $S' = d_A(M')$. Ale to může provést pouze A.

7.3 Idea funkce s vlastností padacích dveří

Zopakujme si vlastnosti systému s veřejným klíčem z odstavce 7.1.

Všichni uživatelé systému, kteří si přejí navzájem komunikovat, používají tentýž šifrovací algoritmus e a tentýž dešifrovací algoritmus d. Každý uživatel U_i má dvojici klíčů (K_i, L_i) tak, že pro každou možnou zprávu M platí identita

$$d(e(M, K_i), L_i) = M, \tag{7.4}$$

kde K_i je zveřejněn a uložen ve veřejném souboru; L_i zůstane utajem a mluvíme o něm jako o *soukromém klíči*; K_i se nazývá *veřejný klíč*. Pokud chce jiný uživatel U_j odeslat uživateli U_i zprávu M, postupuje následovně.

(a) Uživatel U_j najde veřejný klíč K_i uživatel U_i ve veřejném souboru.

(b) Uživatel U_j odešle kryptogram

$$C = e(M, K_i)$$

k uživateli U_i veřejným kanálem.

Bezpečnost systému závisí na funkcích e a d, které mají následující vlastnosti.

Vlastnost 1 Známe-li M a K, mělo by být snadné vypočíst $C = e(M, K)$.

Vlastnost 2 Je-li dán pouze kryptogram C, není snadné výpočetně nají M.

Vlastnost 3 Je-li znám kryptogram C *a* tajný klíč L_i, je snadné určit zprávu M.

Jinak řečeno, vlastnosti 1 a 2 tvrdí, že šifrovací funkce e používající veřejný klíč by měla být jednosměrná, ale vlastnost 3 požaduje navíc existenci „veřejného klíče“. Takováto jednosměrná funkce se nazývá *funkce s vlastností padacích dveří.*

Aby byl takovýto systém prakticky použitelný, je nutné splnění následující podmínky.

Vlastnost 4 Mělo by být snadné generovat „náhodné“ dvojice veřejný/soukromý klíč (K_i, L_i).

Jinak řečeno, mělo by být dostatečně „mnoho“ dvojic (K, L) tak, že je pro Mr. X nemožné sestrojit tabulku vhodných funkčních hodnot.

7.4 RSA-algoritmus

Připomeňme dvě zásadní tvrzení z teorie čísel.

Věta 7.4.1 (Euler) *Nechť* $(c, m) = 1$. *Pak platí* $c^{\varphi(m)} = 1\, mod\, m$.

Věta 7.4.2 (Fermat) *Nechť* p *je prvočíslo* $(c, p) = 1$. *Pak platí* $c^{p-1} = 1\, mod\, p$.

Postup při šifrování RSA-algoritmu

1. Najděme dvě „velká" přirozená čísla p a q a položme $n = p \cdot q$.
2. Najděme „velké a náhodné" přirozené číslo d tak, že je nesoudělné s číslem $(p-1)\cdot(q-1)$.
3. Vypočtěme jediné přirozené číslo e ležící v oboru hodnot $1 \le e \le (p-1)\cdot(q-1)$ ze vztahu
$$e \cdot d = 1 \bmod (p-1)\cdot(q-1).$$
4. Zveřejněme veřejný klíč, který se skládá z dvojice přirozených čísel (e, n).
5. Reprezentujme zprávu M jako přirozené číslo z intervalu $\{1, \ldots, n\}$; rozdělme zprávu M do bloků, je-li příliš velká.
6. Zakódujme M do kryptogramu C dle předpisu
$$C = M^e \bmod n.$$
7. Dešifrujme pomocí soukromého klíče d a předpisu
$$D = C^d \bmod n.$$

RSA-podpisovací schéma

Označme veřejný klíč uživatele I dvojici (e_I, n_I) a soukromý klíč d_I. V praxi je n_I obvykle voleno jako číslo, které je součin dvou náhodně zvolených asi 100-místných prvočísel, $n_I = p_I \cdot q_I$. Prvočísla p_I a q_I jsou osobním tajemstvím uživatele I. Dále si uživatel I zvolí tzv. šifrovací exponent e_I tak, aby byl nesoudělný s $\varphi(n_I)$. Zejména tedy platí, že $(e_I, (p_I - 1)\cdot(q_I - 1)) = 1$. Uživatel I najde číslo d_I tak, že splňuje $e_I \cdot d_I = 1 \bmod \varphi(n_I)$. Toto číslo je výše uvedenými hodnotami jednoznačně určeno. Pak šifrovací předpis pro odesílatele A, který chce zaslat příjemci B podepsanou zprávu M následovně:

1. Očíslujme po řadě písmena latinské abecedy A=01 B=02, ..., Z=26. V praxi se používá desítkové vyjádření v ASCII kódu. Text, který chceme utajit, převedeme do číselné formy a rozdělíme na bloky stejné délky. Číselné vyjádření bloku B označíme M. Přitom požadujeme, aby $1 \leq M < n_A$.

2. Odesílatel A vypočte podpis S jako
$$S = M^{d_A} (\mathrm{mod}\, n_A).$$

3. Pak A vypočte kryptogram
$$C = S^{e_B} (\mathrm{mod}\, n_B).$$

4. Po obdržení kryptogramu C vypočte B podpis
$$S = C^{d_B} (\mathrm{mod}\, n_B).$$

5. Dále B vypočte zprávu
$$M = S^{e_A} (\mathrm{mod}\, n_A).$$

Lemma 7.4.3 *Pro všechna vhodně zvolená M platí*

$$\begin{aligned} S &= M^{d_A} (\mathrm{mod}\ n_A) \\ M &= S^{e_A} (\mathrm{mod}\ n_A). \end{aligned}$$

Důkaz. Můžeme bez újmy na obecnosti předpokládat, že $(M, n_A) > 1$. Dle předpokladu RSA-algoritmu $e_A \cdot d_A = (q_A - 1) \cdot c$ pro vhodné číslo c. Z Fermatovy věty máme
$$p_A^{e_A \cdot d_A - 1} = (p_A^{q_A - 1})^c = 1 \,\mathrm{mod}\, q_A.$$

Po vynásobení číslem p_A máme
$$p_A^{e_A \cdot d_A} = p_A \,\mathrm{mod}\, p_A \cdot q_A.$$

Je-li $(M, n_A) > 1$, je M dělitelné buď p_A nebo q_A. Nechť např. $M = a \cdot p_A$, $1 \leq a < q_A$. Pak
$$(M^{d_A})^{e_A} = M^{e_A \cdot d_A} = a^{e_A \cdot d_A} \cdot {p_A}^{e_A \cdot d_A} = a^{e_A \cdot d_A} \cdot p_A \,\mathrm{mod}\, p_A \cdot q_A.$$

Protože $(a, q_A) = 1$, máme dle Eulerovy věty
$$a^{e_A \cdot d_A} = a \ \mathrm{mod}\ q_A.$$

Po vynásobení číslem p_A máme
$$p_A \cdot a^{e_A \cdot d_A} = p_A \cdot a = M \ \mathrm{mod}\ p_A \cdot q_A.$$

Pro $M = b \cdot q_A$, $1 \leq b < p_A$ se důkaz provede analogicky.

Poznamenejme, že odhlédneme-li od nutného požadavku 7.3, musí být nutně podpis S vypočtený odesílatelem A v definičním oboru šifrovací procedury e_B. Tato poslední podmínka nemusí platit, když použitý systém je RSA; podpis S může být větší přirozené číslo, než je veřejný klíč n_B. Můžeme však zajistit platnost této podmínky tím, že přizpůsobíme velikost bloků naší zprávy tak, že výsledek padne do požadovaného definičního oboru.

Příklad. Zvolme $(e_A, n_A) = (5, 35)$, $(e_B, n_B) = (3, 15)$, $M = 3$. Pak $d_A = 5$, $d_B = 3$. Máme pak $S = 3^5 = 33 \pmod{35}$, $C = 33^3 = 12 \pmod{15}$. B vypočte podpis $S' = 12^3 = 3 \pmod{15}$ a z něho zprávu $M' = 3^5 = 33 \pmod{35}$. Protože $3 \neq 33$, není pro uživatele B splněna podmínka 7.3 a M se nám zobrazí na zcela jinou zprávu M'. Pokud by však bylo $n_A < n_B$ (zvolme např. $(e_B, n_B) = (5, 35)$, $(e_A, n_A) = (3, 15)$, $M = 3$, $d_B = 5$, $d_A = 3$), máme $S = 3^3 = 12 \pmod{15}$, $C = 12^5 = 17 \pmod{35}$, $S' = 17^5 = 12 \pmod{35}$, $M' = 12^3 = 3 \pmod{15}$.

Rivest, Shamir a Adleman (1978) navrhli mnohem elegantnější řešení:

Je zvolena mezní hodnota h pro systém s veřejným klíčem (řekněme $h \sim 10^{199}$). Každý uživatel pak má *dvě* dvojice veřejných klíčů, jednu pro zašifrování a druhou pro ověření podpisu. Označme je po řadě (e_I, n_I) a (f_I, m_I), kde I probíhá množinu uživatelů. Soukromý klíč odpovídající dvojici pro ověření podpisu budeme značit g_I. Zvolený předpis se řídí tím, že šifrovací modul n_i a podpisovací modul m_I by měly pro každého uživatele I splňovat

$$m_I < h < n_I.$$

Počítáme pak následovně:

1. Odesílatel A vypočte podpis S jako
$$S = M^{g_A} \pmod{m_A}.$$
2. Pak A vypočte kryptogram
$$C = S^{e_B} \pmod{n_B}.$$
3. Po obdržení kryptogramu C vypočte B podpis
$$S = C^{d_B} \pmod{n_B}.$$
4. Dále B vypočte zprávu
$$M = S^{f_A} \pmod{m_A}.$$

Snadno se ověří, že tento systém opravdu pracuje a aby bylo zprávy možno podepsat a ověřit všemi uživateli systému, vše co potřebujeme, aby platilo

$$0 \leq \max \mathbf{M} \leq \min\{m_I : I \in \mathbf{U}\}, \tag{7.5}$$

kde $\mathbf{U}$ je množina uživatelů systému.

7.5 Diskuse RSA-algoritmu

V uvedené verzi RSA-algoritmu vystupují veřejné parametry (e_I, n_I) a tajné parametry d_I, p_I a q_I spolu s číselným vyjádřením (části) zprávy M. Rozeberme si požadavky na jejich výběr.

- Při použití RSA-algoritmu každý účastník systému používá dvě (čtyři) 100-místná prvočísla. Kolik jich máme k dispozici? Použitím prvočíselné funkce π, která udává počet prvočísel menších než dopředu zvolené číslo n a odhaduje se pomocí odhadu
$$\pi(n) \doteq \frac{n}{\ln n}$$
získáme přibližný počet prvočísel δ ležících v intervalu $[10^{99}, 10^{100}]$
$$\delta = \pi(10^{100}) - \pi(10^{99}) \doteq 3.9 \cdot 10^{97}.$$
Pravděpodobnost, že by si dva účastníci systému vybrali tutéž dvojici 100-místných prvočísel, je pak řádově 10^{-391}.

- Dalším problémem je nalezení 100-místného náhodného prvočísla. Nejprve pomocí generátoru pseudonáhodných čísel sestrojíme 100-místné náhodné číslo m. V případě, že m bude sudé, nahradíme ho číslem $m+1$. Pak nové číslo m otestujeme některým z testů na prvočíselnost. Pokud m nebude prvočíslo, vyzkoušíme číslo $m+2$ a postup opakujeme až do té doby, než nenajdeme první prvočíslo větší než m. Lze ověřit, že počet pokusů nutných k nalezení prvočísla v okolí čísla m je logaritmickou funkcí čísla m.

Jako další uvedeme příklad jednoduchého pravděpodobnostního algoritmu na zjištění prvočíselnosti čísla m.

Algoritmus na zjištění prvočíselnosti čísla m na k pokusů

```
          BEGIN
                READ (m, k);
                FOR i := 1 TO k DO
                     BEGIN
                           a :=RANDOM(1, m − 1);
                           b := (a**(m − 1) MOD m);
                      IF b <> 1 THEN
                                 BEGIN
                                       WRITE (m, „je složené číslo“);
                                       GO TO KONEC
                                 END
                     END;
                WRITE (m, „je složené prvočíslo“);
KONEC:    END.
```

Funkce RANDOM vybírá pseudonáhodná celá čísla z určeného intervalu. Algoritmus na vstupu načte číslo m a číslo k a na výstupu obdržíme buď pravdivou odpověď, že m je složené číslo nebo odpověď, že se asi jedná o prvočíslo. V případě, že je k dostatečně velké, je pravděpodobnost, že se nejedná o prvočíslo, v případě kladné odpovědi velmi malá.

V praxi máme několik nevýhod RSA-systému:

(a) Odesílatel A může úmyslně „ztratit“ svůj soukromý klíč tak, že ,ačkoliv je uložen v „bance soukromých klíčů“ před startem systému, jí odeslané zprávy se stanou neověřitelnými.

(b) Odesílatel A může úmyslně vydat svůj soukromý klíč d_A a dovolit tak, aby všechny jí adresované zprávy byly řešitelné.

(c) Doba věnovaná šifrování, podepsání, dešifrování a prověření může být nepřiměřená. Totiž teprve nedávno byla nalezena rozumná implementace RSA-algoritmu a v současné době jsou na trhu RSA-čipy, které ale mají rychlost asi 10 Kbit/s. K dispozici jsou i speciální RSA-karty, které zvládnou 100 Kbit/s. Uvážíme-li však, že budoucí ISDN síťový standard elektronické pošty pracuje s 64Kbit/s a že se v průmyslu (lokální sítě apod.) pracuje s rychlostmi kolem 10 Mbit/s, vidíme, že nebude ještě dlouho možno použít RSA-algoritmus za účelem šifrování zpráv, nýbrž hlavně pro správu klíčů a elektronické podpisování. Při tvorbě elektronického podpisu se totiž nejdříve text zkomprimuje a podpisovací algoritmus se aplikuje na komprimát; není tedy nutno podpisovat velké soubory.

7.6 Systémy založené na ruksakové metodě

Jeden z prvních (1978) sytémů s veřejným klíčem byl vyvinut **Merklem** a **Hellmanem** a byl založen na tzv. ruksakovém problému. Přesněji, jedná se o výpočetní problém známý jako PODMNOŽINOVÝ SOUČET definovaný následovně:

Vstup: Kladná reálná čísla $a_1, a_2, \ldots, a_n, t$

Otázka: Existuje podmnožina $J \subseteq \{1, \ldots, n\}$ tak, že

$\sum_{i \in J} a_i = t$?

Tento problém je jedním z klasických NP-úplných problémů.

Zašifrování zprávy

1. Odesílaná zpráva je odeslána v binárním tvaru **m**.

2. Veřejné klíče tvoří soubor n-tic $(a_1, \ldots, a_n)$ kladných přirozených čísel.

3. Binární zpráva $\mathbf{m}$ je rozdělena do bloků a n znacích tak, že $\mathbf{m} = \mathbf{m_1} \ldots \mathbf{m_t}$, kde každé $\mathbf{m_j}$ je n-tice nul a jedniček.

4. Pro každé j, $1 \leq j \leq t$, položme

$$c_= \sum_{i=1}^{n} M_i \cdot a_i,$$

kde $\mathbf{m_j} = (M_1, \ldots, M_n)$.

5. Přeneseme posloupnost (kryptogram) $c_1, \ldots, c_t$.

Dešifrování zprávy

Zdánlivě se příjemce a každý, kdo zachytí kryptogram, zabývají tínže problémem; aby bylo možno rozluštit zprávu z posloupnosti $c_1, \ldots, c_t$ a veřejného klíče $(a_1, \ldots, a_n)$, musí vyřešit t různých NP-úplých problémů, každý pro jedno c_j.

Merkle-Hellmanův systém je založen na skutečnosti, že ne všechny případy NP-úplných problémů jsou obtížně řešitelné. Řekneme, že posloupnost $a_1, \ldots, a_n$ je *superrostoucí*, jestliže pro všechna k, $1 \leq k \leq n$, platí

$$a_{k+1} > \sum_{i=1}^{k} a_i. \tag{7.6}$$

Snadno se dokáže následující tvrzení.

Lemma 7.6.1 *Existuje rychlý algoritmus (v polynomiálním čase) pro vyřešení třídy problémů podmnožinového součtu pro superrostoucí posloupnosti.*

Proof. Předpokládejme, že posloupnost $a_1, \ldots, a_n$ je superrostoucí,. Potřebujeme reprezentovat vstup t jako součet vybrané podposloupnosti posloupnosti $a_1, \ldots, a_n$ nebo rozhodnout, že takovouto reprezentaci nelze najít.

Položme $r = \max\{i : a_i \leq t\}$. Pak $t = a_r + s$, kde nyní potřebujeme najít reprezentaci čísla s jako součet vybrané podposloupnosti posloupnosti $a_1, \ldots, a_{r-1}$ nebo rozhodnout, že takovouto reprezentaci nelze najít. Opakování tohoto postupu nám pak dá naši reprezentaci pro t nebo ojeví, že takovou reprezentaci není možno najít.

$*$

Základ Merkle-Hellmanova systému je následovný:

1. Typický uživatel A si vybere „snadnou“ superrostoucí posloupnost přirozených čísel $e_1, \ldots, e_n$.

2. Uživatel si vybere dvojici „velkých“ nesoudělných přirozených čísel w a N a transformuje pomocí ní vybranou superrostoucí posloupnost do „obtížné“ posloupnosti $T(e_1), \ldots, T(e_n)$ podle předpisu

$$T(e_i) = w \cdot e_i \ (\mathrm{mod} N).$$

Transformovaný vektor $(T(e_1), \ldots, T(e_n))$ se stane *veřejným klíčem* uživatele A. Přitom by mělo být

$$e_1 + e_2 + \cdot + e_n.$$

Lemma 7.6.2 *Buď c kryptogram odeslaný uživatel A při použití obtížného veřejného klíče $T(e_1), \ldots, T(e_n)$ uživatele A a předpisu $T(e_i) = w \cdot e_i \ (mod\, N)$. Lehký kryptogram c' pak získáme z následujícího vzorce:*

$$c' = w^{-1} \cdot c \ (\mathrm{mod}\, N).$$

Důkaz. Položme $a_i = T(e_i)$. Je-li tedy $M = (M_1 M_2 \ldots M_n)$ zpráva, pak máme

$$c = \sum_{i=1}^{n} M_i \cdot a_i.$$

Ale

$$c = \sum_{i=1}^{n} M_i \cdot a_i = \sum_{i=1}^{n} M_i \cdot w \cdot e_i + \sum_{i=1}^{n} M_i \cdot N \cdot d_i = \sum_{i=1}^{n} M_i \cdot w \cdot e_i \ (\mathrm{mod}\, N)$$

pro vhodná přirozená čísla d_i.

Máme tedy

$$w^{-1} \cdot c = \sum_{i=1}^{n} M_i \cdot e_i \ (\mathrm{mod}\, N).$$

Výhody a nevýhody

Zásadní výhodou je relativně vysoká rychlost šifrování odesílatelem. Skutečně, výpočet součtu je velmi rychlý. Viditelnou nevýhodou systému je jeho nelinearita. Skutečně, platí $E(x+y) = E(x) + E(y)$, kde $E(x) = x * a = \sum_{i=1}^{n} x_i a_i$ je operace zašifrování. Navíc v nejnižším bitu součtu $E(x)$ se operace sčítání proměňuje na operaci XOR. Proto nejnižší bit součtu $E(x)$, což je dostupný šifrový text, je roven výsledku operace XOR těch bitů otevřeného textu tj. vektoru x, které stojí u lichých a_i. Bude-li například liché jen a_1 a a_2, dostaneme x_1 or x_2=nejnižší bit $E(x)$. I když to není velká informace, ze šifrového textu by neměla „vyzařovat“. Kvalitní šifrovací systémy nevydávají o otevřeném textu vůbec žádnou využitelnou informaci.

Luštění, sázky a prohry

Merkle, jeden ze spoluautorů výše uvedeného šifrovacího systému, si byl jeho bezpečností tak jist, že na něj vsadil 100 USD. Bezpečností se pak zabývalo mnoho vědců. Herlestam učinil zkušenost, že poměrně často lze zjistit jeden bit otevřené zprávy. Shamir ukázal, že je řešitelný tzv. kompaktní problém rance. S Zippelem pak dokázali řešitelnost Merkle-Hellmanova systému, jestliže luštitel bude znát tajný modul N. Pokrok pak nastal po Lenstrově objevu řešitelnosti „problému celočíselného programování“ v polynomiálním čase. S jeho využitím pak Shamir popsal metodu řešení v polynomiálním čase a tím vyhrál Merklovu sázku. Merkle sice prohrál 100 USD, ale vsadil desetkrát tolik, že iterovaný problém nebude rozbit. E. Brickel v létě 1984 oznámil, že je schopen rozluštit čtyřicetkrát iterovaný problém rance během jedné hodiny.

7.7 Systém s veřejným klíčem se složitostí stejnou jako faktorizace

Uveďme příklad systému s veřejným klíčem, o kterém lze ukázat, že jeho složitost je ekvivalentní s problémem faktorizace. Tvůrcem systému je **Rabin** (1979). Každý uživatel systému vybere dvojici (p,q) velkých různých prvočísel, které uchová v tajnosti. Zároveň si vybere přirozené číslo $B < N = p \cdot q$.

Veřejný klíč bude dvojice (B,N), *soukromý klíč* bude faktorizace (p,q) čísla N.

Šifrovací funkce e zprávy M, kde M je reprezentovatelná jako přirozené číslo v definičním oboru $\{1,\dots,N-1\}$ (v případě potřeby se zpráva rozparceluje na více bloků), je

$$e(M) = M \cdot (M+B) \pmod N. \tag{7.7}$$

Je-li C výsledný kryptogram, pak dešifrovací problém je nalézt M tak, že

$$M^2 + B \cdot M = C \pmod N. \tag{7.8}$$

Kongruenční rovnice a mocninné zbytky

Poznamenejme nejprve, že platí následující tvrzení

Věta 7.7.1 *Kongruenční rovnice*

$$ax = b \ (mod\, m). \tag{7.9}$$

je řešitelná právě tehdy, když $(a,m)/b$. V tomto případě má rovnice právě (a,m) navzájem nekongruentních řešení modulo m.

Důkaz. Výše uvedená podmínka je nutná, neboť v opačném případě nemůže platit rovnost $ax = b + km$ v oboru celých čísel. Buď tedy $d = (a, m)$ a nechť d/b.

1. Nechť $d = 1$. Dle Bezoutovy věty existují celá čísla u, v taková, že $au + mv = 1$. Existují tedy celá čísla x, y splňující $ax + my = b$, tj. platí $ax = b \pmod m$. Řešení x je jednoznačně určeno modulo m, neboť je-li x' jiné řešení splňující $ax' = b \pmod m$, máme $a(x - x') = 0 \pmod m$ a tedy $x = x' \pmod m$.

2. Nechť $d > 1$. Protože nutně d/b, máme po dosazení do vztahu $ax = b + km$ za $a = a'd$, $b = b'd$, $m = m'd$ a po vydělení číslem d kongruenční rovnici
$$a'x = b' \pmod{m'}.$$
Z případu 1 víme, že tato kongruenční rovnice má jediné řešení $x = x_0 \pmod{m'}$. Všechna řešení modulo m tvoří právě d následujících čísel
$$x = x_0, x_0 + m', \ldots, x_0 + (d - 1)m', \pmod m.$$

Budeme chtít vyřešit resp. zjistit, zda následující kongruenční rovnice má řešení v celých číslech pro $n \geq 2$:

$$ax^n = b \pmod m. \tag{7.10}$$

Podobně jako v případě lineárních kongruečních rovnic se lze omezit na případ, kdy $(a, m) = 1$. Použitím Eulerovy věty pak obdržíme rovnici $x^n = ba^{\varphi(m)-1} \pmod m$.

Buďte tedy m, n přirozená čísla taková, že $m \geq 2$, $n \geq 2$, a celé číslo takové, že $(a, m) = 1$. Číslo a se nazývá *n-tý mocninný zbytek* modulo m, je-li řešitelná kongruence

$$x^n = a \pmod m. \tag{7.11}$$

Pro zkoumání takovýchto kongruenčních rovnic využijeme následujících tvrzení.

Věta 7.7.2 *Buďte čísla $m_1, m_2, \ldots, m_r$ navzájem nesoudělná, $a_1, a_2, \ldots, a_r$ a $b_1, b_2, \ldots, b_r$ libovolná celá čísla taková, že $(a_1, m_1) = (a_2, m_2) = \ldots = (a_r, m_r) = 1$. Pak má systém*
$$a_i x = b_i \pmod{m_i} \tag{7.12}$$
pro $1 \leq i \leq r$ právě jedno řešení modulo $m = m_1 \cdot m_2, \cdot \ldots \cdot m_r$.

Důkaz. Zřejmě mají jednotlivé kongruenční rovnice právě jedno řešení, které získáme z Euklidova algoritmu pro čísla a_i a m_i ($a_i \cdot u_i + m_i \cdot v_i = 1$ a pronásobíme-li b_i máme $a_i \cdot x_i + m_i \cdot y_i = b_i$ tj. $a_i x = b_i \pmod{m_i}$). Předpokládejme, že toto řešení je ve tvaru
$$x = c_i \pmod{m_i} \tag{7.13}$$

pro $1 \le i \le r$. Protože máme $(m_i, m_j) = 1$ pro $i \ne j$, máme $(\frac{m}{m_1}, \frac{m}{m_2}, \ldots, \frac{m}{m_r}) = 1$. Zejména tedy existují čísla $y_1, y_2, \ldots, y_r$ tak, že

$$\frac{m}{m_1} \cdot y_1 + \frac{m}{m_2} \cdot y_2 + \ldots + \frac{m}{m_r} \cdot y_r = 1.$$

Položme $e_i = \frac{m}{m_i} \cdot y_i$ pro $1 \le i \le r$. Zřejmě platí

$$e_1 + e_2 + \ldots + e_r = 1 \pmod{m}, \tag{7.14}$$

$$e_i \cdot e_j = 0 \pmod{m} \text{ pro } i \ne j, \tag{7.15}$$

$$e_i \cdot e_i = e_i \pmod{m}, \tag{7.16}$$

$$e_i = \begin{cases} 0 \pmod{m_i} & \text{pro } i \ne j, \\ 1 \pmod{m_i} & \text{pro } i = j. \end{cases} \tag{7.17}$$

Totiž $e_i \cdot e_j = m \cdot c$, $e_i \cdot e_i = \sum_i^r e_j \cdot e_i = 1 \cdot e_i = e_i \pmod{m}$, $e_j = m_i \cdot c'$, $(e_i, m_i) = 1$.

Položme

$$x_0 = c_1 e_1 + c_2 e_2 + \ldots + c_r e_r,$$

máme pak z 7.17, že

$$x_0 = c_i \pmod{m_i}$$

pro $1 \le i \le r$. Je tedy x_0 společné řešení modulo m. Pro každé jiné řešení x_0' modulo m systému 7.13 máme

$$x_0 = x_0' \pmod{m_i}$$

pro $1 \le i \le r$ a tedy také $x_0 = x_0' \pmod{m}$.

Připomeňme, že pro všechna přirozená čísla m tvoří zbytkové třídy $[a]_m$ pro $(a, m) = 1$ multiplikativní abelovskou grupu modulo m. Přitom počet prvků této grupy je právě $\varphi(m)$. Tuto grupu budeme v dalším označovat jako G_m. Věnujme se pro chvíli zkoumání její algebraické struktury. Nechť $m = m_1 m_2 \ldots m_r$, $m_1, m_2, \ldots, m_r$ jsou navzájem nesoudělná. Podle věty 7.7.2 má systém kongruencí

$$x = a_i \pmod{m_i}$$

pro $1 \le i \le r$ právě jedno řešení a modulo m. Přitom platí, že $(a, m_i) = (a_i, m_i)$ pro $1 \le i \le r$. Zejména tedy $(a, m_i) = 1$ právě tehdy, když $(a_i, m_i) = 1$. Opět podle věty 7.7.2 máme jednoznačně určený rozklad na základě rovností 7.14, 7.15, 7.16, 7.17 tvaru

$$[a]_m = [a_1 e_1]_m + \ldots + [a_r e_r]_m. \tag{7.18}$$

Označme jakožto $[a_i^*]_m$ zbytkovou třídu $[e_1 + \ldots + e_{i-1} + a_i e_i + e_{i+1} + \ldots + e_r]_m$. Pak pro pevné i tvoří množina $G_{m_i}^*$ zbytkových tříd $[a_i^*]_m$ podgrupu grupy G_m. Z rovnosti 7.18 obdržíme jednoznačně určený rozklad

$$[a]_m = [a_1^*]_m \ldots [a_r^*]_m. \tag{7.19}$$

Provedeme-li tento rozklad pro všechna $[a]_m \in G_m$, lze výše uvedené formulovat tak, že grupa G_m je přímý součin podgrup $G^*_{m_1}, \ldots, G^*_{m_r}$. Máme zejména izomorfismus mezi grupami G_{m_i} a $G^*_{m_i}$ pomocí zobrazení $[a_i^*]_m \longleftrightarrow [a_i]_{m_i}$.

Řekneme, že a patří modulo m k exponentu d, pokud $(a,m) = 1$, $a^d = 1 \pmod{m}$, ale $a^n \neq 1 \pmod{n}$ pro $1 \leq n < d$.

Lemma 7.7.3 *Patří-li a modulo m k exponentu d, jsou čísla $1, a, a^2, \ldots, a^{d-1}$ modulo m nekongruentní. Je-li dále $a^t = 1 \pmod{m}$, tedy $d|t$.*

Důkaz. Nechť $a^k = a^h \pmod{m}$, $0 \leq h < k < d$. Protože $(a,m) = 1$, je $a^{k-h} = 1 \pmod{m}$. To je však spor s $0 < k-h < d$ a minimalitou d. Položíme-li $t = dq+r$, $0 \leq r < d$, máme

$$1 = a^t = a^{dq+r} = a^r \pmod{m},$$

tj. musí platit $r = 0$.

Lemma 7.7.4 *Patří-li a modulo m k exponentu d a n je přirozené číslo s $(n,d) = 1$, patří a^n rovněž modulo m k exponentu d.*

Důkaz. Nechť a^n patří k exponentu t. Pak z 7.7.3 a $(a^n)^t = 1 \pmod{m}$, obdržíme $d|nt$. Protože $(n,d) = 1$, je nutně $d|t$ a tedy i $d \leq t$. Protože $(a^n)^d = (a^d)^n = 1 \pmod{m}$, je nutně i $t \leq d$. Celkem $t = d$.

Poznamenejme, že číslo g, které modulo m patří k exponentu $\varphi(m)$, se nazývá *primitivní kořen* modulo m. Lze dokázat, že pro každé prvočíslo p vždy existuje primitivní kořen g modulo p, tedy každé číslo od 1 do $p-1$ lze vyjádřit jakožto mocninu g. Speciálně lze ověřit, že pokud $t|\varphi(p)$, má kongruenční rovnice

$$x^t = 1 \pmod{p}, \tag{7.20}$$

právě t navzájem nekongruentních řešení.

Tvrzení 7.7.5 *K modulu m existuje buď žádný nebo $\varphi(\varphi(m))$ modulo m nekonguentních primitivních kořenů.*

Důkaz. Nechť g je primitivní kořen modulo m. Podle lemmatu 7.7.4 je rovněž g^n primitivní kořen modulo m v případě, že platí $(n, \varphi(m)) = 1$. Takovýchto čísel $n \leq \varphi(m)$ je právě $\varphi(\varphi(m))$. Máme tedy v každém případě alespoň $\varphi(\varphi(m))$ primitivních kořenů. To, že nelze nalézt žádné další primitivní kořeny, plyne z lemmatu 7.7.3. Totiž, probíhá-li ν čísla mezi 0 a $\varphi(m)-1$, probíhá pak g^ν grupu G_m. Zvolíme-li ν tak, že $(\nu, \varphi(m)) = t > 1$, pak platí

$$(g^\nu)^{\frac{\varphi(m)}{t}} = (g^{\varphi(m)})^{\frac{\nu}{t}} = 1 \pmod{m}.$$

Pak ale nemůže být g^ν primitivní kořen modulo m.

Tvrzení 7.7.6 *Buď p prvočíslo. Pak G_p je cyklická. Zejména tedy existuje primitivní kořen modulo p.*

Důkaz. Prp $p = 2$ je tvrzení věty triviální. Nechť p je v dalším liché prvočíslo. Pro $d|(p-1)$ označme $\chi(d)$ počet zbytkových tříd z G_m, které patří k exponentu d modulo p. Máme ukázat, že $\chi(p-1) > 0$. Podle tvrzení 7.7.5 je pak dokonce $\varphi(p-1) = \chi(p-1)$.

Nechť tedy existuje nějaké číslo a, které patří k exponentu d. Pak dle lemmatu 7.7.3 jsou čísla tvaru $1, a, a^2, \ldots, a^{d-1}$ navzájem nekongruentní řešení rovnice $x^d - 1 = 0 \pmod p$. Toto lze přepsát pomocí polynomiální kongruence následovně

$$x^d - 1 = (x-1)(x-a)\cdots(x-a^{d-1}) \pmod p.$$

Zároveň jsou výše uvedená čísla také všechna řešení této kongruence. Podle lemmatu 7.7.4 pak i a^k patří k exponentu d, pokud $(d,k) = 1$. To znamená, že mezi řešení přináleží $\varphi(d)$ čísel, která patří k exponentu d. Nutně pak buď $\chi(d) = 0$ nebo $\chi(d) = \varphi(d)$.

Provedeme-li výčet všech prvků z G_m podle toho, ke kterému exponentu patří, je

$$\sum_{d|(p-1)} \chi(d) = p - 1.$$

Je ale dobře známo, že

$$\sum_{d|(p-1)} \varphi(d) = p - 1.$$

Nutně pak $\chi(d) = \varphi(d)$.

Buď g primitivní kořen modulo m, $(a,m) = 1$ a μ buď jednoznačně určené číslo mezi 0 a $\varphi(m) - 1$ z kongruenční rovnice

$$g^\mu = a \pmod m.$$

Pak říkáme, že μ je *index (diskrétní logaritmus)* čísla a vzhledem k bázi g. Píšeme pak $\mu = \log_g a \bmod \varphi(m)$. Přitom platí pravidla pro logaritmování součinu, mocniny atd.

Tvrzení 7.7.7 *Pro diskrétní logaritmování platí následující zákony:*

1. $\log_g ab = \log_g a + \log_g b \pmod{\varphi(m)}$,
2. $\log_g a^n = n \log_g a \pmod{\varphi(m)}$,
3. $\log_g 1 = 0 \pmod{\varphi(m)}$,
4. $\log_g g = 1 \pmod{\varphi(m)}$,

5. $\log_g(-1) = \frac{1}{2}\varphi(m) \pmod{\varphi(m)}, m > 2.$

Důkaz. Z $a = g^{\log_g a} (\mathrm{mod}\, m)$ a $b = g^{\log_g b} (\mathrm{mod}\, m)$ obdržíme $ab = g^{\log_g a + \log_g b} (\mathrm{mod}\, m)$. Porovnáme-li toto s $ab = g^{\log_g ab} (\mathrm{mod}\, m)$, obdržíme první vlastnost. Vlastnosti 2 a 3 plynou bezprostředně z vlastnosti 1. Z $g = g^{\log_g g} (\mathrm{mod}\, m)$ obdržíme 4. Pátá vlastnost je založena na Fermat-Eulerově větě:

$$g^{\varphi(m)} - 1 = \left(g^{\frac{\varphi(m)}{2}} - 1\right)\left(g^{\frac{\varphi(m)}{2}} + 1\right) = 0 (\mathrm{mod}\, m).$$

Tvrzení 7.7.8 *Buď p liché prvočíslo tak, že číslo a není dělitelné p. Kongruenční rovnice*

$$x^n = a \pmod{p^r}$$

má právě $d = (n, p^{r-1}(p-1))$ nekongruentních řešení, pokud d dělí $\log_g a$. Jinak je tato kongruence neřešitelná.

Důkaz. Tvrzení věty se logaritmováním převede na lineární kongruenční rovnici

$$n\log_g x = \log_g a \pmod{p^{r-1}(p-1)}.$$

Zbytek plyne z tvrzení 7.7.1.

Tvrzení 7.7.9 *Buď p liché prvočíslo tak, že číslo a není dělitelné p, $d = (n, p^{r-1}(p-1))$. Kongruenční rovnice*

$$x^n = a \pmod{p^r}$$

má právě řešení právě tehdy, když platí kongruenční rovnice

$$a^{\frac{1}{d}p^{r-1}(p-1)} = 1 \pmod{p^r}. \qquad (7.21)$$

Důkaz. Buď g primitivní kořen modulo p^r. Podle věty 7.7.8 je výše uvedená kongruence řešitelná právě tehdy, když existuje číslo h tak, že $\log_g a = h \cdot d$. Pak platí $a = g^{\log_g a} = g^{h \cdot d} \pmod{p^r}$. Tedy $a^{\frac{1}{d}p^{r-1}(p-1)} = g^{h \cdot p^{r-1}(p-1)} = 1 \pmod{p^r}$. Nechť obráceně platí kongruenční rovnice 7.21. Položme $\mu = \log_g a$. Protože $a = g^{\mu} \pmod{p^r}$, máme $g^{\frac{\mu}{d} \cdot p^{r-1}(p-1)} = 1 \pmod{p^r}$. Protože g je primitivní kořen modulo p^r, je $\frac{\mu}{d}$ celé číslo, tj. d dělí μ. Tedy dle tvrzení 7.7.8 je kongruenční rovnice řešitelná.

Věta 7.7.10 *Buď $f(x)$ polynom v proměnné x s celočíselnými koeficienty. Pak počet řešení kongruenční rovnice*

$$f(x) = 0 \pmod{m}, m = \Pi_{i=1}^{r} p_i^{s_i} \qquad (7.22)$$

je číslo $N = n_1 n_2 \cdot \ldots \cdot n_r$, přičemž n_i je počet řešení rovnice

$$f(x) = 0 \pmod{p_i^{s_i}} \qquad (7.23)$$

pro $1 \le i \le r$.

Důkaz. Řešitelnost kongruenční rovnice 7.22 nastává právě tehdy, když je systém kongruečních rovnic 7.23 řešitelný. V případě řešitelnosti každé jednotlivé kongruenčí rovnice označme $x = c_i \pmod{p_i^{s_i}}$ řešení i-té kongrunční rovnice. Obdržíme pak lineární systém kongruencí, který má jednoznačně určené řešení (mod m). Probíhají-li c_i všech n_i nekongruentních řešení, získáme celkem N řešení (mod m). ▌

Důsledek 7.7.11 *Počet řešení kongruenční rovnice*

$$x^s = x\ (mod\, m), m = \Pi_{i=1}^{r} p_i^{s_i} \tag{7.24}$$

je číslo $N = n_1 n_2 \cdot \ldots \cdot n_r$*, přičemž* n_i *je počet řešení rovnice*

$$x^s = x \pmod{p_i^{s_i}} \tag{7.25}$$

pro $1 \leq i \leq r$*.*

Tvrzení 7.7.12 *Počet řešení kongruenční rovnice*

$$x^s = x\ (mod\, p), \tag{7.26}$$

kde p *je prvočíslo, je roven* $1 + (p-1, s-1)$*.*

Důkaz. Uvažme dva případy. Nechť x je číslo soudělné s p tj. $x = p \pmod{p}$ - takové x je pouze jedno (p) a je řešením. Nechť x je nesoudělné s p. Množina všech nesoudělných čísel s p tvoří cyklickou grupu stupně $p-1$ a z předchozího víme, že existuje právě $(p-1, s-1)$ prvků této grupy splňujících 7.26. ▌

Důsledek 7.7.13 *Počet řešení kongruenční rovnice*

$$x^s = x\ (mod\, m), m = \Pi_{i=1}^{r} p_i \tag{7.27}$$

je číslo $N = \Pi_{i=1}^{r}(1 + (p_i - 1, s - 1))$*.*

Kvadratické kongruence

Uvažme pro celá čísla a, b, c, m $(m > 1, a \neq 0 \pmod{m})$ kvadratickou kongruenci

$$ax^2 + b \cdot x + c = 0 \pmod{m}. \tag{7.28}$$

Vynásobením $4a$ převedeme rovnici na tvar

$$(2ax + b)^2 = b^2 - 4ac \pmod{m}. \tag{7.29}$$

To znamená, že jsme schopni plně vyřešit kvadratickou kongruenci 7.28, jestliže umíme vyřešit speciální případ

$$x^2 = a \pmod{m}. \tag{7.30}$$

Tím se celá problematika převede na problém *kvadratických zbytků*. Z důkazu věty 7.7.10 víme, že se lze dále omezit na řešení rovnice

$$x^2 = a \pmod{p^s}, \tag{7.31}$$

kde p je prvočíslo. Zároveň lze předpokládat, že $(a, p) = 1$. Totiž v případě, že p dělí a máme

1. $x = 0 \pmod{p}$, pokud je $s = 1$,
2. v případě $s > 1$ by muselo být $x = py$ tj. $py^2 = a' \pmod{p^{s-1}}, a = pa'$. Nutně pak $a' = pa''$ tj. získáme kongruenční rovnici $y^2 = a \pmod{p^{s-2}}$,.

Uvažme nyní kongruenční rovnici tvaru

$$x^2 = a \pmod{p^s}, (a, p) = 1. \tag{7.32}$$

Snadným ověřením získáme řešení pro $p = 2$.

1. $s = 1$: Existuje právě jedno řešení a to $x = 1$.
2. $s = 2$:
   (a) $a = 1 \pmod{4}$: Existují právě dvě řešení.
   (b) $a = -1 \pmod{4}$: Neexistuje žádné řešení.
3. $s \geq 3$:
   (a) $a = 1 \pmod{8}$: Existují právě čtyři řešení.
   (b) $a \neq 1 \pmod{8}$: Neexistuje žádné řešení.

Věta 7.7.14 *Je-li p liché prvočíslo, pak má kongruence 7.32 buď žádné nebo právě dvě řešení. Je-li a kvadratický zbytek modulo p, je také kvadratický zbytek modulo p^s a obráceně.*

Důkaz. Víme, že $(2, p^{s-1}(p-1) = 2$. Podle věty 7.7.8 má pak kongruenční rovnice 7.32 právě dvě řešení, pokud $\log a = 0 \pmod{2}$ a žádné řešení, pokud $\log a = 1 \pmod{2}$. Musíme ještě ukázat, že $\log a$ je nezávisle na s dělitelné 2 nebo ne. Buď g primitivní kořen modulo p^s pro libovolné $s \geq 1$ a $\mu_s = \log_g a$ vzhledem k modulu p^s. Ze vztahu

$$a = g^{\mu_s} \pmod{p^s}, a = g^{\mu_s} = g^{\mu_1} \pmod{p},$$

plyne $\mu_s = \mu_1 \pmod{p-1}$ a proto $\mu_s = \mu_1 \pmod{2}$.

Stačí se tedy zřejmě omezit na případ, že $s = 1$.

Věta 7.7.15 *Je-li p liché prvočíslo, pak máme právě tolik kvadratických zbytků jako nezbytků. Kvadratické zbytky modulo p jsou určeny $a = 1^2, 2^2, \dots, \frac{p-1}{2}^2 \bmod p$.*

Důkaz. Uvedená čísla jsou zřejmě modulo p nekongruentní. Totiž je-li $b^2 = c^2 \pmod p$, kde $1 \leq b, c \leq \frac{p-1}{2}$, máme $(b-c)(b+c) = 0 \pmod p$. Protože $1 < b+c < p$, máme $b-c = 0 \pmod p$, tj. $b = c$. Protože dále $(p-k)^2 = k^2 \pmod p$, musí být každý kvadratický zbytek kongruentní s jedním z výše uvedených čísel. Tímto je tvrzení dokázáno.

Lemma 7.7.16 *Řešení rovnice*

$$x^2 + B \cdot x = C \pmod{p \cdot q} \tag{7.33}$$

lze obdržet jako kombinaci řešení u, v rovnic

$$x^2 + B \cdot x = C \pmod p \tag{7.34}$$

$$x^2 + B \cdot x = C \pmod q \tag{7.35}$$

a přirozených čísel a, b splňujících

$$a = 1 \pmod p, \quad a = 0 \pmod q, \quad b = 0 \pmod p, \quad b = 1 \pmod q, \tag{7.36}$$

a pak

$$x = a \cdot u + b \cdot v$$

splňuje 7.33.

Důkaz. Plyne bezprostředně z předchozích tvrzení.

Buď p liché prvočíslo, p nedělí číslo a. *Legendrův symbol* $\left(\frac{a}{p}\right)$ definujeme jako

$$\left(\frac{a}{p}\right) = \begin{cases} +1 & \text{pokud je } a \text{ kvadratický zbytek modulo } p, \\ -1 & \text{pokud je } a \text{ kvadratický nezbytek modulo } p. \end{cases}$$

Mimo jiné je vhodné vytvořit pravidla pro výpočet Legendrova symbolu. Evidentní jsou následující vlastnosti

$$\begin{aligned} \left(\frac{a}{p}\right) &= \left(\frac{b}{p}\right) \quad \text{pro } b = a \pmod p \\ \left(\frac{a^2}{p}\right) &= 1 \end{aligned}$$

Z Fermat-Eulerovy věty víme, že $a^{p-1} = 1 \pmod p$ a proto platí $a^{\frac{p-1}{2}} = \pm 1 \pmod p$. Podle věty 7.7.9 je podmínka $a^{\frac{p-1}{2}} = 1 \pmod p$ dostatečná a nutná pro řešitelnost kongruenční rovnice $x^2 = a \pmod p), (a, p) = 1$. Máme tedy tzv. *Eulerovo kritérium*:

Věta 7.7.17

$$\left(\frac{a}{p}\right) = a^{\frac{p-1}{2}} \bmod p.$$

Z tohoto kritéria lze odvodit řadu důležitých faktů:

Tvrzení 7.7.18

$$\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right).$$

Důkaz. Z Eulerova kritéria máme, že

$$\left(\frac{ab}{p}\right) = (ab)^{\frac{p-1}{2}} = a^{\frac{p-1}{2}} \cdot b^{\frac{p-1}{2}} = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right) \bmod p.$$

Tvrzení 7.7.19

$$\left(\frac{-1}{p}\right) = (-1)^{\frac{p-1}{2}}, \left(\frac{2}{p}\right) = (-1)^{\frac{p^2-1}{8}}.$$

Důkaz. První vztah plyne bezprostředně z Eulerova kritéria. Abychom dokázali druhý vztah, uvažme součin

$$\prod_{k=1}^{\frac{p-1}{2}} (-1)^k k = (\frac{p-1}{2})!(-1)^{\frac{p^2-1}{8}}.$$

Je-li v součinu číslo k liché, zaměníme $(-k)$ modulo p číslem $(p-k)$ o obdržíme rovnost

$$\prod_{k=1}^{\frac{p-1}{2}} (-1)^k k = 2 \cdot 4 \cdot 6 \ldots \cdot (p-1) = (\frac{p-1}{2})!2^{\frac{p-1}{2}} \bmod p.$$

Protože ale p nedělí $(\frac{p-1}{2})!$, máme

$$(\frac{p-1}{2})!2^{\frac{p-1}{2}} =$$

Z Eulerova kritéria plyne tvrzení.

Lemma 7.7.20 *Je-li p prvočíslo tvaru $4k-1$ a d kvadratický zbytek modulo p, řešení kongruenční rovnice tvaru*

$$y^2 = d \pmod p \tag{7.37}$$

je dáno předpisem

$$y = d^k \pmod p. \tag{7.38}$$

Důkaz. Z Eulerova kritéria máme, že

$$\left(\frac{d}{p}\right) = 1 = d^{\frac{p-1}{2}} \bmod p.$$

Protože $k = \frac{1}{4}(p+1)$, máme

$$d^{\frac{1}{4}(p+1)}d^{\frac{1}{4}(p+1)} = d^{\frac{1}{2}(p+1)} = d^{\frac{1}{2}(p-1)}d = d \bmod p.$$

Zkombinujeme-li předchozí úvahy, dostaneme následují tvrzení

Tvrzení 7.7.21 *Za předpokladu, že jak p tak q jsou kongruentní s 3 modulo 4, lze dešifrovací proceduru provést v polynomiálním čase.*

Důkaz. Příjemci, který zná faktory p a q a ví, že kryptogram je kvadratický zbytek, stačí jenom aplikovat předchozí lemmata.

Rabin ve skutečnosti dokázal víc než 7.7.21. Totiž dokázal, že i v případě, že prvočísla p a q nejsou v tomto speciálním tvaru, kongruenční rovnice modulo p a modulo q lze řešit náhodným algoritmem v polynomiálním čase. Poznamenejme, že praktickou nevýhodou Rabinova schématu je, že příjemce obdrží čtyři možné zprávy, z nichž ná vybrat tu správnou. Obvykle to lze provést tím, že má nějakou dodatečnou informaci - např. že po převedení z binárního do textového tvaru je zpráva psaná v angličtině.

Mr. X však nezná faktory p a q čísla N a musí se zabývat mnohonásobně obtížnějším problémem. Že je tomu skutečně tak, plyne z níže uvedené druhé Rabinovy věty.

Věta 7.7.22 *Označme D_N množinu všech takových d, $0 \leq d < N$, že existuje řešení kongruenční rovnice*

$$y^2 = d \ (\bmod N). \tag{7.39}$$

Jestliže pro alespoň $\lceil \frac{|D_N|}{log N} \rceil$ takovýchto d jsme schopni najít takové y, pak jsme schopni najít faktor N v náhodné polynomiální době.

Lemma 7.7.23 *Jsou-li $x, y \in \mathbf{Z}_N$ celá čísla modulo N taková, že*

$$x^2 = y^2 \ (\bmod N), \qquad x \neq \pm y \ (\bmod N), \tag{7.40}$$

jsou pak $(x+y, N)$ a $(x-y, N)$ dělitelé N. Zejména pro $N = p \cdot q$, p a q prvočísla je $(x+y, N)$ prvočíselný dělitel N.

Důkaz. $x^2 = y^2 \ (\bmod N)$ implikuje $x^2 = y^2 + rN$, kde $r \in \mathbf{Z}$. Tudíž $(x-y)(x+y) = rN$.

Věta 7.7.24 *Rozšifrování Rabinova systému s veřejným klíčem je ekvivalentní nalezení efektivního algoritmu pro faktorizaci.*

Důkaz. Předpokládejme, že máme algoritmus $\mathcal{A}$,který pro dané (q, N) a pro $\lceil \frac{|D_N|}{\log N} \rceil$ takovýchto q dává na výstupu odmocninu z q modulo N. Pak můžeme faktorizovat N iterováním následujících kroků: Vyberme náhodně z ze $\mathbf{Z}_N$ tak, že $(z, N) = 1$ a vypočtěme $q = z^2 \pmod N$. Vložme na vstup algoritmu $\mathcal{A}$ dvojici (q, N). Pokud $\mathcal{A}$ má za výstup druhou odmocninu z q různou od z nebo $-z$ modulo N, pak 7.7.23 nám dává, že jsme schopni faktorizovat N. Očekávaný počet iterací algoritmu bude malý, protože je $\frac{1}{2\log N}$-ní šance na faktorizaci N v každé iteraci.

7.8 Jak se napadá RSA-algoritmus?

Pokusy o narušení RSA-algoritmu prostřednictvím faktorizace modulu $N = p \cdot q$

Faktorizace modulu N je nepoměrně obtížnější než jeho konstrukce, tj. nalezení prvočísel p a q. V době vzniku RSA-algoritmu (1978) byla 50-místná prvočísla bezpečná, což dnes už není pravda. Proto se v současné době pracuje se 100-místnými prvočísly. Přitom není vyloučené, že bude možno najít algoritmus na faktorizaci, který pracuje v polynomiálmím čase. Zároveň se nepodařilo dokázat, že by faktorizace šifrovacího modulu byla ekvivalentní s bezpečností RSA-systému. Mohla by se totiž najít metoda, jak tento systém narušit bez faktorizace N. Zatím jsou však pokusy o narušení bezpečnosti RSA-systému založeny hlavně na faktorizaci. Lze například dokázat, že výpočet dešifrovacího exponentu t je ekvivalentní faktorizaci šifrovacího modulu N.

Prvním algoritmem, který nám napadne, je tzv. *pokusné dělení* (Trial Division) čísly $2, 3, \ldots, \lceil \sqrt{N} \rceil$. Postup lze urychlit tak, že dělíme jen čísly $2, 3$ a pak čísly tvaru $6k - 1, 6k + 1$ pro $k = 1, 2, \ldots$.

Další používanou metodou je *Pollardova* $p - 1$ *a* $p + 1$ metoda. Základem metody je následující tvrzení:

Věta 7.8.1 *Buď* $n = p \cdot q$, p, q, r *prvočísla,* $r - 1/b$, r/n *a* n *nedělí* a. *Pak* $r/(n, a^b - 1)$.

Důkaz. Podle Fermatovy věty platí $a^{r-1} = 1 \bmod r$ a tedy i $a^b = 1 \bmod r$ tj. $r/(a^b - 1)$. Zároveň však r/n.

Aby se výše uvedená věta dala využít, je nutno najít vhodná čísla a, b. Nalezení a je snadné. Stačí zvolit nějaké malé prvočíslo a přesvědčit, zda je či není dělitelem n - pokud by bylo dělitelem, našli bychom faktorizaci čísla n a tím byli hotovi.

Volba b je obtížnější, je třeba ho najít postupným zkoušením. Přitom se obvykle za b volí čísla tvaru

$$b_j = nsn(1, 2, \ldots, j).$$

Tato čísla je vhodné volit z toho důvodu, že mají mnoho vlastních dělitelů a je tedy velká šance na splnění podmínky $r - 1/b$.

Algoritmus se hodí na nalezení menších prvočíselných dělitelů čísla n. Touto metodou bylo např. faktorizované číslo $2^{257} - 1$ jako součin tří různých prvočísel.

Další známou metodou je *Pollardova rho-metoda* (Monte Carlo), kterou se obvykle najdou malé prvočíselné dělitele modulu N asi po $\sqrt{p}$ cyklech programu.

Metoda začíná výběrem libovolné nelineární funkce f s celočíselnými koeficienty, nejčastěji $f(x) = x^2 + c$, $c \neq 0, -2$ a volbou počáteční hodnoty x_0, kterou lze zvolit náhodně. V dalších krocích se rekurentně počítají hodnoty posloupnosti $x_{j+1} = f(x_j) \bmod N$, $j = 0, 1, 2, \ldots$.

Pomocí pravděpodobnostních úvah lze dokázat, že výsledná posloupnost bude skoroperiodická. To znamená, že po jisté době lze očekávat výskyt dvou hodnot x_j, x_k, pro které platí

$$x_j \neq x_k \bmod N, \qquad N = p \cdot q$$

$$x_j = x_k \bmod p.$$

To ale znamená, že $(x_j - x_k, N) = p$. Hledání největšího společného dělitele lze však provést Euklidovým algoritmem s malou složitostí. Algoritmus se ještě trochu upraví: kdyby se tímto způsobem porovnávaly všechny rozdíly $x_k - x_j$ pro všechna $j < k$, počet operací by neúměrně narůstal. Proto postupujeme následovně:

předpokládejme, že máme už spočítané x_k, přičemž k je $h + 1$-bitové číslo, $2^h \leq k < 2^{h+1}$. Označme $j = 2^h - 1$. Pak najdeme $(x_k - x_j, N)$. Pokud prvočíslo p nenalezneme, postup zopakujeme pro $k + 1$. Nevýhodou takovéhoto postup je, že pravděpodobně nenajdeme „první dvojici“ x_k, x_j.

Věnujme se pro okamžik tzv. *Fermatově faktorizaci*, která je založena na tvrzení 7.7.23 tj. že $x - y$ je pravděpodobně netriviální dělitel čísla N. Je-li navíc $N = p \cdot q$, je pak i

$$N = \left(\frac{p+q}{2}\right)^2 - \left(\frac{p-q}{2}\right)^2. \tag{7.41}$$

Protože p a q jsou různá prvočísla, nemusíme se bát toho, že by N bylo úplným čtvercem nějakého prvočísla. Pokud jsou prvočísla p a q blízko, resp. rozdíl $|p-q|$ je malý, je číslo $\frac{p+q}{2}$ jen o málo větší než $\sqrt{N}$. Pak ale stačí pro $x = 1 + \lceil\sqrt{N}\rceil$, $2 + \lceil\sqrt{N}\rceil, \ldots$ počítat $x^2 - N = u$. Je-li $u = y^2$ úplný čtverec, bude $p = x - y$ hledané prvočíslo.

Další z možných metod je metoda tzv. *řetězových zlomků*. Na základě této metody lze navrhnout procesor v ceně asi 1000000 \$, který rozloží 100-místné dekadické číslo asi za 1 měsíc. Pro 200-místné dekadické číslo je ale odhad 3.8 miliónů let, což garantuje postačující bezpečnost celého systému.

Připomeňme si, že tvůrci RSA vsadili 100 USD na to, že nikdo nerozluští jejich anglický text zašifrovaný algoritmem RSA s veřejným modulem $e = 9007$ a známým modulem N, který je součinem dvou 64- a 65-ciferných prvočísel. Rivest v roce 1977 spočítal, že na roznásobení uvedeného modulu N by bylo nejlepší faktorizační metodou potřeba 40 000 000 000 000 000 let. Proto se nebáli „investovat" 100 USD do sázky. V dubnu 1994 bylo však oznámeno, že se toto 129-ciferné číslo N podařilo roznásobit. Zároveň tak byl odhalen otevřený text zašifrované zprávy, která neměla nikdy spatřit světlo světa: „The magic words are squemish ossifrage".

Poznamenejme k výše uvedenému, že v roce 1873 se zdálo nemožné faktorizovat deseticiferné číslo. V roce 1977 byla však už známá Pollardova rho metoda faktorizace. S její výkonností také Rivest počítal při odhadu uvedeného času na faktorizaci našeho čísla N. Od té doby se však objevily další metody faktorizace: ECM (elliptic curve method), QS (quadratic sieve factoring) a NFS (number field sieve factoring algorithm). Pollardova rho metoda na nalezení 64-ciferného součinitele potřebovala odhadem 4×10^{32} modulárního násobení, tj. asi 1.3×10^{16} let – Rivest uvažoval, že by jedno molekulární násobení mohlo trvat jednu nanosekundu. ECM však už dnes potřebuje už jen 5×10^{15} modulárního násobení, tj. asi 2 měsíce. Ve skutečnosti se takovéto rychlosti modulárního násobení nedosahuje a reálný odhad by byl cca 15 000 let, což je však obrovský pokrok oproti 10^{16} letům.

Na rozdíl od ECM a Pollardovy rho metody metoda QS, použitá k nalezení 64-ciferného součinitele u našeho čísla N, závisí mnohem více na přístupu do paměti než na výkonu procesoru. Na základě analýzy reálně použitých typů počítačů a jimi spotřebovaného času na faktorizaci bylo spočítáno, že k nalezení 64-ciferného součinitele se spotřebovalo celkem 4000 až 6000 tzv. MIPS roků. Přitom jeden MIPS rok je množství operací, které za jeden rok vykoná počítač s výkonem jeden milion operací za vteřinu. Je to tedy $365,25 \times 24 \times 3600 \times 1000000$ =cca. $3.16 \cot 10^{13}$ operací. Počítáme-li průměrný výkon jednoho počítače v experimentu 10 MIPS s tím, že na experimentu pracoval jen polovinu dne tj. s reálným výkonem pouze 5 MIPS, bylo k faktorizaci použito cca. 1000 let práce. Zároveň je z výše uvedeného vidět úžasný nárůst výkonnosti faktorizačních metod v posledních letech. Faktorizaci 154-místného čísla (512 bitů) pak bude trvat cca 500000 MIPS let. Tent výkon by mohli zajistit všichni účastníci sítě INTERNET. Dostáváme pak výpočetní kapacitu 20 miliónů MIPS - tj. faktorizace by proběhla během devíti dní.

Důsledky nových faktorizačních algoritmů pro bezpečnost RSA-šifrovacího systému

Nejspolehlivější cesta, jak narušit veřejnou síť využívající RSA-kryptosystém, je nalezení dešifrovacího exponentu, označme si ho třeba t. Jednou z takovýchto možností je poznání čísel $p-1$ a $q-1$, tj. faktorizace šifrového modulu $N = pq$. Slabým místem Pollardovy $p-1$ a $p+1$ metody je nalezení vhodného čísla b. Aby jsme úlohu faktorizace čísla N pro nekompetentní osobu zkomplikovali, musíme zvolit prvočísla p a q tak, aby $p-1$ $(q-1)$ mělo velký prvočíselného dělitel r a $p+1$ $(q+1)$ velký prvočíselný dělitel d. Zároveň je vhodné požadovat, aby číslo $r-1$ mělo rovněž velký prvočíselný dělitel e. Proto prvočísla splňující kongruenční rovnice

$$\begin{aligned} p &= 1 \bmod r \\ p &= d - 1 \bmod d \\ r &= 1 \bmod e. \end{aligned} \tag{7.42}$$

(kde r, d a e jsou velká náhodná prvočísla) se nazývají *silná prvočísla*.

Aby jsme se zabezpečili i proti metodám založeným na Fermatově faktorizaci, musíme zvolit silná náhodná prvočísla p a q tak, aby rozdíl $|p-q|$ byl několik řádů. Pokud budeme mít efektivní metodu na nalezení silných náhodných prvočísel, pak splnění poslední podmínky nám nezpůsobí žádné komplikace. Takováto metoda byla poprvé navržená Gordonem v roce 1985.

Další možné útoky na RSA-šifrovací systém

Ukážeme, že dva účastníci RSA-šifrovacího systému nemohou mít stejný šifrovací modul N. Uvažujme účastníky A a B, $N_A = N_B = N$. Pak musí platit

$$(s_A, \varphi(N)) = 1 \quad \text{a} \quad s_B \cdot t_B - 1 = k \cdot \varphi(N) \tag{7.43}$$

pro nějaké celé číslo k. Uvidíme, že účastník B je schopen díky společnému N číst každou zprávu určenou účastníkovi A a také podpisovat účastníka A.

Totiž účastník B je schopen použitím Euklidova algoritmu vypočíst $f = (s_B \cdot t_B - 1, s_A)$. Označme

$$n = \frac{s_B \cdot t_B - 1}{f}. \tag{7.44}$$

Pak $(n, s_A) = 1$ a $(f, \varphi(N)) = 1$, protože šifrovací exponent s_A je nesoudělný s N. Odtud pak

$$n = \frac{k}{f}\varphi(N) \tag{7.45}$$

a přitom číslo f dělí číslo k. Proto je n násobek $\varphi(N)$. Z nesoudělnosti čísel n a s_A plyne existence čísel u, v tak, že

$$u \cdot n + v \cdot s_A = 1. \tag{7.46}$$

Bez újmy na obecnosti lze předpokládat, že $v > 0$. Pak

$$v \cdot s_A = 1 - u \cdot n = 1 \bmod \varphi(N) \tag{7.47}$$

a

$$t_A \cdot s_A = 1 \bmod \varphi(N). \tag{7.48}$$

Zejména tedy

$$(v - t_A) \cdot s_A = 0 \bmod \varphi(N). \tag{7.49}$$

a

$$v = t_A \bmod \varphi(N). \tag{7.50}$$

Takovéto v je dešifrovacím exponentem zpráv, které jsou zasílány účastníkovi A. Totiž, je-li

$$y = x^{s_A} \bmod (N), \tag{7.51}$$

zpráva určená pro A, pak platí

$$y^v = x^{s_A \cdot v} = x^{s_A \cdot v + s_A \cdot l \cdot \varphi(N)} = x \bmod (N), \tag{7.52}$$

pro vhodné $l \in \mathbf{Z}$. Pokud chce B odeslat zprávu jinému účastníkovi C, stačí podepsat zprávu místo dešifrovacím exponentem účastníka A exponentem v.

Jinou z možných příčin narušení RSA-šifrovacího systému by mohla být skutečnost, že tatáž zpráva je odeslána více účastníkům šifrovacího systému tak, že je zašifrovaná různými šiframi tohoto systému. RSA-algoritmus nemůže používat stejné šifrovací exponenty.

Z důvodu jednoduchosti uvažujme tři účastníky šifrovacího systému, kteří mají šifrovací klíče (s, N_i), $i = 1, 2, 3$. Můžeme bez újmy na obecnosti předpokládat, že moduly jsou navzájem nesoudělné. Pokud by tomu tak nebylo, našli bychom faktorizaci a tím byli hotovi. Zašleme-li těmto třem účastníkům stejnou zprávu x, $x < \min N_i$, pak Mr. X, který zachytí její zašifrované varianty y_i, $i = 1, 2, 3$, může zprávu x lehce dešifrovat. Postup Mr. X bude následovný. Z kongruenčních rovnic

$$y_1 = x^s \bmod N_1 \tag{7.53}$$
$$y_2 = x^s \bmod N_2 \tag{7.54}$$
$$y_3 = x^s \bmod N_3 \tag{7.55}$$

dostaneme pro $N = N_1 N_2 N_3$

$$y_1 N_2 N_3 = x^s N_2 N_3 \bmod N \tag{7.56}$$
$$y_2 N_1 N_3 = x^s N_1 N_3 \bmod N \tag{7.57}$$
$$y_3 N_1 N_2 = x^s N_1 N_2 \bmod N. \tag{7.58}$$

Po jejich sečtení obdržíme

$$y_1 N_2 N_3 + y_2 N_1 N_3 + y_3 N_1 N_2 = x^s \cdot (N_2 N_3 + N_1 N_3 + N_1 N_2) \bmod N. \tag{7.59}$$

Je-li $s = 3$, pak $x^s < N$ a Mr. X může z poslední kongruence přímo vypočítat zprávu x - vynásobením prvkem inverzním k prvku $(N_2 N_3 + N_1 N_3 + N_1 N_2)$ a standardním odmocněním. Výsledek lze zobecnit na d účastníků šifrovacího systému.

Speciální přístupy k dešifrování RSA-šifrovacího systému

Můžeme-li počítat s jistou neopatrností účastníka sítě, zašle Mr. X účastníkovi A zprávu $x^s \cdot a^s$, kde x^s je zašifrovaná zpráva, ktero Mr. X zachytil a pozměnil ji pronásobením číslem a^s. Veřejný šifrovací klíč je dvojice (s, N). Tedy účastník A obdrží zprávu

$$y' = a^s \cdot x^s \bmod N \tag{7.60}$$

a po dešifrování $x' = (y')^t = xa \bmod N$. Pokud bude A neopatrný a umožní x' přístup k číslu x', pak můžeme jednoduše spočítat

$$x = a^{-1} \cdot x' \bmod N. \tag{7.61}$$

Další možný přístup souvisí s *protokolem* při vytváření spojení mezi účastníky, kdy se A i B navzájem identifikují. Pokud chce účastník B navázat spojení s účastníkem A, zvolí libovolnou zprávu x a vyšle pomocí dešifrovacího exponentu t_B šifru $y = x^{t_B} \bmod N_B$. Účastník A zná šifrovací exponent účastníka B a proto si může zkontrolovat $y^{s_B} = x^{t_B \cdot s_B} = x \bmod N_B$. Nyní použije účastník A svůj dešifrovací exponent t_A a výše uvedený postup zopakuje. Zřejmě je $x < \min\{N_A, N_B\}$. Jak bude probíhat vlastní útok? Pokud účastník B získá dva takovéto podpisy od účastníka A

$$y_1 = x^{t_A} \bmod N_A, \tag{7.62}$$

$$y_2 = x^{t_A} \bmod N_A, \tag{7.63}$$

je pak schopný vytvořit třetí hodnověrný podpis účastníka A bez znalosti jeho dešifrovacího exponentu t_A

$$y = (x_1 x_2)^{t_A} \bmod N_A, \tag{7.64}$$

a ten zneužít při podpisu nějaké zprávy (B zná jak x_1 tak x_2). Proto je vhodné doplnit při podpisování zprávu x nějakou aktuální redundantní a nepředvídanou informací.

Třetí z možných přístupů je následující. Nechť šifrovací modul $N = p \cdot q$ má n-bitovou reprezentaci a prvočísla p a q mají $\frac{n}{2}$-bitovou reprezentaci. Předpokládejme, že Mr. X má možnost získat nějakým způsobem $\frac{n}{2}$-bitovou informaci o modulu N. Potom ho samozřejmě může faktorizovat - přímo se zeptá na jedno z prvočísel p a q. Dá se ukázat, že šifrovací modul N je možno faktorizovat polynomiálním algoritmem, pokud má Mr. X možnost získat jen $\frac{n}{3}$ bitů.

Na poslední z přístupů k rozšifrování RSA-šifry poukázali jako první G.J. Simmons a M.J. Norris. Tento přístup využívá algebraické vlastnosti RSA-kryptosystému.

Algebraické vlastnosti RSA-kryptosystému a iterovaný útok na jeho bezpečnost

Uvažujme množinu zpráv rozšířenou o nulový prvek, tj. $M = \{0, 1, 2, \ldots, N - 1\}$. Pro každé s, $(s, \phi(N)) = 1$ je zobrazení

$$T(s, x) = x^s \bmod N \tag{7.65}$$

permutací množiny M, která nechává nulový prvek na místě. Lze tedy množinu M s násobením považovat za konečnou komutativní pologrupu s nulovým prvkem, $T(s, -) : M \to M$ je homomorfismus pologrup zachovávající nulový prvek.

Ukažme si možnost narušení systému iterovaným šifrováním. Z konečnosti množiny M víme, že existuje číslo h tak, že

$$T(s, -)^{h+1} = T(s, -) \tag{7.66}$$

a tedy pro každé zprávu x platí $T(s, x)^h = x$. Navíc lze pro konkrétní zprávu najít takové minimální h - tj. délku cyklu, který obsahuje zprávu x.

Tento postup je však prakticky realizovatelný jen v případě, když h bude relativně malé číslo, např. menší než milión. Je přitom dokázáno, že při vhodném výběru prvočísel p a q je pravděpodobnost nalezení takovéhoto malého h menší než 10^{-90}. Vzhledem k tomu, že počet elementárních částic v nám známém vesmíru je řádově 10^{80}, lze každou pravděpodobnost menší než 10^{-80} považovat za nulovou.

Popišme si nyní pologrupu M. Ta je sjednocením čtyř disjunktních grup a obsahuje přesně čtyři idempotenty - $0, 1, e_1, e_2$. Poslední dva idempotenty dostaneme jako řešení rovnic

$$a_1 \cdot p = 1 \bmod q, \qquad \text{resp.} \qquad a_2 \cdot q = 1 \bmod p, \tag{7.67}$$

kde $a_1 \cdot p = e_1$ a $a_2 \cdot q = e_2$ leží v M. Příslušné grupy jsou tedy

$$\begin{aligned} G(0) = \{0\},\ G(1) &= \{x : (x, p \cdot q) = 1\}, \\ G(e_1) &= \{x : x = p \cdot a \bmod q, a = 1, 2, \ldots, q - 1\}, \\ G(e_2) &= \{x : x = q \cdot b \bmod p, b = 1, 2, \ldots, p - 1\}. \end{aligned}$$

To znamená, že počet prvků v jednotlivých grupách je $1, (p-1)(q-1), q-1$ a $p - 1$.

Nejprve určeme počet těch zpráv $x \in M$, které zůstanou při permutaci $T(s,-) = T_s$ na místě. Ptáme se tedy na počet řešení rovnice

$$T(s,x) = x^s = x \bmod N. \tag{7.68}$$

Lze dokázat, že všechna takováto řešení tvoří podpologrupu Z_1 pologrupy M, která obsahuje přesně

$$|Z_1| = [1 + (s-1, p-1)][1 + (s-1, q-1)] \tag{7.69}$$

prvků. V případě, že zvolíme parametry s, p, q tak, že

$$s = 2r + 1,\ p = 2p_1 + 1,\ q = 2q_1 + 1, \tag{7.70}$$

kde p_1, q_1 a r jsou navzájem různá prvočísla, je $|Z_1| = 9$ a to je zřejmě nejmenší možný počet zpráv, které se nezašifrují.

Podívejme se nyní, jak to vypadá při iterovaném šifrování. Chceme vědět, kolik různých zašifrovaných zpráv můžeme tímto způsobem přečíst, zopakujeme-li šifrování $h+1$-krát. Rovnost 7.66 přepíšeme na tvar

$$x^{s^{h+1}} = x^s \bmod N,\ \text{resp. } x^{s^h} = x \bmod N. \tag{7.71}$$

Všechna takováto řešení opět tvoří podpologrupu Z_h pologrupy M, která obsahuje přesně

$$|Z_h| = [1 + (s^h - 1, p-1)][1 + (s^h - 1, q-1)] \tag{7.72}$$

prvků.

Lemma 7.8.2 *a) $Z_l \cap Z_h = Z_d$, kde $d = (l,h)$,*

b) Dělí-li číslo l číslo h, je $Z_l \subseteq Z_h$.

Důkaz. a) Nechť $x \in G(f)$, kde $f \neq 0$ je jeden z idempotentů pologrupy M. Rovnice

$$x^{s^l - 1} = f \bmod N \text{ a } x^{s^h - 1} = f \bmod N, \tag{7.73}$$

které získáme vydělením vztahu 7.71 v příslušné grupě $G(f)$, mají společné řešení právě tehdy, když $(s^l - 1, s^h - 1) = s^d - 1$, kde $d = (l,h)$.

Tvrzení b) je speciálním případem pro $l = (l,h)$.

Lemma 7.8.3 *Označme $Z_h^* = \{x : x = x^{s^h}, x^{s^l} \neq x$ pro všechna $l < h\}$, $D_h = \{d : d/h, d \neq h\}$. Pak*

a) $Z_h^* = Z_h - \bigcup\{Z_d : d/h, d \neq h\}$,

b) $|Z_h^*| = |Z_h| - [\sum\{|Z_d| : d \in D_h\} - \sum\{|Z_{d_1} \cap Z_{d_2}| : d_1, d_2 \in D_h, d_1 \neq d_2\} + \ldots + (-1)^{|D_h|}|\bigcap\{Z_d : d \in D_h,\}|.\Big]$, *c) Pokud je $h = p^{\alpha}$, p prvočíslo, je $Z_h^* = Z_h - Z_{\frac{h}{p}}$.*

Důkaz. a) Nechť $x \in Z_h^*$, pak $x \notin Z_l$ pro libovolné $l < h$, $(l,h) \neq 1$. Tedy $Z_h^* \subseteq Z_h - \bigcup\{Z_l : l < h, (l,h) \neq 1\}$ tj. $Z_h^* \subseteq Z_h - \bigcup\{Z_d : d/h, d \neq h\}$. Opačná implikace je zřejmá. Zároveň protože $Z_h \supseteq \bigcup\{Z_d : d/h, d \neq h\}$, máme $|Z_h^*| = |Z_h - \bigcup\{Z_d : d/h, d \neq h\}|$.

b) Plyne z principu exkluze a inkluze.

c) Plyne z inkluzí

$$Z_1 \subseteq Z_p \subseteq Z_{p^2} \subseteq \ldots \subseteq Z_{p^{\alpha-1}}$$

a z rovnosti $Z_{\frac{h}{p}} = \bigcup\{Z_d : d/h, d \neq h\}$.

Pologrupu M můžeme tedy napsat jako disjunktní sjednocení množin Z_h^*. Úloha navrhovatele šifrovacího systému je, aby $|Z_h^*| = 0$ pro malá h. Množinu Z_h^* můžeme pomocí zobrazení T_s popsat následovně

$$Z_h^* = \{x : T_s^h(x) = x \text{ a } T_s^l(x) = x \text{ pro } l < h\}.$$

Lze ukázat, že univerzálním dešifrovacím exponentem je $h_0 = \lambda(\lambda(N))$, kde λ je tzv. Carmichaelova funkce, která číslu $n = p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdot \cdot p_r^{\alpha_r}$ přiřadí číslo

$$\lambda(n) = \begin{cases} 1 & \text{jestliže } n = 1, \\ 2^{\alpha-2} & \text{jestliže } n = 2^{\alpha}, \alpha > 2, \\ \varphi(n) & \text{jestliže } n = 2, 4, p^{\alpha}, p \text{ – liché prvočíslo}, \\ nsn\{\lambda(p_1^{\alpha_1}), \ldots, \lambda(p_r^{\alpha_r})\} & \text{jinak.} \end{cases}$$

Zejména tedy pro každou zprávu $x \in M$ a pro každé $1 \leq s \leq N-1$, $(s,N) = 1$ platí

$$T_s^{h_0}(x) = x. \tag{7.74}$$

Zároveň lze dokázat, že tento univerzální dešifrovací exponent nelze nahradit menším číslem. Je zřejmé, že permutace T_s tvoří grupu. Lze dokázat, že počet prvků této grupy je $\varphi(\lambda(N))$. Každá z permutací T_s generuje v této grupě nějakou konečnou cyklickou podgrupu $\{T_s, T_s^2, \ldots\}$. Její exponent (exponentem grupy G nazýváme takové celé číslo h tak, že $a^h = 1$ pro všechny prvky a grupy G) musí dělit exponent h_0 celé grupy. Proto nutná podmínka neprázdnosti množiny Z_h^* je, aby h dělilo h_0.

Nyní se budeme zajímat o výběr vhodných parametrů s, p a q. Po analýze, kterou jsme provedli, lze psát

$$M = \bigcup\{Z_h^* : h \text{ dělí } h_0\}, \text{ resp. } |M| = \sum\{|Z_h^*| : h \text{ dělí } h_0\}, \tag{7.75}$$

kde některé sčítance $|Z_h^*|$ mohou být nulové. Naším úkolem je zvolit s a N tak, aby $|Z_h^*| \neq 0$ pro velká h, tj. aby pravděpodobnost úspěchu při pokusu o narušení RSA-algoritmu iterovaným útokem byla co nejmenší. Protože víme, že

$$h_0 = \lambda(\lambda(N)) = \lambda(\text{nsn}\{p-1, q-1\}),$$

takovouto nutnou podmínkou je, aby číslo h_0 mělo velké prvočíselné dělitele. Vezmeme-li do úvahy podmínku 7.70, pak dostaneme

$$h_0 = \lambda(\mathrm{nsn}\{a_1p_1, b_1q_1\}) = \lambda(p_1q_1\mathrm{nsn}\{a_1, b_1\}), \tag{7.76}$$

kde a_1, b_1 jsou libovolná náhodně zvolená malá čísla, $p-1 = a_1p_1$, $q-1 = b_1q_1$. Podle definice Carmichaelovy funkce λ je pak h_0 násobek čísla $\mathrm{nsn}\{p-1, q-1\}$. Aby toto číslo bylo co největší, můžeme zvolit $p_1 = a_2p_2 + 1$ a $q_1 = b_2q_2 + 1$, kde a_2, b_2 jsou libovolná náhodně zvolená malá čísla a p_2 a q_2 jsou přibližně 90-ti ciferná prvočísla. Pak bude $h_0 = p_2 \cdot q_2 \cdot a$, pro vhodné celé číslo a.

Dále dokážeme, že toto h_0 je pro většinu transformací T_s nejmenším exponentem v příslušné grupě.

Zabývejme se nyní exponentem zpráv $x \in M$. Zpráva x nebude z grupy $G(1) \subseteq M$ právě tehdy, když bude násobkem prvočísla p nebo q. Pravděpodobnost takovéto situace, že $(x, N) = (x, pq) \neq 1$ pro náhodně zvolené x bude

$$\frac{|G(e_1)| + |G(e_2)| + 1}{N} = \frac{p+q-1}{N} \leq \frac{1}{p} + \frac{1}{q} \leq 10^{-90}, \tag{7.77}$$

což je možno považovat za nulovou pravděpodobnost. Tento výsledek podtrhuje spolehlivost RSA-systému při vhodné volbě klíče. Tvrdí, že pro odesilatele zprávy nemá praktický význam, aby počítal číslo (x, N), které může být rovné 1, p nebo q.

Předpokládejme tedy, že $(x, N) = 1$ a prvočísla jsou vybraná výše uvedeným způsobem

$$\begin{aligned} p &= a_1 \cdot p_1 + 1, \qquad & q &= b_1 \cdot q_1 + 1 \\ p_1 &= a_2 \cdot p_2 + 1, \qquad & q_1 &= b_2 \cdot q_2 + 1 \end{aligned} \tag{7.78}$$

kde a_i, b_j jsou libovolná náhodně zvolená malá čísla, p, q, p_1, q_1, p_2, q_2 jsou náhodně zvolená prvočísla, $p_2, q_2 \geq 10^{90}$. Ukážeme, že pro většinu zpráv x je jejich exponent násobkem čísla p_1q_1.

Připomeňme si následující známou větu z obecné algebry.

Věta 7.8.4 (L. Sylow) *Nechť G je abelovská grupa, $|G| = n$. Nechť dále p^α je největší mocnina prvočísla p, která dělí n. Pak grupa G obsahuje jedinou podgrupu, která má přesně p^α prvků. Tato podgrupa se nazývá Sylowovská.*

Důsledek 7.8.5 *Nechť $|G| = p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdot \ldots \cdot p_k^{\alpha_k}$ je kanonický rozklad čísla $n = |G|$. Pak G je izomorfní s kartézským součinem všech svých Sylowovských podgrup G_{p_i}.*

Aplikujeme-li předchozí důsledek na naši situaci, dostáváme, že pro podgrupu $G(1)$ platí $|G(1)| = t_1^{\alpha_1} \cdot t_2^{\alpha_2} \cdot \ldots \cdot t_k^{\alpha_k} \cdot p_1 \cdot q_1$. Je tedy grupa $G(1)$ izomorfní s kartézským součinem grup $G' \times G_{p_1} \times G_{p_2}$, kde $G' = G_{t_1^{\alpha_1}} \times G_{t_2^{\alpha_2}} \times \ldots \times G_{t_k^{\alpha_k}}$. Počet prvků $x \in G(1)$, jejichž exponent nebude soudělný se součinem p_1q_1 je

$$|G'| \cdot (p_1 + q_1) - |G'|.$$

Proto pravděpodobnost, že náhodně zvolené $x \in G(1)$ nebude mít exponent h, který je násobek $p_1 \cdot q_1$, bude rovna

$$\frac{|G'| \cdot (p_1 + q_1) - |G'|}{|G'| \cdot p_1 \cdot q_1} = \frac{p_1 + q_1 - 1}{p_1 \cdot q_1} \leq 10^{-90}. \tag{7.79}$$

Naprostá většina prvků pologrupy M má tedy exponent, který je násobkem prvočísel p_1 a q_1, tj. $h = a \cdot p_1 \cdot q_1$.

Aplikujeme-li tedy tento postup na grupu τ různých transformací T_s pro pevně zvolený modul N, víme, že se jedná o komutativní grupu, která má přesně $\varphi(\lambda(N))$ prvků a její exponent je $h_0 = \lambda(\lambda(N))$. Zvolíme-li parametry p a q v souladu s 7.78, dostaneme

$$\begin{aligned} \varphi(\lambda(N)) &= \varphi(p_1q_1 \cdot \mathrm{nsn}\{a_1, b_1\}) = && (7.80)\\ &= (p_1 - 1) \cdot (q_1 - 1) \cdot \varphi(\mathrm{nsn}\{a_1, b_1\}) = && (7.81)\\ &= a_2 \cdot b_2 \cdot \varphi(\mathrm{nsn}\{a_1, b_1\}) \cdot p_2 \cdot q_2. && (7.82) \end{aligned}$$

Snadno lze spočítat prvky $x \in G(1)$, jejichž exponent není dělitelný součinem p_2q_2. Pak pravděpodobnost, že při iterovaném šifrování budeme úspěšní, je

$$\frac{a \cdot (p_2 + q_2) - |G'|}{|G'| \cdot p_2 \cdot q_2} = \frac{p_2 + q_2 - 1}{p_2 \cdot q_2} \leq 10^{-90}. \tag{7.83}$$

Navíc exponent h pro většinu transformací T_s je řádově 10^{180}.

Poznámka. Poznamenejme, že exituje ještě další zřejmá možnost, jak úspěšně narušit RSA-algoritmus a to za předpokladu, že známe $\varphi(N)$. Pak je snadné nají tajný dešifrovací klíč t. Ale jak je snadno vidět, znalost $\varphi(N)$ vede k faktorizaci N. Platí totiž identity

$$p + q = n - \varphi(N),\ (p - q)^2 = (p + q)^2 - 4N,\ q = \frac{1}{2}[(p + q) - (p - q)]. \tag{7.84}$$

Rozluštění RSA-kryptosystému za 100 USD

Tvůrci RSA vsadili 100 USD na to, že nikdo nerozluští anglický text a zašifrovaný RSA s veřejným exponentem $e = 9007$ a modulem $N =$ 1143816257578886766923577997614661010218

7.9 Diskrétní logaritmus

V tomto odstavci se budeme zabývat dalším příkladem toho, co lze v současnosti považovat za další jednosměrnou funkci. Definujme tedy *diskrétní logaritmus.* Uvažme n tak, že má primitivní kořen a, tj. platí $(a, n) = 1$ a pro všechna d, $1 \leq d \leq \varphi(n) - 1$ je $a^d \neq 1$. Pokud pro x, $1 \leq x \leq \varphi(n) - 1$, platí

$$y = a^x \bmod n, \tag{7.85}$$

říkáme, že x je *diskrétní logaritmus z y při základu a modulo n* a píšeme $x = \log_a y \bmod \varphi(n)$. Důležitost toho, aby a bylo primitivní kořen, spočívá v tom, že nám tím garantuje pro každé y, $1 \leq y \leq n - 1$, $(y, n) = 1$ existenci jediného takového x tj. diskrétní logaritmus je korektně definovaná funkce.

Platí pak následující tvrzení.

Tvrzení 7.9.1 *Exponenciální funkce definovaná v 7.85 je jednosměrná, tj. provedení umocňování je snadné, ale logaritmování je obtížné.*

Důkaz. Umocňování lze provést pomocí $2 \cdot \lceil \log n2 \rceil$ násobení modulo n tj. umocňování je funkce ležící v třídě operací, které lze provést v polynomiálním čase. V současné době není znám žádný algoritmus pracující v polynomiálním čase na výpočet diskrétního logaritmu.

Využití diskrétních algoritmů pro bezpečnou distribuci klíčů

Jeden za základních problémů klasické kryptografie je způsob bezpečné distribuce klíčů. Jaká je jistota, že když nemůžeme bezpečně přenášet zprávy, že klíče budou bezpečné?

Diffie a **Hellman** navrhli elegantní způsob vyřešení tohoto problému. Závisí na jednosměrné povaze problému diskrétní logaritmizace. Uvažme seznam uživatelů $(U_i : 1 \leq i \leq N)$, kteří spolu chtějí navzájem komunikovat. Buď p velké prvočíslo (o mnoho větší než N) a nechť a je primitivní kořen z p. Typický uživatel U_i si vygeneruje, nezávisle na ostatních uživatelích, pseudonáhodné číslo X_i v rozmezí od 1 do $p - 1$ a ponechá ho v tajnosti. Zároveň prohlásí za svůj *veřejný klíč* přirozené číslo

$$Y_i = a^{X_i} \bmod p. \tag{7.86}$$

Přejí-li si uživatelé U_i a U_j komunikovat soukromě, použijí za svůj klíč číslo

$$K_{ij} = a^{X_i X_j} \bmod p. \tag{7.87}$$

Uživatel U_i si vypočte K_{ij} tím, že si najde ve veřejně přístupném souboru Y_j a použije vztah

$$K_{ij} = {Y_j}^{X_i} \bmod p. \tag{7.88}$$

Podobně to provede i uživatel U_j

$$K_{ij} = Y_i^{X_j} \mod p. \tag{7.89}$$

Je-li p prvočíslo zapsané v dvojkové soustavě pomocí b bitů, je na umocnění potřeba nejvýše $2b$ násobení modulo p. Naopak však Mr. X potřebuje více než polynomiální počet operací, aby byl schopen napadnout systém. Poznamenejme, že doposud není známo, zda prolomení tohoto systému je ekvivalentní výpočtu diskrétního logaritmu.

Šifrovací systém bez klíčů

Uvažme následující šifrovací systém. Předpokládejme, že uživatel A chce poslat zprávu uživateli B a že tato zpráva je reprezentovatelná jako přirozené číslo v intervalu $\{0, 1, 2, \ldots, p-1\}$, kde p je velké prvočíslo. Uživatel A si vybere přirozené číslo a nesoudělné s $p-1$. Totéž provede uživatel B pro číslo b. Komunikace mezi A a B sestává ze tří kroků. Nejdřív A pošle B celé číslo

$$C = M^a \mod p. \tag{7.90}$$

Potom B odešle A číslo

$$D = C^b \mod p. \tag{7.91}$$

Pak A určí celé číslo a' tak, že $a \cdot a' = 1 \mod p - 1$ a zašle B číslo

$$E = D^{a'} \mod p. \tag{7.92}$$

Následovně příjemce B dešifruje zprávu podle předpisu

$$F = E^{b'} \mod p, \tag{7.93}$$

kde b' je celé číslo tak, že $b \cdot b' = 1 \mod p - 1$ tj. existuje celé číslo t splňující $b \cdot b' = 1 + t \cdot (p-1)$.

Ukažme, že $F = M$. Totiž

$$\begin{aligned} F &= E^{b'} = \left(D^{a'}\right)^{b'} = D^{a'b'} = C^{a'b'b} = \left(C^{a'}\right)^{b'b} = \left(C^{a'}\right)^{1+t(p-1)} \mod p \\ &= C^{a'} \cdot \left(C^{a'}\right)^{t(p-1)} = C^{a'} = (M^a)^{a'} = M^{aa'} = M \mod p. \end{aligned} \tag{7.94}$$

Vážnou nevýhodou tohoto systému je 3-násobná časová náročnost.

Šifrovací systém s veřejným klíčem založený na diskrétním logaritmování

Všichni uživatelé systému znají velké prvočíslo p spolu s primitivním kořenem a modulo p. Platí tedy, že $(a, p) = 1$, $a^{p-1} = 1 \mod p$ a pro všechna d, $1 \leq d \leq \varphi(p) - 1$ je $a^d \neq 1 \mod p$.

Soukromý klíč uživatele B je náhodně vybrané přirozené číslo x_B, $1 \leq x_B \leq p - 1$.

Veřejný klíč uživatele B je číslo $y_B = a^{x_B} \mod p$.

Předpokládejme, že A chce odeslat B zprávu M, $1 \leq M \leq p-1$. A postupuje následovně:

1. Vybere náhodně číslo k tak, že $1 \le k \le p-1$.
2. Vypočte „klíč“ $K = {y_B}^k \bmod p$.
3. Zašifruje zprávu M jako dvojici přirozených čísel (C_1, C_2), kde

$$C_1 = a^k \bmod p, \qquad C_2 = KM \bmod p.$$

   Je tedy enkrypce určena zobrazením

$$e : M \mapsto C = (C_1, C_2),$$

   které nám zdvojnásobí délku zprávy.

Přitom toto zašifrování má tu vlastnost, že stejná zpráva šifrovaná podruhé nemusí dát tentýž kryptogram vzhledem k náhodnosti k.

Dešifrování provádí uživatel B následovně:

1. Vypočte K jako

$$K = {y_B}^k = a^{x_B k} = \left(a^k\right)^{x_B} = {C_1}^{x_B} \bmod p.$$

2. Vypočte M tím, že podělí C_2 číslem K modulo p.

Zřejmě je tento algoritmus snadno napadnutelný každým, kdo je schopen provést diskrétní logaritmování v dostatečně krátké době. Doposud však není známo, zda je úspěšné napadnutí tohoto systému ekvivalentní diskrétnímu logaritmování.

7.10 Rychlejší podpisy ale méně bezpečnosti

Pokud se nás příliš nedotýká bezpečnost systému, pak může být šifrovací procedura popsaná v předchozím paragrafu modifikovaná tak, že umožní výpočet rychlejší téměř o polovici. Uvažme následující algoritmus.

1. Odesílatel A vypočte podpis S zprávy M užitím svého vlastního soukromého klíče a obdrží

$$S = d_A(M).$$

2. A odešle podpis S otevřeným kanálem příjemci B.
3. Užitím veřejného klíče odesílatele A vypočte B zprávu

$$M = e_A(S).$$

B je pak ve stejné pozici jako předtím, když ověřuje, že A skutečně odeslal zprávu M. Ale, pokud Mr. X zachytil přenášený podpis S, pak může Mr. X vypočítat zprávu M stejně jako B. Tedy tento algoritmus nemá skutečné utajení. Je však mnohem rychlejší a jeho rychlost ho dělá cenným pro ty aplikace, kde je podstatným faktorem autenticita a ne utajení. Pokud jsem připraveni zcela zapomenout na utajení, pak zřejmý protokol pro odesílatele A na odeslání podepsané zprávy pro B je poslat otevřeným kanálem podepsanou zprávu (M, S), kde

$$S = d_A(M).$$

Uvažme následující příklad. *Rabinovo podpisovací schéma* je odvozeno ze systému s veřejným klíčem, kde šifrovací funkce má tvar

$$e(x) = x \cdot (x + B) \mod N, \tag{7.95}$$

kde N je součin dvou prvočísel (ta jsou utajena) a (N, B) je veřejný klíč typického uživatele. Bez újmy na obecnosti lze předpokládat, že $B = 0$, a tedy

$$e(x) = x \cdot x \mod N, \quad d(x) = \sqrt{x} \mod N. \tag{7.96}$$

Tedy uživatel A podepíše zprávu podpisem

$$S = \sqrt{M} \mod N_A; \tag{7.97}$$

přitom zde použije své znalosti faktorizace N_A. Příjemce podepsané zprávy $(M, \sqrt{M})$ musí pouze ověřit autenticitu zprávy kontrolou, že

$$M = (\sqrt{M})^2 \mod N_A. \tag{7.98}$$

Je okamžitě vidět, že toto schéma není zcela dokonalé - ne všechna přirozená čísla menší než N_A jsou čtverci modulo N_A. Aby to šlo napravit, Rabin zavedl následující postup:

Chce-li A podepsat zprávu M, přidá k ní binární řetězec U dohodnuté délky k. Výběr U je náhodný. Uživatel A pak zajistí kompresí nebo jinou vhodnou funkcí c, že

$$c(MU) \leq N_A. \tag{7.99}$$

Funkce c je veřejně známá a proto veřejně zkontrolovatelná.

Uživatel A nyní použije svůj soukromý klíč, aby zjistila, zda toto přirozené číslo $c(MU)$ je čtverec modulo N_A. Pokud tomu tak není, vybere opět náhodně jiný binární řetězec délky k a postup opakuje do té doby, než najde číslo $c(MU)$, které je čtverec modulo N_A. Lze dokázat, že průměrný počet pokusů pro nalezení takovéhoto U je malý.

Uživatel A nyní použije jako svůj podpis zprávy M dvojici (U, S). Kdokoliv si přeje ověřit, že A skutečně zprávu M odeslala, musí pouze provést

- výpočet $c(MU) = M_1$,
- kontrolu, že $M_1 = S^2 \mod N_A$.

7.11 Kódy opravující chyby jakožto systém s veřejným klíčem

Edgar Allan Poe, který se považoval za dobrého kryptoanalytika, se jednou vyjádřil v novinách, že je schopen bezprostředně rozluštit každý kryptogram (tím mínil monoabecední substituční šifru). G.W. Kulp tuto výzvu přijal a zaslal mu 43-slovní šifrový text. V dalším článku Poe dokázal, že tento šifrogram byl shlukem náhodných charakterů bez jakéhokoliv významu. V roce 1975 byl Kulpův kryptogram rozluštěn B. J. Winskelem a M. Lysterem. Ke Kulpově hlavní chybě se navíc přidalo alespoň 15 menších (snad tiskových) chyb - z toho pak vyplynuly důvody pro obtížnost dešifrování tohoto kryptogramu.

Ať je to úmyslné nebo nikoliv, zavedení chyn zcela jistě zmate nepřátelského kryptoanalytika a toto je pak základem chytrého návrhu systému s veřejným klíčem od R. J. McEliece z roku 1978. Tento systém pracuje následovně:

Typický uživatel, řekněme A, má za svůj veřejný klíč binární matici $\hat{G}$ typu $k \times n$; zasílá-li uživatel B binární zprávu M uživateli A, použije následující kódovací algoritmus.

1. B rozdělí zprávu do bloků velikosti k, a pokračuje s každým blokem zvlášť.

2. B zakóduje každý blok $\mathbf{m}$ pomocí předpisu

$$\mathbf{c} = \mathbf{e}(\mathbf{m}) = \mathbf{m}\hat{G} + \mathbf{z}, \tag{7.100}$$

   zde $\mathbf{z}$ je náhodný vektor chyb délky n a váhy menší nebo roven t, přirozené číslo bylo t předtím sjednano dopředu.

Soukromý klíč uživatele A je dvojice matic (S, P), kde S je regulární matice typu $k \times k$ a P je náhodná permutační matice. Platí pak

$$\hat{G} = SGP, \tag{7.101}$$

kde $\mathbf{G}$ je matice lineárního kódu typu $k \times n$, který je schopen odhalit a opravit až t chyb. Dešifrovací proces uživatele A je následující:

1. $\hat{\mathbf{c}} = \mathbf{c}\mathbf{P}^{-1}$=$(\mathbf{mSGP} + \mathbf{z})\mathbf{P}^{-1} = \mathbf{mSG} + \mathbf{z}\mathbf{P}^{-1}$.

2. Protože $\mathbf{z}$ je náhodný vektor s váhou menší nebo rovnu t a kód generovaný maticí G opraví t chyb, dekóduje pak dekódovací algoritmus $\hat{\mathbf{c}}$ do $\mathbf{mS}$. Přitom $\mathbf{r} = \mathbf{z}\mathbf{P}^{-1}$ je tzv. *chybový vektor*. Označíme-li H tzv. *kontrolní matici* splňující $G \cdot H^\top = 0$, je obraz chybového vektoru $\mathbf{r}$ tzv. *syndromový vektor* $\mathbf{s}$.

3. Určíme $\mathbf{m}$ pomocí násobení S^{-1} zprava.

McEliece navrhoval použít Goppovy kódy jako základ. Jedná se o třídu dobrých lineárních kódů s velmi efektivním dekódovacím algoritmem. Jejich parametry jsou tvaru

$$n = 2^a, \quad k = n - at, \quad d \geq 2t + 1,$$

pro daná čísla t a a. Je-li tedy $n = 1024 = 2^{10}$ a $t = 50$, tj, $k = 524$, nepřítel se musí zabývat luštěním kryptogramu, kde se v každé části délky 1024 vyskytuje až 50 náhodných chyb.

7.12 Digitální peníze

Dnešní Internet je obchodem, který je otevřen 24 hodin denně a sedm dní v týdnu a očekává se, že letos jej navštíví přes 100 milionů lidí. Řada světových firem dělá vše pro to, aby dosud převážně pasivní návštěvníci začali na Internetu také nakupovat a zejména platit.

Digitální peníze jsou technologií, která, jednoduše řečeno, umožňuje zaplatit číslem - na Internetu i v obchodě (např. platební kartou v supermarketu). Digitální peníze navíc nejsou ničím jiným než binárními řetězci, tj. čísly, a placení čísly se zdá být dost podivné a ne příliš bezpečné. Jak třeba poznáme, že 458967154 znamená 100 Kč nebo 90 Kč, nebo že je to vymyšlené číslo? Na digitální peníze jsou však kladeny ještě další nároky.

Požadavky na digitální peníze

Většina lidí u plateb na Internetu vyžaduje na prvním místě bezpečnost. Vždyť jde o peníze zákazníků, prodejců, poskytovatelů finančních služeb a komerčních bank. Protože emise digitálních peněz snižuje potřebu běžné hotovosti na trhu, do hry může vstoupit i centrální banka. Nejdiskutovanější problém digitálních peněz je problém dvojí útraty. Přestože čísla, na rozdíl od klasických bankovek, umí okopírovat i dítě, digitální peníze by vícenásobnou platbu neměly umožnit. Digitální peníze by také měly poskytnout podobnou míru anonymity a soukromí zákazníkům i obchodníkům jako klasické peníze. Banka by například z odevzdávané tržby neměla být schopna zjistit, kdo a za co tyto peníze v příslušném obchodě utratil. Vlastník nebo příjemce digitálních peněz by měl mít možnost zkontrolovat jejich platnost. Měly by dále umožnit platby off-line, tj. bez on-line účasti banky nebo nějakého autorizačního centra, a neměly by záviset na fyzické formě nosiče a na komunikačním prostředku, jímž jsou přenášeny. Digitální bankovky a mince vyšších hodnot by také měly být dělitelné a směnitelné za menší a naopak (až na úroveň nejnižší možné částky, např. haléře). Zdánlivě nás nemusí zajímat, kolik stojí tisk, počítání, distribuce a ochrana klasických peněz, protože je to starost bank nebo provozovatelů různých služeb. Ve skutečnosti to ale zaplatíme my, buď na daních (u výdajů státní banky), nebo na poplatcích (u komerčních bank). Cena digitálních peněz je proto velkým stimulem jejich zavedení v řadě systémů a strategickou konkurenční výhodou. Čím bude menší, tím větší bude profit všech zainteresovaných. Digitální peníze by měly být přenositelné z osoby na osobu, bez nutnosti je ihned po platbě odevzdávat bance. Z jednoduchosti tvorby a kopírování digitálních řetězců v libovolném množství (na rozdíl od tisku nebo kopírování klasických peněz) vyplývá požadavek nepadělatelnosti digitálních peněz. Digitální peníze (čísla reprezentující hodnotu) by tedy měl umět vytvářet jen k tomu oprávněný subjekt.

Základní myšlenka digitálních peněz

Existující projekty digitálních peněz nesplňují všechny zmíněné požadavky, ale už se jejich prostřednictvím platí a mají před sebou, spolu s dalšími elektronickými platebními protokoly, velkou budoucnost. Za digitálními penězi stojí vysoká matematika a minimálně čtyři revoluční objevy posledních dvaceti let. Jsou to objevy asymetrické kryptografie, konkrétního systému RSA, digitálních podpisů (digital signature) a slepých digitálních podpisů (blind digital signature).

Víme, že u RSA máme veřejné číslo N (modul šifrování), veřejný exponent E (pro zašifrování) a tajný exponent D (pro dešifrování). Veřejný exponent E a modul šifrování N se může zveřejnit, zatímco tajný exponent D zná např. jen banka emitující digitální bankovku. Banka si nejprve zvolí číslo bankovky C a poté vypočte její digitální podpis $p(C) = (CD) \bmod N$, který připojí za číslo bankovky. Kdokoliv pak může snadno zkontrolovat, že binární řetězec $(C, p(C))$ je "číslo" vydané bankou. Každý může umocnit podpis $p(C)$ na veřejný exponent E, a je-li vše v pořádku, musí (podle Eulerovy věty) vyjít podepisované číslo: $(p(C)E) \bmod N = CDE \bmod N = C$. Platnost bankovky "$C, p(C)$" tedy může být kýmkoliv zkontrolována. Připomeňme, že čísla E a D mají takovou vlastnost, že pro libovolné X platí $XDE \bmod N = X \bmod N$.

Bance vydávající digitální peníze zbývá už jen zveřejnit různé exponenty $E(i)$ pro různé nominální hodnoty (i) emitovaných "bankovek". Například $E(1000)$ pro tisícikorunové bankovky, $E(0.01)$ pro haléřové bankovky apod. U systému DigiCash jsou například nominální hodnoty voleny ve dvojkové soustavě, tj. 0.01, 0.02, 0.04, 0.08, … 10.24 atd. Klient mající zájem o digitální peníze si pak dá ze svého konta strhnout například 1000 Kč a za ně obdrží příslušný počet digitálních bankovek žádaných nominálních hodnot. Jimi (ve skutečnosti čísly), je-li na to obchodník na Internetu připraven, může pak elegantně zaplatit. Podobně je tomu i v obchodě, kde místo domácího pécéčka za nás platbu obstará platební čipová karta.

Problém sledovatelnosti

Už i u nás můžeme být sledováni podle plateb, které provádíme různými platebními prostředky, i podle dalších dat, která jsou o nás veřejně dostupná.

Všechny naše platby současnými platebními kartami (větší nákupy, čerpání pohonných hmot), rodná čísla, bydliště a spoustu jiných údajů je možné dát pěkně dohromady, aby se z nich složil náš profil. Profil budoucího potenciálního zákazníka firmy XY. Prodej databází potenciálních klientů je na Západě běžnou věcí a podobné případy známe i u nás. Na Západě však sběr takových informací, ve velké šíři a dokonalosti, existuje jako obchod mnohem déle. Je to zásah nejen do soukromí a nemusí jít jen o zákazníky. Může jít i o hledání potenciálních cílů pro kriminální účely (databáze psychicky labilních jedinců, drogově závislých, majetných, vlastníků konkrétních typů aut a jejich technického stavu).

U digitálních peněz je proto otázka anonymity často kladena na přední místo. Výše uvedené řešení je ale problematické v tom, že když se bankovky ihned vracejí bance zpět, banka ví, kde klient nakupoval.

Problém dvojí útraty

Uvedená informace se začíná zeslabovat, pokud digitální bankovka obíhá mezi lidmi delší dobu. V tom případě zase ale zesiluje nebezpečí jejího padělání. Některý z klientů si může pořídit N jejích kopií (pokud je uložena na pevném disku jeho PC, ani to nedá tolik práce) a kopiemi zaplatí N různým obchodníkům. Tyto falešné bankovky určitou dobu obíhají na trhu, než se dostanou do vydávající banky k zúčtování. Ten, kdo bankovku (či falešnou kopii) vrátí první, bude mít štěstí. Banka mu ji připíše na konto a zařadí ji na seznam už utracených bankovek. Další obchodníci, kteří vrátí ostatní falešné nebo i pravou bankovku bance, už nedostanou nic. I když je toto riziko reálné, zdá se, že existují systémy, které ho připouštějí, a že budou uvedeny ve značně širokém měřítku do praxe. Jak tento problém řešit? Jsou minimálně dvě možná řešení - složitější platební protokol nebo uložení digitálních peněz v hardwaru odolném proti narušení. Je však nutné vzít v úvahu, že útočníci mohou k vylákání čísel digitálních bankovek použít falešný terminál, např. ukradený a upravený, samostatně zkonstruovaný nebo řádný terminál s předsunutím falešné hardwarové části ap.

Slepé podpisy

Druhý z hlavních problémů digitálních peněz - riziko sledovatelnosti - umožňuje snížit na minimum vynález slepých podpisů (David Chaum, 1985). Podstata spočívá v tom, že číslo (veřejné) digitální bankovky si volí klient a že z tohoto čísla a jeho digitálního podpisu zkonstruuje nové číslo (anonymní digitální bankovky) a jeho platný digitální podpis. Banka netuší, jaké číslo anonymní bankovky podepsala, ale klientovi odepíše z účtu správnou sumu (např. 100 Kč) za jím přetvořenou bankovku. Banka přes kopírovací papír podepíše (orazítkuje) něco, co nevidí. Matematicky funguje tento postup tak, že klient si zvolí číslo cílové anonymní bankovky jako C a náhodný „zaslepující“ koeficient R. Protože zná veřejný klíč banky E, dá jí podepsat hodnotu $(C*(RE))$ mod N. Banka vytvoří digitální podpis této „zaslepené“ bankovky běžným způsobem, tj. $p = ((C*(RE))D)$ mod N, což se rovná $CD * R$ mod N. Klient ale umí tuto hodnotu vydělit jemu známým koeficientem R a „za zády banky“ obdrží hodnotu CD mod N. To je ale digitální podpis čísla C. Banka číslo C nezná, ale je schopna ověřit, že je to číslo její stokorunové bankovky. Pravost ověří běžným způsobem - umocněním podpisu CD mod N na veřejný bankovní exponent E, čímž skutečně obdrží číslo C. Nyní už tedy klient může utratit bankovku C s digitálním podpisem CD mod$N$. Pokud si v klidu projdete mechanismus, jak se klient dostal „pod kůži“ bance, uvidíte, že tomu tak je proto, že systém RSA je multiplikativní - jinak řečeno: „podpis součinu“ se rovná „součinu podpisů“.

Zopakujme si ještě jednou přesný postup

1. Klient (PC, čipová karta) generuje anonymní (A) a veřejné (V) číslo téže bankovky, anonymní překryje kopírovacím papírem a veřejnou bankovkou, obojí předá bance k podpisu.

2. Banka digitálně podepíše horní a přes kopírovací papír i dolní bankovku a odešle je zpět klientovi (zároveň mu z konta odešle 100 Kč).

3. Klient si ponechá *podepsanou* anonymní bankovku a veřejnou vrátí bance.

Rizika

Určitě vás napadla i další možná rizika digitálních peněz. Například u RSA by se pomocí právě uvedeného multiplikativního pravidla dala vytvořit nová bankovka ze dvou pravých! Budeme-li mít bankovky $C1$ a $C2$ s příslušnými digitálními podpisy $p1$, $p2$, můžeme skutečně vytvořit platnou, ale falešnou bankovku $(C3, p3)$ tak, že $C3 = C1 * C2$ a $p3 = p1 * p2$. Platební protokoly musí však pamatovat nejen na tato triviální pravidla, ale i na řadu dalších. Vzhledem k tomu, že oblast digitálních peněz je mladá, objevuje se mnoho nových nástrojů, myšlenek, platebních protokolů, ale i řada nepodložených tvrzení o jejich vlastnostech a bezpečnosti. To je ale v překotném vývoji běžné a soutěžení může přinést jedině prospěch.

7.13 Náročnější protokoly

7.13.1 Bezpečné volby

Volení pomocí počítačů se nebude používat pro obecné volby, dokud nebude existovat protokol, krerý by jak zabezpečoval soukromí jednotlivých voličů, tak zabraňoval podvodům. Ideální protokol by měl splňovat přinejmenším těchto šest požadavků:

1. Mohou volit pouze autorizovaní voliči.
2. Nikdo nemůže volit více než jednou.
3. Nikdo nemůže být schopen určit, koho volil někdo jiný.
4. Nikdo nemůže duplikovat hlas někoho jiného. (Ukazuje se, že toto je nejsilnější požadavek.)
5. Nikdo nemůže změnit hlas někoho jiného, aniž by nebyl odhalen.
6. Každý volič si může být jist tím, že jeho hlas byl započten do konečného hodnocení.

   Navíc mohou mít některá volební schémata ještě následující požadavek:

7. Každý ví, kdo volil a kdo ne.

Před tím, než popíšeme komplikované volební protokoly, podívejme se na některé jednodušší.

Jednoduchý volební protokol č. 1

1. Každý volič zašifruje svůj hlas veřejným klíčem CTF (Central Tabulation Facility - ústřední vyhodnocovací komise).
2. Každý volič zašle svůj hlas k rukám CTF.
3. CTF dešifruje hlasy, vyhodnotí je a výsledky zveřejní.

Tento protokol s sebou nese několik problémů. CTF neví, odkud pocházejí jednotlivé hlasy, takže ani neví, zda pocházejí od právoplatných voličů. Neví, zda tito voliči nevolili více než jednou. Kladnou stránkou protokolu je to, že nikdo nemůže změnit hlas někoho jiného; ale nikdo by se ani neobtěžoval pokoušet se změnit hlas někoho jiného, když je daleko jednodušší volit opakovaně ve prospěch svého výběru.

Jednoduchý volební protokol č. 2

1. Každý volič podepíše svůj hlas svým soukromým klíčem.
2. Každý volič zašifruje svůj podepsaný hlas veřejným klíčem CTF.
3. Každý volič pošle svůj hlas CTF.
4. CTF dešifruje hlasy, prověří podpisy, vyhodnotí hlasy a zveřejní výsledky.

Tento protokol splňuje vlastnosti jedna a dvě: pouze autorizovaní voliči mohou volit a nikdo nemůže volit více než jednou - CTF by zaznamenala hlasy přijaté v kroku 3. Každý hlas je podepsán soukromým klíčem voliče, takže CTF ví, kdo volil, kdo nikoliv a kolikrát volil každý volič. Jestliže přijde hlas, který není podepsaný právoplatným voličem nebo se jedná o hlas voliče, který již volil, CTF jej ignoruje. Díky digitálnímu podpisu nemůže nikdo změnit hlas někoho jiného, dokonce i kdybychom vypustili bod 2.

Problémem tohoto protokolu je to, že podpis je součástí hlasu; CTF ví kdo koho volil. Zašifrováním hlasů pomocí veřejného klíče CTF zabráníme komukoliv, aby odposlouchával a zjišťoval tak, kdo koho volil, ovšem musíte zcela důvěřovat CTF. Je to podobné tomu, když nad vámi stojí člen volební komise ve volební místnosti.

Tyto dva příklady ukazují, jak těžké je dosáhnout prvních tří požadavků bezpečného volebního protokolu, nehledě na ostatní.

Volení se slepými podpisy

Potřebujeme nějakým způsobem oddělit hlas od voliče při zachování autentikace. Právě toto umožňuje protokol slepého podpisu 7.12.

1. Každý volič generuje 10 sad zpráv, každá sada obsahuje platný hlas pro každou možnou variantu (např. je-li hlasem odpověď na zjišťovací otázku, každá sada obsahuje dva hlasy, jeden pro „ano“ a druhý pro „ne“.) Každá zpráva navíc obsahuje náhodně vygenerované identifikační číslo, kvůli zabránění možné shody s identifikačním číslem zpráv od jiných voličů, dostatečně velké.

2. Každý volič jednotlivě zaslepí všechny zprávy a pošle je spolu s jejich 9 z 10 zaslepujících faktorů k rukám CTF.

3. CTF zkontroluje v databázi, zda volič již dříve nepředložil své zaslepené hlasy pro podpis. Otevře devět sad, aby zjistila, zda jsou dobře utvořené a pak jednotlivě podepíše každou zprávu v sadě. Pošle je zpět voliči, jehož jméno si uloží do databáze.

4. Volič odslepí zprávy a zůstane mu sada hlasů podepsaných CTF. (Tyto hlasy jsou podepsané, ale nezašifrované, takže volič snadno vidí, který hlas je „ano“ a který je „ne“.)

5. Volič si zvolí jeden z hlasů (to je demokracie) a zašifruje jej veřejným klíčem CTF.

6. Volič zašle hlas CTF.

7. CTF dešifruje hlasy, ověří podpisy, zkontroluje databázi vůči duplicitnímu identifikačnímu číslu, uloží si sériové číslo a vyhodnotí hlasy. Výsledek voleb rozešle mezi všechna sériová čísla a jim příslušející hlasy.

Zlomyslný volič, nazvěme ho Mallory, nemůže takový systém podvést. Tento protokol zajišťuje, že hlasy jsou jednoznačné. Pokud se pokusí poslat svůj hlas dvakrát, CTF zaznamená duplicitní sériové číslo v kroku 7 a vyřadí druhý hlas. Pokusí-li se získat více hlasů podepsaných v kroku 2, CTF to objeví v kroku 3. Mallory nemůže generovat své vlastní hlasy, protože nezná soukromý klíč CTF. Z téhož důvodu nemůže ani odposlouchávat a měnit hlasy někoho jiného.

„Odstřihni a vyber" protokol v kroku 3 zajišťuje, že hlasy jsou jednoznačné. Bez tohoto kroku by si Mallory mohl vytvořit sadu hlasů, které jsou až na identifikační číslo stejné a nechat je všechny prohlásit za platné.

Zlomyslná CTF není schopna určit, kdo jak volil. Díky protokolu slepého podpisu je CTF zabráněno v nahlížení do sériových čísel hlasů před tím, než jsou odevzdány, CTF nemůže přiřadit zaslepenému hlasu, který podepsala, hlas, který eventuelně bude odevzdán. Rozesláním seznamu sériových čísel a jim příslušejících hlasů umožní voličům ujistit se, že jejich hlas byl korektně započítán.

Stále však jsou zde problémy. Jestliže krok 6 není anonymní a CTF může zaznamenávat, kdo zaslal který hlas, pak snadno zjistí, kdo jak volil. Zatímco, pokud obdrží hlasy v zamčené hlasovací urně a vyhodnotí je později, toto zjistit nemůže. Navíc, zatímco CTF není schopno přiřadit hlasy k voličům, může generovat velkou spoustu podepsaných platných hlasů a podvádět pak jejich přijetím. A jestliže Alice zjistí, že CTF změnila její hlas, nemá žádnou možnost, jak to dokázat. Protokol, který se snaží napravit tyto problémy, je následující.

Hlasování se dvěmi ústředními komisemi

Jedním řešením je rozdělení CTF na dvě volební vedení. Žádná strana by neměla sílu, aby podváděla sama sebe. Následující protokol používá s komisí Central Legitimization Agency (CLA) k potvrzení voličů a CTF ke sčítání hlasů.

1. Každý volič zašle zprávu CLA s žádostí o registrační číslo.

2. CLA zašle voliči zpět náhodné registrační číslo. CLA udržuje seznam těchto registračních čísel. CLA také uchovává seznam příjemců těchto registračních čísel pro případ, že by někdo chtěl volit dvakrát.

3. CLA pošle seznam registračních čísel CTF.

4. Každý volič si zvolí náhodně identifikační číslo. Vytvoří zprávu, která obsahuje toto číslo, registrační číslo obdržené od CLA a svůj hlas. Tuto zprávu pošle CTF.

5. CTF zkontoluje registrační číslo oproti seznamu, který obdržela od CLA v kroku 3. Pokud zde je registrační číslo, CTF je zaškrtne (aby zabránila někomu volit dvakrát). CTF přidá identifikační číslo do seznamu lidí, kteří již volili určitého kandidáta a připočte mu jeden hlas.

6. Poté, co jsou přijaty všechny hlasy, CTF rozešle výsledek spolu se seznamem identifikačních čísel a kandidátů, které jim příslušející voliči volili.

Podobně jako v předchozím protokolu, každý volič se může najít v seznamu identifikačních čísel. To mu dává důkaz, že jeho hlas byl započítán. Samozřejmě, že všechny zprávy zasílané mezi jednotlivými stranami v protokolu musí být zašifrované a podepsané, aby se zabránilo někomu vydávat se za někoho jiného nebo odposlouchávat přenos.

CTF nemůže změnit hlasy, protože každý volič se může podívat po svém identifikačním čísle. Pokud jej nenajde, nebo najde u něj jiný hlas, než který volil, okamžitě ví, že došlo ke změně jeho hlasu. CTF nemůže naplnit volební urnu, protože ta je kontrolována CLA. CLA ví, kolik voličů ověřila a jejich registrační čísla a tyto modifikace detekuje.

Mallory, který není právoplatným voličem, se může pokusit o podvod tím, že bude hádat registrační čísla. Toto nebezpečí se dá minimalizovat tak, že počet možných registračních čísel bude mnohem větší než počet aktuálních registračních čísel, například stociferná čísla pro milión voličů. Samozřejmě, že registrační čísla musí být generována náhodně.

Naproti tomu, CLA je v jistém ohledu stále ještě důvěryhodnou autoritou. Může tedy ověřit i neprávoplatné voliče nebo právoplatné ověřit vícekrát. Toto riziko se minimalizuje povinností CLA zveřejňovat seznam ověřených voličů (ale ne jejich registračních čísel). Jestliže počet voličů na tomto seznamu je menší než počet započítaných hlasů, je to špatně. Avšak, je-li více ověřených voličů než započítaných hlasů, může to znamenat, že někteří oprávnění voliči nevolili.

Tento protokol je zranitelný z hlediska kooperace mezi CLA a CTF. V tomto případě mohou porovnat databáze a zjistit, kdo koho volil.

Volení s jednoduchou ústřední komisí

Mnohem složitější protokol může být použit, chceme-li překonat kooperaci mezi CLA a CTF. Tento protokol je identický s předchozím se dvěma modifikacemi:

- CLA a CTF jsou jedinou organizací a
- pro anonymní distribuci registračních čísel v kroku 2 se používá ANDOS (*all-or-nothing disclosure of secrets*).

Protože protokol pro anonymní distribuci klíčů zabraňuje CTF, aby věděla který volič má jaké registrační číslo, nemůže CTF porovnávat registrační čísla s přijatými hlasy. CTF ale musí být důvěryhodnou v tom smyslu, že neposkytne registrační čísla neprávoplatným voličům. Tento problém se dá vyřešit slepými podpisy.

Vylepšené hlasování s jednoduchou ústřední komisí

Tento protokol také používá ANDOS. Splňuje všechny požadavky dobrého hlasovacího protokolu. Nesplňuje sice sedmý požadavek, ale má navíc ještě další dvě vlastnosti:

1. Volič může změnit názor (tj. vzít zpět svůj hlas a volit znovu) v rámci daného časového intervalu.
2. Když volič zjistí, že jeho hlas nebyl započítán, může to identifikovat a napravit, aniž by ohrozil tajnost svého hlasu.

Protokol pak vypadá následovně:

1. CTF zveřejní seznam legitimních voličů.
2. V rámci specifikovaného času oznámí volič CTF, zda bude hlasovat.
3. CTF zveřejní seznam voličů, kteří se budou účastnit voleb.
4. Každý volič obdrží identifikační číslo, I, pomocí ANDOS protokolu.
5. Každý volič si vygeneruje dvojici veřejný klíč, soukromý klíč: k, d. Jestliže v je jeho hlas, vygeneruje následující zprávu a pošle ji CTF:

   $$I, E_k(I, v)$$

   Tato zpráva musí být zaslána anonymně.
6. CTF potvrdí přijetí hlasu zasláním:

   $$E_k(I, v)$$

7. Každý volič pošle CTF:

   $$I, d$$

8. CTF dešifruje hlasy. Na konci voleb zveřejní výsledky voleb a pro každý různý hlas i seznam všech hodnot $E_k(I, v)$, které daný hlas obsahoval.
9. Zjistí-li volič, že jeho hlas byl započítán nesprávně, protestuje zasláním CTF:

   $$I, E_k(I, v), d$$

10. Pokud si volič přeje změnit svůj hlas z v na v', zašle CTF:

$$I, E_k(I, v'), d$$

Jiný protokol používá slepé podpisy místo ANDOS protokolu, ale je v podstatě shodný. Kroky 1–3 jsou přípravou pro volení. Jejich úkolem je zjistit a zveřejnit počet aktuálních voličů. Přestože někteří se nebudou voleb účastnit, redukuje se tak možnost přidání podvodných voličů CTF.

V kroku 4 mohou dva voliči obdržet totéž identifikační číslo. Tato možnost se minimalizuje opět větším počtem možných identifikačních čísel než kolik je aktuálních voličů. Pokud dva voliči pošlou hlas se stejným identifikačním číslem, CTF vygeneruje nové identifikační číslo I', vybere jednoho voliče a zveřejní:

$$I', E_k(I, v)$$

Vlastník tohoto hlasu to zjistí a pošle druhý hlas opakováním kroku 5 s novým identifikačním číslem.

Krok 6 umožňuje voličům zkontorlovat, zda CTF přijalo jejich hlas v pořádku. Není-li hlas započítán, zjistí to voliči v kroku 9. Za předpokladu, že voličův hlas je správný v kroku 6, zpráva, kterou pošle v kroku 9 dokáže, že jeho hlas nebyl započítán.

Jedním z problémů tohoto protokolu je, že zkorumpovaná CTF by mohla zabrat hlasy lidí, kteří sice odpověděli v kroku 2, ale ve skutečnosti nevolili. Další problém je složitost protokolu ANDOS. Autoři doporučují rozdělit velkou populaci voličů do menších populací, například na volební okrsky.

Jiným, vážnějším problémem je to, že CTF může zanedbávat započítání hlasu. Tento problém se nepodaří vyřešit: Alice tvrdí, že CTF úmyslně nezapočítává její hlas, ale CTF tvrdí, že Alice nikdy nevolila.

Volení bez ústřední volební komise CTF

Následující protokol pracuje zcela bez CTF, voliči vidí jeden druhého. Jeho autorem je Michael Merritt. Protokol nemůže být implementován prakticky pro více než hrstku lidí, nicméně je užitečné se z něj poučit.

Alice, Bob, Carol a Dave volí ano nebo ne (0 nebo 1). Předpokládá se, že každý volič má veřejný a soukromý klíč. Dále předpokládáme, že každý zná veřejný klíč ostatních.

1. Každý volič si zvolí odpověď a udělá následující:

   (a) Ke svému hlasu přidá náhodný řetězec.

   (b) Zašifruje výsledek předchozího kroku veřejným klíčem Davea.

   (c) Zašifruje výsledek předchozího kroku veřejným klíčem Carol.

(d) Zašifruje výsledek předchozího kroku veřejným klíčem Boba.

(e) Zašifruje výsledek předchozího kroku veřejným klíčem Alice.

(f) K výsledku předchozího kroku přidá nový náhodný řetězec a zašifruje jej veřejným klíčem Davea. Zaznamená si hodnotu náhodného řetězce.

(g) K výsledku předchozího kroku přidá nový náhodný řetězec a zašifruje jej veřejným klíčem Carol. Zaznamená si hodnotu náhodného řetězce.

(h) K výsledku předchozího kroku přidá nový náhodný řetězec a zašifruje jej veřejným klíčem Boba. Zaznamená si hodnotu náhodného řetězce.

(i) K výsledku předchozího kroku přidá nový náhodný řetězec a zašifruje jej veřejným klíčem Alice. Zaznamená si hodnotu náhodného řetězce.

Je-li E šifrovací funkce, R_i je náhodný řetězec a V hlas, jeho zpráva vypadá takto:

$$E_A(R_5, E_B(R_4, E_C(R_3, E_D(R_2, E_A(E_B(E_C(E_D(V, R_1))))))))$$

Každý volič si zaznamenává mezivýsledky v každém bodě výpočtu. Tyto výsledky použije později v protokolu na potvrzení, že jeho hlas byl započítán.

2. Každý volič zašle svou zprávu Alici.

3. Alice dešifruje všechny hlasy svým soukromým klíčem a pak odstraní všechny náhodné řetězce na této úrovni.

4. Alice zamíchá pořadím hlasů a výsledek pošle Bobovi. Každý hlas nyní vypadá takto:

$$E_B(R_4, E_C(R_3, E_D(R_2, E_A(E_B(E_C(E_D(V, R_1)))))))$$

5. Bob dešifruje všechny hlasy svým soukromým klíčem, podívá se, jestli jeho hlas je mezi nimi, odstraní všechny náhodné řetězce na této úrovni a výsledek pošle Carol. Každý hlas nyní vypadá takto:

$$E_C(R_3, E_D(R_2, E_A(E_B(E_C(E_D(V, R_1))))))$$

6. Carol dešifruje všechny hlasy svým soukromým klíčem, podívá se, jestli jeho hlas je mezi nimi, odstraní všechny náhodné řetězce na této úrovni a výsledek pošle Daveovi. Každý hlas nyní vypadá takto:

$$E_D(R_2, E_A(E_B(E_C(E_D(V, R_1)))))$$

7. Dave dešifruje všechny hlasy svým soukromým klíčem, podívá se, jestli jeho hlas je mezi nimi, odstraní všechny náhodné řetězce na této úrovni a výsledek pošle Alici. Každý hlas nyní vypadá takto:

$$E_A(E_B(E_C(E_D(V, R_1))))$$

8. Alice dešifruje všechny hlasy svým soukromým klíčem, podívá se, jestli její hlas je mezi nimi, podepíše všechny hlasy a výsledek pošle Bobovi, Carol a Daveovi. Každý hlas nyní vypadá takto:

$$S_A(E_B(E_C(E_D(V, R_1))))$$

9. Bob zkontroluje a vymaže podpis Alice. Dešifruje všechny hlasy svým soukromým klíčem, podívá se, jestli jeho hlas je mezi nimi, podepíše všechny hlasy a výsledek pošle Alici, Carol a Daveovi. Každý hlas nyní vypadá takto:

$$S_B(E_C(E_D(V, R_1)))$$

10. Carol zkontroluje a vymaže podpis Boba. Dešifruje všechny hlasy svým soukromým klíčem, podívá se, jestli jeho hlas je mezi nimi, podepíše všechny hlasy a výsledek pošle Alici, Bobovi a Daveovi. Každý hlas nyní vypadá takto:

$$S_C(E_D(V, R_1))$$

11. Dave zkontroluje a vymaže podpis Carol. Dešifruje všechny hlasy svým soukromým klíčem, podívá se, jestli jeho hlas je mezi nimi, podepíše všechny hlasy a výsledek pošle Alici, Bobovi a Carol. Každý hlas nyní vypadá takto:

$$S_D(V, R_1)$$

12. Všichni zkontrolují a vymaží podpis Davea. Ujistí se, zda jejich hlas je mezi nimi (podívají se po svém náhodném řetězci mezi hlasy).

13. Každý odstraní náhodné řetězce a započítá hlasy.

Nejenže tento protokol funguje, ale také zaručuje, že nikdo z účastníků nemůže podvádět. Alice, Bob, Dave a Carol ihned zjistí, že někdo podvádí. Není zapotřebí ani CTF, ani CLA. Abychom pochopili, jak protokol pracuje, pokusme se podvádět.

Pokusí-li se někdo hlasovat vícekrát, Alice to pozná v kroku 3, když obdrží více hlasů než je lidí. Pokud se takto pokusí podvést Alice, zjistí to Bob v kroku 4.

Mnohem horší je zaměnit někoho hlas za jiný. Protože hlasy jsou zašifrovány různými veřejnými klíči, kdokoliv si může vyrobit tolik platných hlasů, kolik potřebuje. Dešifrovací protokol má dvě kola: první kolo sestává z kroků 3–7, druhé z kroků 8–11. Výměna hlasu je detekována různě v různých kolech.

Jestliže někdo nahradí hlas někoho jiného za jiný ve druhém kole, jeho podvod je objeven okamžitě. V každém kroku jsou hlasy podepsány a poslány všem voličům. Jestliže jeden (nebo více) voličů zjistí, že jeho hlas už není mezi těmito hlasy, okamžitě zastaví protokol. Protože hlasy jsou podepsány v každém kroku a protože každý volič se může ve druhém kole protokolu vrátit zpět, je snadné zjistit, kdo záměnu provedl.

Záměna hlasu v prvním kole protokolu je záludnější. Alice to nemůže provést v kroku 3, protože Bob, Carol nebo Dave to odhalí v kroku 5, 6 nebo 7. Bob by to mohl provést v kroku 5. Pokud zamění Carolin nebo Daveův hlas (připomeňme, že neví, který hlas je kterého voliče), Carol nebo Dave toto poznají v kroku 6 nebo 7. Nevědí sice, kdo do jejich hlasu zasáhl (ačkoliv by to musel být někdo, kdo již z hlasy manipuloval), vědí však, že jejich hlas je zaměněn. Má-li Bob štěstí a trefí se svou záměnou do hlasu Alice, ona to nepozná až do druhého kola. Potom zjistí, že její hlas chybí v kroku 8. Pořád neví, kdo zaměnil její hlas. V prvním kole jsou hlasy míchány a krok od kroku odpodepisovány, takže je nemožné se v protokolu o krok vrátit zpět, aby se zjistilo, kdo záměnu provedl.

Další formou podvodu je pokus odhalit, kdo jak volil. Díky míchání hlasů v prvním kole je nemožné, aby se někdo vrátil protokolem zpět a snažil se přiřadit hlasy voličům. Vypouštění náhodných řetězců v prvním kole je také velmi důležité pro zachování anonymity. Kdyby nebyly odstraňovány, účinek míchání hlasů by mohl být zrušen znovuzašifrováním nebezpečných hlasů veřejným klíčem „míchače“. Tak, jak je protokol navržen, je důvěrnost voličů zabezpečena.

Dokonce ještě silnější, díky iniciálnímu náhodnému řetězci R_1 jsou identické hlasy zašifrovány odlišně v každém kroku protokolu. Nikdo nezná výsledek hlasu do kroku 11.

Jaké jsou problémy tohoto protokolu? Za prvé, protokol obsahuje spoustu výpočtů. Popsaný příklad měl pouze 4 voliče a byl komplikovaný. Protokol by nikdy nepracoval v reálných volbách s desítkami tisíc voličů. Za druhé, Dave zná výsledek voleb dříve než ostatní. Přestože této znalosti nemůže využít, dává mu to jistou informaci navíc, kterou ostatní nemají. Na druhé straně to je problém všech centralizovaných volebních schémat.

Třetím problémem je to, že Alice může zkopírovat kohokoliv hlas, přestože dopředu neví, čí hlas to je. Na vysvětlení, proč je to problém, předpokládejme, že volí tři lidé: Alice, Bob a Eva. Evu nezajímá výsledek voleb, ale pouze to, jak volila Alice. Takže zkopíruje Alicin hlas a výsledek voleb pak zaručeně bude shodný s hlasem Alice.

Další volební schémata

Bylo navrženo mnoho ještě složitějších bezpečných volebních protokolů. V podstatě jsou dvojího druhu. Jednak se jedná o protokoly podobné protokolu hlasování bez Ústřední volební komise, ve kterém je hlas každého voliče zamíchán tak, že již není možné tento hlas k tomuto voliči přiřadit.

Dále existují protokoly, ve kterých se voliči rozdělí do jednotlivých volebních okrsků tak, že voliči nemohou být podvedeni. Takové protokoly zabrání pouze tomu, aby se různé části vlády (nebo toho, kdo volby administruje) spikly proti voliči.

Základní myšlenkou takových protokolů je, že každý volič rozdělí svůj hlas na několik částí. Například, je-li hlas typu „ano“ - „ne“ a 1 znamená „ano“ a 0 „ne“, volič generuje několik čísel, jejichž součet dá 1 nebo 0. Tyto části zašifruje a zašle komisím, každou jedné. Každá komise zkontroluje přijaté části (existují protokoly zjišťující jejich korektnost) a výsledný hlas je pak součtem všech částí. Existují i protokoly zajišťující podmínku, že části hlasu každého voliče se opravdu sčítají na 0 nebo 1.

Jiný protokol od Davida Chauma umožňuje sledovat voliče, kteří se pokoušejí volby podvrhnout. Nicméně, volby pak musí být zahájeny znovu, tentokráte bez podvodných voličů, a proto se tento protokol nehodí pro volby většího rozsahu.

Další, složitější protokol, který řeší tyto problémy, lze nalézt v literatuře citované v knize [19]. Dokonce existuje i protokol používající multiple-key ciphers. Jiný volební protokol použitelný pro volby většího rozsahu dovoluje voličům zdržet se hlasování.

Volební protokoly fungují, ale zároveň napomáhají tomu, že za použití protokolu je jednodušší si hlasy koupit nebo je naopak prodávat. Tato motivace je tím spíš silnější, jakmile si kupující může být jist, že prodávající bude hlasovat tak, jak slíbil. Některé protokoly jsou navrženy tak, aby pro voliče nebylo možné dokázat jinému, že volil určitým způsobem.

7.13.2 Bezpečné víceúčastnické počítání

Jedná se o protokol, ve kterém se vytvoří skupina několika lidí, kteří společně speciálním způsobem počítají nějakou funkci více proměnných. Každý člen skupiny poskytuje jednu nebo více proměnných. Výsledek funkce je znám každému ve skupině, ale nikdo se nemůže dovědět nic o vstupech ostatních členů kromě těch, které jsou evidentní z výsledku funkce. Zde je příklad.

Protokol č. 1

Jak může skupina lidí spočítat průměrnou mzdu svých členů, aniž by kdokoliv znal plat kteréhokoliv jiného člena ve skupině?

1. Alice ke svému platu připočítá tajné náhodné číslo, zašifruje výsledek veřejným klíčem Boba a pošle mu ho.
2. Bob dešifruje Alicin výsledek svým soukromým klíčem. Přičte svůj plat k číslu přijatému od Alice, zašifruje výsledek veřejným klíčem Carol a pošle mu ho.
3. Carol dešifruje Bobův výsledek svým soukromým klíčem. Přičte svůj plat k číslu přijatému od Boba, zašifruje výsledek veřejným klíčem Davea a pošle mu ho.
4. Dave dešifruje Carolin výsledek svým soukromým klíčem. Přičte svůj plat k číslu přijatému od Carol, zašifruje výsledek veřejným klíčem Alice a pošle jí ho.
5. Alice dešifruje Daveův výsledek svým soukromým klíčem. Odečte své náhodné číslo z kroku 1, aby zjistila součet všech platů.
6. Alice vydělí výsledek počtem lidí (v našem případě 4) a zveřejní výsledek.

Tento protokol předpokládá, že každý je poctivý. Pokud někdo zalže a zfalšuje výši svého platu, výsledný průměr nebude správný. Vážnějším problémem je, že Alice může zfalšovat výsledek. V kroku 5 odečte číslo, které se jí zlíbí a nikdo jiný nepozná, že výsledek je špatně. V tomto jednání se dá Alici zabránit tak, že se od ní bude požadovat zavázání se k danému náhodnému číslu použitím schématu bitového závazku z kapitoly 4.9, ale jakmile Alice prozradí své tajné číslo na konci protokolu, může si Bob spočíst její plat.

Protokol č. 2

Alice a Bob jsou spolu v restauraci a přou se o to, kdo je starší. Nechtějí však druhému říci, kolik jim skutečně je. Mohou tedy svůj věk pošeptat do ucha třetí důvěryhodné straně (například číšníkovi), která porovná tato dvě čísla a výsledek oznámí Alici i Bobovi.

Tento protokol má však dva problémy. První, průměrný číšník nemusí mít schopnost vyřešit težší problém než je pouhé porovnání dvou čísel a druhý, pokud Alice a Bob opravdu budou trvat na utajení své informace, budou nuceni utopit číšníka.

Asymetrická kryptografie nabízí mnohem lepší řešení. Existuje protokol, který umožní Alici, jež zná hodnotu a a Bobovi, jenž zná hodnotu b, společně určit, zda $a < b$, aniž by Alice dostala jakoukoliv informaci o hodnotě b a Bob o hodnotě a. Navíc jsou Alice i Bob přesvědčeni o platnosti výpočtu. Protože je tento algoritmus jen částí protokolu, detaily je možno nalézt v [19].

Samozřejmě, že tento protokol nebrání před aktivními útočníky. Nikdo a nic nezabrání Alici ani Bobovi, aby o svém věku nelhali. Kdyby Bob byl programem, který by slepě prováděl tento protokol, Alice se dozví jeho věk (je věk počítačového programu doba, která uplynula od jeho napsání, nebo od jeho spuštění?) opakovaným spouštěním protokolu. Alice může udat svůj věk jako 60. Poté, co zjistí, že je starší než Bob, spustí protokol s tím, že jí je 30. Po zjištění, že Bob je starší, spustí protokol a oznámí, že má 45 let a tak dále, dokud se nedozví Bobův věk s přesností, jakou si přeje.

Za předpokladu, že účastníci nepodvádějí, je snadné tento protokol rozšířit pro větší počet účastníků. Jakýkoliv počet lidí si zjistí setříděnou posloupnost svých věků pomocí důvěryhodné aplikace protokolu a žádný z účastníků se nedozví věk jiného.

Protokol č. 3

Alice a Bob mají své koníčky. Rádi by si našli partnera se společnými zájmy. Použijeme seznamovací kanceláře, která bude používat bezpečného protokolu. Seřadíme koníčky podle abecedy. Diskrétně pomocí modemového spojení se budou Alice a Bob účastnit protokolu. Společně mohou určit, zda sdílejí stejné koníčky. Pokud ano, mohou se seznámit, pokud ne, mohou se s kanceláří rozloučit a být si jisti, že jejich koníčky zůstanou důvěrné. Nikdo, dokonce ani seznamovací kancelář, je už nebude vědět. Jak to tedy pracuje?

1. Alice zašifruje svůj koníček do řetězce sedmi čísel pomocí jednocestné hašovací funkce.

2. Alice použije tohoto sedmimístného čísla jako telefonního čísla, vytočí je a zanechá zprávu pro Boba. Jestliže nikdo neodpoví nebo číslo není obsluhováno, aplikuje jednocestnou hašovací funkci na telefonní číslo dokud nenajde někoho, kdo by s ní pomocí protokolu komunikoval.

3. Alice sdělí Bobovi, kolikrát aplikovala jednocestnou hašovací funkci na jejího koníčka.

4. Bob aplikuje stejněkrát hašovací funkci na svého koníčka. Také použije toto sedmimístné číslo jako telefonní číslo a zeptá se osoby na druhé straně, zda má nějaké vzkazy.

Poznamenejme, že Bob může útočit pomocí vybraného otevřeného textu. Může totiž aplikovat jednocestnou hašovací funkci na obvyklé koníčky a vytočit výsledná telefonní čísla, aby se podíval na zanechané vzkazy pro něj. Tento protokol funguje pouze, když takovýchto možných otevřených textů je dostatečně mnoho na to, aby to pro Boba přestalo být praktické.

Existuje též matematický protokol podobný protokolu č. 2. Alice zná a, Bob zná b a společně určí, zda $a = b$ tak, že Bob neví nic o a a Alice neví nic o b. Detaily jsou v [19].

Protokol č. 4

Problém je následující: sedmičlenná rada pravidelně hlasuje o nějakém problému. Každý z členů rady může hlasovat *ano* nebo *ne*. Kromě toho mají dvě strany možnost volit tzv. „super-hlasy“: $S-ano$ a $S-ne$. Nemusí však těmito hlasy volit, pokud chtějí, mohou volit obyčejnými obvyklými hlasy. Pokud nikdo nepoužije možnost volit super-hlasy, pak rozhoduje většina hlasů. V případě jednoho nebo dvou ekvivalentních super-hlasů jsou všechny ostatní obvyklé hlasy ignorovány. V případě dvou super-hlasů, které si protiřečí, opět rozhoduje většina ostatních hlasů. Nyní chceme protokol, který by bezpečně realizoval tento způsob voleb.

Ilustrujme na dvou příkladech volební proces. Předpokládejme, že máme 5 obvyklých voličů $N_1, \dots, N_5$ a 2 super-voliče S_1 a S_2. Zde jsou jejich hlasy na řešení problému č. 1:

S_1	S_2	N_1	N_2	N_3	N_4	N_5
$S-ano$	*ne*	*ne*	*ne*	*ne*	*ano*	*ano*

V tomto případě záleží jen na hlasu $S-ano$ a výsledek je tedy „ano“. Následuje hlasování na problém č. 2:

S_1	S_2	N_1	N_2	N_3	N_4	N_5
$S-ano$	$S-ne$	*ne*	*ne*	*ne*	*ano*	*ano*

Nyní se super-hlasy vzájemně vylučují a tedy vyhrává většina ostatních hlasů „ne“.

Pokud není nutné utajovat, zda rozhodujícím hlasem byl super-hlas nebo jen obyčejný hlas, pak je jednoduché nalézt aplikaci bezpečného volebního protokolu. Chceme-li tuto skutečnost utajit, pak je nutné použít složitějšího protokolu.

Tento způsob volení by se mohl vyskytnout skutečně v reálném životě. Může jít o část korporace, kde určití lidé mají větší váhu hlasu než ostatní nebo např. o část Organizace spojených národů, kde jisté národy mají opět větší volební sílu.

Víceúčastnické nepodmíněné bezpečné volební protokoly

Jedná se o jednoduchý případ obecného teorému: libovolnou funkci o n proměnných může spočítat n hráčů tak, že všichni budou znát výsledek, ale žádná množina méně než $n/2$ hráčů nedostane žádnou další informaci, která nevyplývá buď z jejich vlastních vstupů nebo z hodnoty výsledku.

Bezpečné vyhodnocování pomocí logických obvodů

Alice má vstup a a Bob vstup b. Společně chtějí vypočítat nějakou funkci $f(a, b)$ tak, že Alice se nic nedozví o vstupu Boba a Bob nic o vstupu Alice. Obecný problém se nazývá bezpečné vyhodnocování pomocí obvodů. Alice a Bob mohou vytvořit libovolný booleovský obvod. Tento obvod přijme vstup od Alice a od

Boba a produkuje výsledek. Bezpečný protokol pro tento obvod splňuje následující 3 podmínky:

1. Alice může vložit svůj vstup, aniž by byl Bob schopen se jej dozvědět.
2. Bob může vložit svůj vstup, aniž by byla Alice schopna se jej dozvědět.
3. Alice i Bob si mohou spočítat výstup a ujistit se, že je správný a že s žádnou z částí nikdo nemanipuloval.

7.13.3 Šíření anonymních zpráv

Nemůžete si vyjít na oběd s bandou kryptografů, aniž byste si nepřidělali vrásek na čele. David Chaum uvádí problém obědvajících kryptografů:

> Tři kryptografové usednou k obědu ve své oblíbené tříhvězdičkové restauraci. Jejich číšník jim sdělí, že v souladu se zvyklostí hotelu se účet platí anonymně. Oběd tedy může zaplatit buď jeden z kryptografů nebo NSA. Všichni kryptografové respektují právo ostatních zaplatit anonymně, ale zajímá je, zda zaplatila NSA.

Jak kryptografové Alice, Bob a Carol zjistí, že někdo z nich platí za oběd a přitom se zachovala anonymita plátce?

Chaum pokračuje v řešení problému:

> Každý z kryptografů si hodí poctivou mincí vedle svého menu mezi sebe a kryptografa po pravici tak, ža pouze oni dva vidí, co padlo. Pak každý z kryptografů nahlas řekne, zda obě mince, které vidí – tu, kterou hodil on sám a tu, kterou hodil soused po levici – padly na stejnou stranu nebo ne. Jestliže jeden z nich je plátce, řekne opak, než co vidí. Lichý počet rozdílů znamená, že platil kryptograf. Sudý počet pak, že plátcem je NSA (předpokládá se, že oběd byl zaplacen pouze jednou). Nyní, pokud platil kryptograf, žádný z ostatních dvou neví, kdo to byl.

Na důkaz toho, že protokol opravdu funguje, zkusme si představit Alici, která se snaží zjistit, který z kryptografů platil za oběd (předpokládáme, že ani Alice, ani NSA za oběd neplatily). Vidí-li dvě odlišné mince, pak oba kryptografové, Bob a Carol, řekli buď „stejné“ nebo oba řekli „různé“. (Připomeňme, že lichý počet kryptografů, kteří řekli „různé“, znamená, že jeden z nich platil.) Jestliže oba řekli „různé“, pak plátcem je kryptograf sedící blíže k minci, jež je stejná se zakrytou mincí (tou, kterou mezi sebou hodili Bob a Carol). Jestliže oba řekli „stejné“, pak plátcem je kryptograf sedící blíže k minci, která je různá od zakryté mince. Nicméně, když Alice vidí dvě mince, že jsou stejné, pak buď Bob řekl „stejné“ a Carol řekla „různé“, nebo Bob řekl „různé“ a Carol řekla „stejné“.

Jestliže zakrytá mince je stejná jako obě mince, které vidí Alice, pak je plátcem kryptograf, který řekl „různé“. Pokud je zakrytá mince různá od mincí, které vidí, pak je plátcem kryptograf, který řekl „stejné“. Ve všech těchto případech potřebuje Alice k tomu, aby určila, zda platil Bob nebo Carol, vědět, jaký je výsledek hození mincí mezi Bobem a Carol.

Konkrétní aplikace jsou samozřejmě velmi vzdáleny sezení kolem stolu u oběda. Jako příklad uveďme protokol na rozesílání anonymních zpráv.

1. Uživatelé jsou uspořádáni do kruhu.
2. V pravidených intervalech házejí sousedící uživatelé poctivou mincí za použití některého z bezpečných protokolů pro házení mincí zabraňujících tajnému odposlechu.
3. Po každém hodu každý uživatel pouze oznámí buď „stejné“ nebo „různé“.

Jestliže si Alice přeje rozeslat zprávu, jednoduše začne invertovat svoji odpověď v těch kolech, která odpovídají 1 v binární reprezentaci její zprávy. Například, kdyby zpráva byla „1001“, invertovala by svoji odpověď, řekla pravdu, řekla pravdu, a invertovala odpověď. Kdyby výsledky jejich hodů byly „různé“, „stejné“, „stejné“, „stejné“, řekla by „stejné“, „stejné“, „stejné“, „různé“.

Zaznamená-li Alice, že výsledek protokolu nesouhlasí se zprávou, kterou se pokouší vysílat, ví, že se současně s ní snaží vysílat i někdo jiný. Proto zastaví vysílání zprávy a počká náhodný počet kol před tím, než se pokusí zprávu opět vyslat. Přesné parametry musí být určeny na základě množství zpráv zasílaných po síti, ale myšlenka by měla být jasná. Aby byla věc zajímavější, tyto zprávy mohou být zašifrovány veřejným klíčem jiného uživatele. Pak, když všichni dostanou zprávu (skutečná implementace by měla přidat nějaký druh standardního počátečního a ukončovacího řetězce zprávy), pouze ten pravý příjemce ji umí dešifrovat a přečíst. Nikdo jiný neví, kdo tuto zprávu poslal. Nikdo jiný ani neví, kdo by ji mohl přečíst. Pak je také zbytečné dělat analýzu toku zpráv, která trasuje a překládá vzorky komunikace mezi účastníky, i když zprávy samotné mohou být zašifrovány.

Alternativou k házení mincí mezi sousedními stranami by mohlo být udržování společného souboru náhodných bitů. Tyto soubory by mohly být uloženy na CD-ROM, nebo jeden ze sousedů by je mohl generovat a posílat ostatním stranám, samozřejmě zašifrované. Alternativně by se mezi sebou mohli dohodnout na kryptograficky bezpečném generátoru pseudonáhodných čísel a každý z nich pak generovat tentýž řetězec pseudonáhodných bitů.

Problémem tohoto protokolu je, že zatímco podvodný účastník nemůže číst žádnou zprávu, může nabourat systém tím, že bude lhát v kroku 3. Existuje modifikace předchozího protokolu, která detekuje pokus o nabourání; problém se nazývá „Obědvající kryptografové na diskotéce“.

Kapitola 8

Šifrový standard DES a jeho kolegové

8.1 Potřeba a historie vzniku DES

Obrovský nárůst počítačové komunikace a nástup elektronických bankovních systémů v sedmdesátých letech byly jedním z hlavních faktorů pro rozhodnutí exekutivy Spojených států amerických na zajištění standardu pro bezpečný a spolehlivý transfer dat Federální banky a ostatních bank. Bylo tedy potřeba zajistitit ochranu dat jak v počítačích zpracovávaných a tamtéž ukládaných, tak i dat počítači přenášených.

Soutěž na tvorbu šifrovacího algoritmu vyhlásilo ministerstvo obchodu Spojených států amerických v roce 1973. Organizoval ji National Bureau of Standards (NBS). Požadovaným vlastnostem však nevyhověl žádný ze zaslaných algoritmů - základní idea byla, že šifrovací proces by mělo být možno provádět na malém čipu. Důsledkem pak by byla masová výroba těchto čipů, jejichž použití by zajišťovalo naprostou bezpečnost transferu dat. Přitom sám algoritmus měl být bezpečný, pochopitelný, dostupný, výkonný, jeho bezpečnost neměla záviset na utajení algoritmu, vhodný pro nejrůznější aplikace a ekonomický při elektronické realizaci – tj. čip by měl být levný. V srpnu 1974 byla výzva opakována.

Tentokrát se algoritmus podařilo nalézt. Bylo akceptováno šifrovací schéma navržené firmou IBM. Její výzkumný tým pod vedením dr. Tuchmana jej založil na zdokonalení šifrovacího algoritmu LUCIFER, který IBM používala pro své potřeby. Na rozdíl od algoritmu LUCIFER, jež používal klíč o délce 128 bitů, délka klíče navržená pro NBS byla 64 bitů a z toho bylo 8 bitů odstraněno hned na začátku šifrování.

NBS, IBM a NSA (National Security Agency) se dohodly, že NSA provede ohodnocení bezpečnosti DES (Data Encryption Standars), jak byl později nový algoritmus nazván, a IBM umožní jeho bezplatné využívání na území Spojených států amerických. DES byl patentován 24.2. 1975 a v březnu téhož roku byl zveřejněn pro všeobecné veřejné hodnocení.

Přes některé výhrady byl 23.11. 1976 přijat jako federální standard a jako takový zveřejněn 15.1. 1977. Algoritmus byl určen pro ochranu neutajovaných dat v civilním sektoru i vládních institucí vyjma ozbrojených složek, kde nemohl být používán ani k ochramě neutajovaných dat. Předpokládaná doba použití byla 10-15 let.

Algoritmus měl být po svém přijetí v roce 1977 každých pět let hodnocen a jeho platnost potvrzována NBS. V roce 1984 zahájila NSA program „Commercial COMSEC Endorsement Program" (CCEP), určený pro zabezpečování ochramy vládních informací, v rámci něhož mělo být připraveno i nahrazování DES. Byly vyvinuty hardwarové bezpečnostní moduly, provádějící šifrové algoritmy navržené pro tentokrát NSA. Tyto algoritmy byly typu 1 a typu 2. První byl určen pro utajované vládní informace a druhý pro neutajované vládní informace (měl nahradit DES). Později byla aplikace zařízení s algoritmem typu 2 rozšířena i na soukromý sektor.

Po 1.1. 1988 neměla už NSA v úmyslu doporučovat DES pro vládní použití. Bylo však zjištěno, že by to způsobilo značné potíže v bankovním sektoru. DES byl v této době už masově používán v nejrůznějších zařízeních včetně mezinárodního bankovního spojení. To vedlo ministerstvo obchodu USA ke zmírnění stanoviska NSA. V lednu 1988 NBS znovu potvrdil možnost používat DES v dalších 5 letech, avšak ne pro federální nefinanční použití. Ve federálních nefinančních institucích musel být DES nahrazován algoritmy vyvinutými NSA v rámci CCEP.

Algoritmy jsou utajovány a nesmí být exportovány ze Spojených států amerických. Čipy, které je realizují, mají ochranný kryt. Výrobci produktů musí při jejich výrobě dodržovat zvláštní postup, stanovený NSA. Tato opatření na ochranu nových algoritmů přinášejí do celé koncepce jejich využívání velké rozpory. Jejich softwarová implementace by nebyla schválena, protože by zde nemohla být zajištěna ochrana algoritmu před jeho odhalením. Přitom v mnoha aplikacích je softwarová implementace nezbytná. Otázkou zůstávají i mezinárodní spoje, kde se předpokládá vývoz těchto zařízení. Je pravděpodobné, že dnes je už povolen přísně regulovaný vývoz, i když algoritmy zůstávají nadále utajeny. Vzhledem k tomu, že uvedená obranná opatření fungují velmi dobře, o zařízeních nevíme téměř nic, dokonce neznáme, ani jejich výkonové charakteristiky. Také není dosud nic známo o rozhodnutí, které mělo být přijato NBS v únoru r. 1993, týkajícím se dalšího existence DES ve finančním sektoru pro léta 1993–1997.

DES se od roku 1977 stal nejrozšířenějším kryptografickým systémem ve světě. Je všeobecným nástrojem bezpečnosti ve veřejném i soukromém sektoru. Ve finančním sektoru se používá k ochraně všech aspektů síťové komunikace a autentizace a de facto se stal mezinárodním standardem. Je včleněn do maloobchodních i velkoobchodních systémů, je přístupný v nedrahém hardwaru a jako volný software. Pouze ve Spojených státech amerických vývoz hardwaru i softwaru s DES dosud podléhá kontrole. DES je dostupný ve všech možných formách: ve formě čipů, zásuvkových modulů do PC nebo celých šifrovacích jednotek. Jsou k dispozici oficiální programy pro softwarovou realizaci a pro kontrolu správnosti jeho realizace v zařízeních. Standardizaci na bázi DES podporuje pět největších standardizačních organizací: ABA, ANSI, GSA, ISO a NBS. Standardy NBS jsou používány jako základ standardů dalších organizací. DES se používá pro šifrování PIN (Personal Identification Number) v různých druzích platebních, přístupovýc aj. karet. Také v několika amerických vládních organizacích vytváří DES základní mechanismus ochrany (ministerstvo ekonomiky, ministerstvo spravedlnosti). V USA na bázi DES vznikl kryptografický průmysl. Je proto pravděpodobné, že výroba a zavádění DES budou nadále pokračovat.

8.2 Popis šifrovacího algoritmu DES

DES patří do třídy blokových šifer. Otevřený text (OT) je rozdělen na bloky po 64 bitech. Každý z těchto bloků M otevřeného textu je jako celek zpracováván na blok C šifrového textu (ŠT) také o délce 64 bitů. Píšeme jako obvykle

$$C = E_K(M), \quad M = D_K(C),$$

kde K je klíč o délce 56 bitů (vznikne z osmibajtového klíčového slova vynecháním jeho paritních bitů).

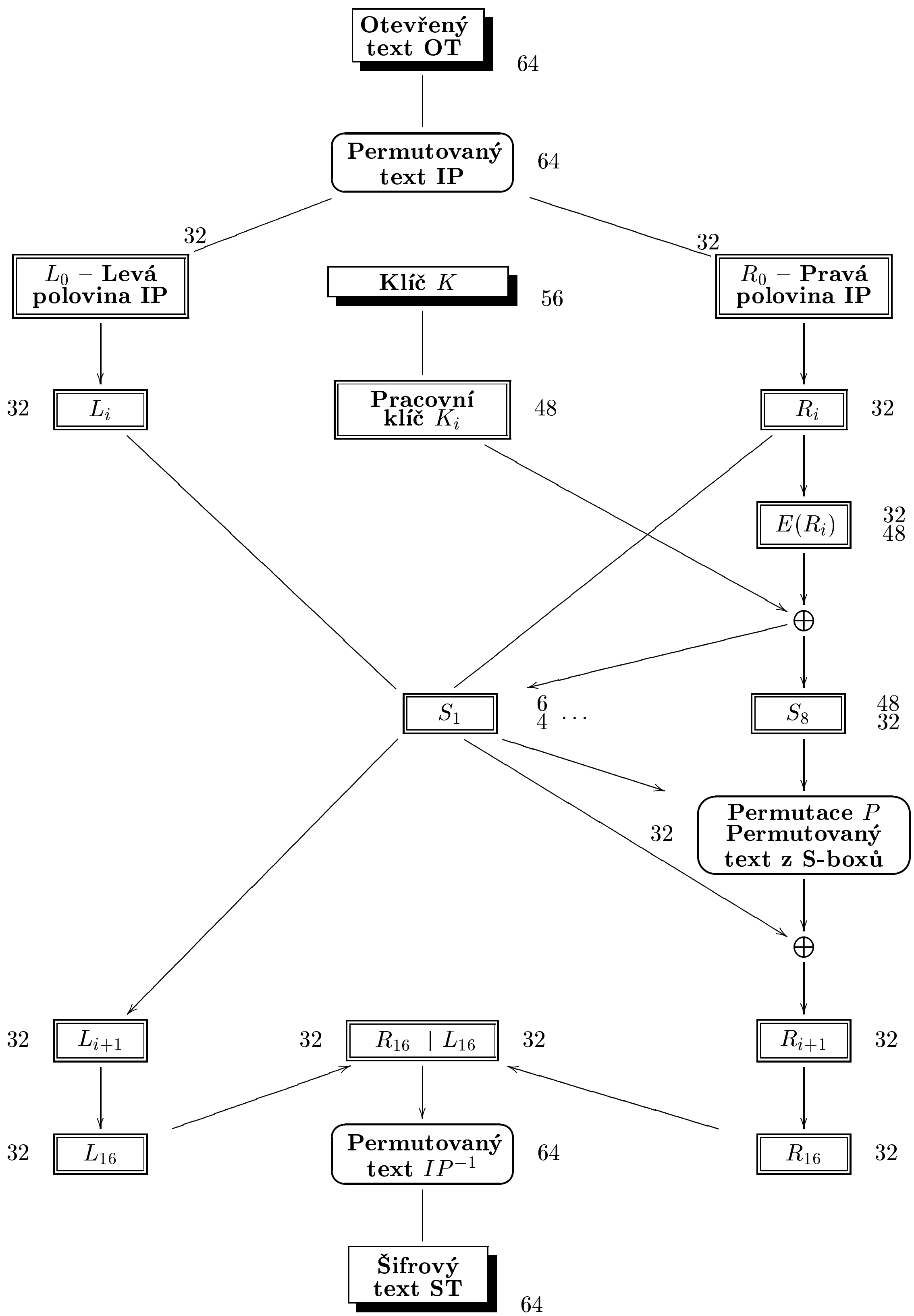
Zpracování bloku M probíhá v 16 krocích. V každém z nich je z klíče K vybráno podle určitého postupu 48 bitů tvořících pracovní klíč K_i. Vlastní blokový algoritmus při šifrování zpracovává blok M tak, že M je nejprve podroben *počáteční permutaci* IP, která permutuje všech 64 bitů, a poté je rozdělen na pravou polovinu R_0 a levou polovinu L_0, každá o 32 bitech. Pak probíhá 16 totožných kroků, kdy je z dvojice (L_i, R_i) za účasti pracovního klíče K_i vytvořena dvojice (L_{i+1}, R_{i+1}), $i = 0, 1, \ldots, 15$. R_i je nejprve rozšířen z 32 bitů na 48 bitů prostřednictvím expanze E tak, že

$$\begin{aligned} E(R_i) &= E(r_1, r_2, \ldots, r_{31}, r_{32}) \qquad (8.1)\\ &= (r_{32}, r_1, r_2, r_3, r_4, r_5, r_4, r_5, r_6, r_7, r_8, r_9, \ldots, r_{28}, r_{29}, r_{30}, r_{31}, r_{32}, r_1), \end{aligned}$$

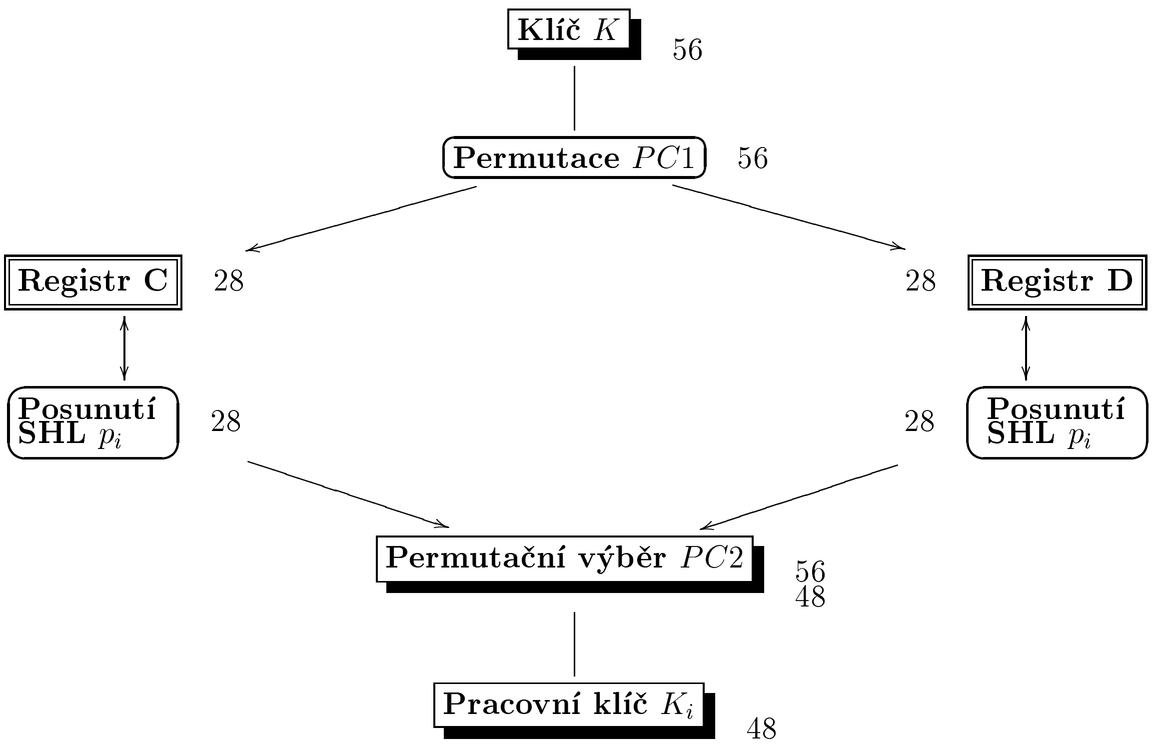
a k výsledku je v modulu 2 tj. bit po bitu přečten klíč K_i. Výstup $E(R_i) \oplus K_i$ je rozdělen do osmi částí po 6 bitech, které procházejí přes substituční boxy S_1 až S_8. Tyto *S-boxy* jsou v podstatě pamětí ROM 6×4 bity a výstup je tedy celkem 8×4=32-bitový. Většinou se tyto boxy popisují i zadávají tak, že krajní bity vstupu (1. a 6. bit) vybírají jeden ze čtyř možných boxů 4×4 bity a výstup těchto boxů se uvádí v tabulce. Přitom se jejich vstupní i výstupní 4-bitová hodnota pro jednoduchost zapisuje dekadicky jako 0 až 15. Jsou to permutace na množině $\{0, 1, \ldots, 15\}$. Výstup z S-boxů, tj. 8×4=32 bity, je upraven *permutací* P (32 na 32 bitů). Vzniklé 32-bitové slovo je označováno $f(R_i, K_i)$, neboť je funkcí klíče K_i a slova R_i. Poslední operace v i-tém kroku je

$$L_{i+1} = R_i \qquad R_{i+1} = L_i \oplus f(R_i, K_i). \tag{8.2}$$

Po proběhnutí 16. kroku je provedena záměna L_{16} a R_{16} a 64-bitový blok (R_{16}, L_{16}) je permutován *permutací* IP^{-1} (permutací inverzní k IP). Výsledek je již $C = E_K(M)$.


Otevřený text OT
64
Permutovaný text IP
64
32
32
L_0 – Levá polovina IP
Klíč K
56
R_0 – Pravá polovina IP
32
L_i
Pracovní klíč K_i
48
R_i
32
$E(R_i)$
32
48
S_1
6
4 . . .
S_8
48
32
Permutace P Permutovaný text z S-boxů
32
32
L_{i+1}
32
R_{16} | L_{16}
32
R_{i+1}
32
32
L_{16}
Permutovaný text IP^{-1}
64
R_{16}
32
Šifrový text ST
64

Tvorba klíčů K_i



Tvorba pracovních klíčů K_i probíhá při šifrování tak, že klíč K o 56 bitech je podroben permutaci $PC1$ a poté je naplněn do dvou 28-bitových *registrů* C a D. Obsah registrů C, D je v každém kroku i, $i = 0, 1, \ldots, 15$ cyklicky posunut doleva o p_i bitů (posun p_i je v krocích 0, 1, 8 a 15 jednobitový a ve zbývajících dvoubitový) a jejich výsledné 56-bitové zřetězení je podrobeno *permutačnímu výběru* $PC2$. $PC2$ svůj vstup 56 bitů nejen permutuje, ale i redukuje na 48 bitů vynecháním některých bitů. Výstup z $PC2$ už tvoří pracovní klíč K_i. Klíče K_i lze vytvářet buď souběžně se zpracováním otevřeného textu nebo předem. Postup formování klíčů K_i je takový, že v uvedených 16×48=768 bitech je každý bit klíče K obsažen 12 až 15 a objevuje se na různých pozicích. Tímto je popis šifrovací části blokového algoritmu uzavřen.

Filozofie algoritmu je taková, aby při dešifrování nemuselo být použito zcela jiné hardwarové schéma. Dešifrování $M = D_K(C)$ probíhá stejným postupem jako šifrování! Pouze pořadí výběru klíčů K_i je obrácené - místo K_0, K_1, …, K_{15} se používá pořadí K_{15}, K_{14}, …, K_2, K_1, K_0. Vytváříme-li klíče postupně, pak ve výše uvedeném popisu se místo SHL musí použít SHR a tabulka posunů $(0, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1)$, jinak zůstane vše zachováno. Tento princip zpracování (L_i, R_i) na (L_{i+1}, R_{i+1}) se nazývá *Feistelův princip* podle dr. Horsta Feistela, kryptologa IBM a vynálezce šifrovacího algoritmu LUCIFER.

Tabulka permutací

	IP	**PC1**	**PC2**	**P**
1	58	57	14	16
2	50	49	17	7
3	42	41	11	20
4	34	33	24	21
5	26	25	1	29
6	18	17	5	12
7	10	9	3	28
8	2	1	28	17
9	60	58	15	1
10	52	50	6	15
11	44	42	21	23
12	36	34	10	26
13	28	26	23	5
14	20	18	19	18
15	12	10	12	31
16	4	2	4	10
17	62	59	26	2
18	54	51	8	8
19	46	43	16	24
20	38	35	7	14
21	30	27	27	32
22	22	19	20	27
23	14	11	13	3
24	6	3	2	9
25	64	69	41	19
26	56	52	52	13
27	48	44	31	30
28	40	36	37	6
29	32	63	47	22
30	24	55	55	11
31	16	47	30	4
32	8	39	40	25

	IP	**PC1**	**PC2**
33	57	31	51
34	49	23	45
35	41	15	33
36	33	7	48
37	25	62	44
38	17	54	49
39	9	46	39
40	1	38	56
41	59	30	34
42	51	22	53
43	43	14	46
44	35	6	42
45	27	61	50
46	19	53	36
47	11	45	29
48	3	37	32
49	61	29	
50	53	21	
51	45	13	
52	37	5	
53	29	28	
54	21	20	
55	13	12	
56	5	4	
57	63		
58	55		
59	47		
60	39		
61	31		
62	23		
63	15		
64	7		

S-box S1

		(x_2, x_3, x_4, x_5)															
x_1	x_6	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	**0**	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	**1**	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
1	**0**	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
1	**1**	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S-box S2

		(x_2, x_3, x_4, x_5)															
x_1	x_6	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	**0**	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
0	**1**	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
1	**0**	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
1	**1**	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S-box S3

		(x_2, x_3, x_4, x_5)															
x_1	x_6	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	**0**	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
0	**1**	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
1	**0**	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	**1**	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S-box S4

		(x_2, x_3, x_4, x_5)															
x_1	x_6	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	**0**	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
0	**1**	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
1	**0**	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
1	**1**	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S-box S5

		(x_2, x_3, x_4, x_5)															
x_1	x_6	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	**0**	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
0	**1**	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
1	**0**	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
1	**1**	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S-box S6

		$(\mathbf{x_2}, \mathbf{x_3}, \mathbf{x_4}, \mathbf{x_5})$															
$\mathbf{x_1}$	$\mathbf{x_6}$	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	**0**	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
0	**1**	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
1	**0**	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
1	**1**	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S-box S7

		$(\mathbf{x_2}, \mathbf{x_3}, \mathbf{x_4}, \mathbf{x_5})$															
$\mathbf{x_1}$	$\mathbf{x_6}$	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	**0**	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
0	**1**	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	**0**	1	11	4	13	12	3	7	14	10	15	6	8	0	5	9	2
1	**1**	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S-box S8

		$(\mathbf{x_2}, \mathbf{x_3}, \mathbf{x_4}, \mathbf{x_5})$															
$\mathbf{x_1}$	$\mathbf{x_6}$	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	**0**	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
0	**1**	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
1	**0**	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
1	**1**	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Vlastnosti DES

Cílem počáteční permutace IP je provést rozprostření vlivu bitů z jedněch bajtů otevřeného textu M do ostatních. Permutace IP^{-1} je zde proto, aby se v dešifrovacím procesu napravil účinek IP. Kryptograficky je tato permutace nevýznamná a v rozborech je zanedbáváná. Skutečně, při útoku se znalostí otevřeného textu M, tj. při znalosti odpovídajících si dvojic M otevřeného textu a C šifrového textu, si vliv permutace IP na otevřený i šifrový text snadno eliminujeme. Uvažujeme-li $IP^{-1}(M)$ a $IP(C)$, je to totéž, jako by schéma tyto permutace vůbec neobsahovalo.

Také závěrečná výměna slov L a R je zde jen proto, aby dešifrovací proces byl shodný se šifrovacím, tj. pro možnost zahájení rekonstrukce předchozích dvojic (L, R) z následujících.

Základem schématu je transformace f, která vytváří tzv. substitučně-permutační síť. V podstatě se skládá ze substituce (S-boxy) a permutace P a zároveň zajišťuje vliv klíče na proces zpracování otevřeného textu. Klíč je na proměnné R načítán v modulu 2 jako heslo na otevřený text u Vernamovy šifry, permutace P nám připomíná historické transpoziční systémy a S-boxy substituční systémy. DES je pak jejich součinová šifra.

DES jako celek je substituční systémem, pracujícím se slovy délky 64 bitů. Je to tedy kódová kniha, která má 2^{64} kódových výrazů, neboť to je počet všech možných otevřených textů. Kódové výrazy se nevyhledávají, ale na základě znalosti klíče se vypočítávají. Ten, kdo nezná klíč, by měl být postaven před neřešitelnou úlohu „dekódování". Cílem algoritmu je, aby v uvedené kódové knize neexistovala jiná skrytá souvislost mezi klíčem K, otevřeným textem M a šifrovým textem C, která by byla využitelná k luštění.

Jednou z vlastností DESu, která se rovněž statisticky testovala, je vliv změny jednoho bitu v otevřeném textu M (resp. v klíči K), aby pravděpodobnost změny každého bitu šifrového textu C byla asi 50%. Tímto bude mj. zaručeno, že šifrové výrazy odpovídající dvěma málo se lišícím otevřeným textů budou naprosto odlišné. Podobně to je i s požadavkem vlivu klíče na šifrový text.

Dále se požadovalo, aby neexistovala žádná korelace mezi otevřeným textem M a šifrovým textem C a mezi šifrovým textem C a klíčem K. Tyto vlastnosti byly statisticky potvrzeny a testovány. Výběrové statistické testy pak mohou potvrdit jen některé vlastnosti, ale nemohou vyloučit nekonečně mnoho dalších. To se týká například vlastnosti komplementárnosti, kterou budeme diskutovat dále. Jak na ni máme přijít statistickými testy, když nevíme, že vůbec existuje?

Dalšími požadovanými vlastnostmi jsou konfúze a difúze. Jedná se o to, aby měl každý bit klíče K a otevřeného textu M vliv na každý bit šifrového textu C a aby tento vliv byl velmi komplikovaný. Na složitost zde pak mají největší vliv S-boxy. Každý výstupní bit S-boxu je nelineární funkcí všech vstupních bitů (při vyjádření operacemi $\oplus$ a $\wedge$). V případě, že by S-boxy realizovaly lineární funkci, celé schéma by realizovalo pouze lineární funkci. Potom by ovšem všech 64 bitů šifrového textu C bylo jen různými lineárními kombinacemi bitů otevřeného textu M a klíče K. Ale takovéto schéma bychom mohli okamžitě rozluštit řešením soustavy lineárních rovnic. Nelineárním vlastnostem se při studiu DES věnovala velká pozornost, neboť zajišťují požadovanou konfúzi. Bohužel, kritéria návrhu S-boxů zůstávají doposud tajná a přitom právě zde bylo nalezeno mnoho slabin.

Režimy činnosti DES

U DES byly stanoveny 4 základní módy činnosti, které byly odvozeny od různých potřeb. Základem je vlastní blokový algoritmus, tj. zobrazení $E_K : X \to X$, kde $X = \{0,1\}^{\{1,2,\dots,64\}}$ je množina všech 64-bitových bloků. Tento algoritmus je zároveň prvním módem.

1. **ECB** – Elektronická kódová kniha, píšeme $C = E_K(M)$, kde M je otevřený text a C je šifrový text. Jedná se skutečně o kódovou knihu, neboť při opakovaných blocích otevřeného textu M jsou šifrové zprávy C stejné. Proto by

bylo možné při šifrování zpráv tímto módem z operativních informací domýšlet část zašifrovaného obsahu bez nutnosti jejich luštění. Formalizované části zpráv by bylo možné opakovat v zašifrovaném tvaru (platební příkazy, standardní hlášení). Proto se tento mód nepoužívá k šifrování zpráv, ale jen k šifrování klíčů.

2. **CBC** – Zřetězení šifrového textu, symbolicky zapsáno jako

$$C_n = E_K(M_n \oplus C_{n-1}), \tag{8.3}$$

kde $(M_n)_n$ je systém bloků otevřeného textu. Tento mód využívá výstup z jednoho kroku šifrování k modeifikaci nového vstupu, takže každý blok C_n šifrového textu je závislý nejen na právě šifrovaném bloku M_n otevřeného textu, ale i na všech předchozích blocích otevřeného textu a šifrového textu.

3. **CFB** – Zpětná vazba ze šifrového textu, píšeme

$$C_n = M_n \oplus E_K(I_n), \tag{8.4}$$

kde I_n je vstupní registr posunutý o k bitů doleva a doplněný zprava k bity C_{n-1}. Přitom M_n je proud $k-$bitových znaků otevřeného textu. Je to mód vhodný tam, kde se data zpracovávají po znacích nebo po částech menších než 64 bitů. Zpětná vazba je vedena ze šifrového textu, ale jen v délce k bitů, přičemž u výstupu z blokového algoritmu je využíváno také jen k bitů. Je to systém podobný proudové šifře, avšak zachovávající vlastnost závislosti šifrového textu na předchozím otevřeném textu a šifrovém textu.

4. **OFB** – Zpětná vazba z výstupu, symbolicky zapsáno jako

$$H_n = E_K(J_n), \qquad C_n = M_n \oplus H_n, \tag{8.5}$$

kde J_n je vstupní registr posunutý o k bitů doleva a doplněný zprava k bity H_{n-1}, M_n je proud k-bitových znaků otevřeného textu a H_n je vytvořený proud hesla. Tento mód pracuje podobně jako Vernamova šifra, pouze zdroj hesla není náhodný. Mód byl navržen pro potřeby, kdy je nežádoucí, aby se chyby vzniklé na komunikačním kanálu rozšiřovaly působením šifrového algoritmu do otevřeného textu. Jde například o přenosy dat s vysokou rychlostí a redundancí (kódovaná řeč, videosignál, satelitní spoje aj.).

Poznamenejme, že u módů CBC, CBF a OFB je nutné „vstupní registr“ nejprve na začátku naplnit nějakou hodnotou. Ta se nazývá *inicializační vektor* a odesílá se většinou na začátku zprávy.

8.3 Kritika šifrového standardu

Příliš krátký klíč

Prvními velkými kritiky návrhu DES se stali Diffie a Hellman ze Stanfordské univerzity. Ve svém dopise NBS nejvíce kritizovali malý objem klíče – 56 bitů. Poukazovali na to, že nehledě na jiný možný útok k rozbití DES postačí pouhé zkoušení možných klíčů. To může provádět kdokoli, kdykoli a se zaručeným úspěchem. Dokazovali, že to bylo možné i s tehdejší technologií (1975). Uvažovali o sestrojení speciálního stroje, který by pro danou dvojici otevřený text M a šifrový klíč C hledal použitý klíč vyzkoušením všech jeho možností. V době kritiky, tj. v říjnu 1975, odhadovali, že by s tímto strojem vyluštili za jeden den jednu takovou úlohu za cenu 10 000 \$. Speciální stroj by potřeboval vyzkoušet 2^{56} tj. cca. 10^{17} klíčů za jeden den, což je přibližně 10^{12} klíčů za sekundu. Při ceně 10 \$ za čip by tento stroj stál 20 000 000 \$ (10 000 000 \$ na čipy a zbytek na ostatní náklady). Při úplném odepsání stroje za 5 let by tak denní provoz stál 10 000 \$. Ve skutečnosti se s padesátiprocentní pravděpodobností narazí na správný klíč už po vyzkoušení poloviny klíčů. To by hledání klíče dvakrát zrychlilo a tím i o polovinu snížilo náklady. Dále argumentovali tím, že během 5 let se cena technologie sníží v průměru 10-krát, a proto by za 10 let tento stroj stál pouze 200 000 \$. Přitom si v odhadu nedělali nárok na přesnost v rozmezí jednoho řádu! Doporučovali rozšířit klíč na 128 nebo 256 bitů nebo šifrovat víckrát za sebou s použitím nezávislých klíčů, tedy šifrový text

$$C = E_{K_1}(E_{K_2}(\dots E_{K_m}(M)\dots)).$$

Pracovní konference NBS

NBS reagoval na jejich argumenty svoláním pracovní konference, která se konala 30.-31. srpna 1976 a byla zaměřena na to, zda předpovědi a tvrzení Diffieho a Hellmana jsou realistické. Závěr z konference byl, že tehdejší technologií by jejich stroj nebylo možné nebylo možné sestrojit, a pokud se ho sestrojit podaří, nebude to dříve než po roce 1990 (a to asi za 72 000 000 \$). Proti zvětšení klíče bylo argumentováno tím, že by to vedlo k prodražení čipů, horším podmínkám vývozu a že to není nezbytné ani pro zamýšlené aplikace, ani pro uvažovanou životnost schématu (10-15 let).

Diffie a Hellman na to odpověděli článkem „*Exhaustive Cryptoanalysis of the NBS Data Encryption Standard*“ (1977), v němž vyčerpávajícím způsobem a podloženými argumenty dokazovali, že jejich odhady jsou správné (dnes víme, že měli pravdu). Mezitím i interní studie IBM odhadla, že by takový stroj mohl být sestrojen do roku 1981 a za cenu 200 000 000 \$, tedy v rámci jejich tolerance.

Komplementárnost a pracovní bloky

Protože Diffie, Hellman a dalších pět jejich kolegů nesouhlasili s tím, že IBM a NSA utajují výsledky svého zkoumání bezpečnosti DES a návrhová kritéria, zorganizovali během srpna 1976 vlastní krátké studiem DES „*Results of an initial attempt to cryptoanalyze the NBS Data Encryption Standard*" (1976). Z něho vyplynuly další závažné slabosti DES: vlastnost komplementárnosti, určité pravidelnosti v S-boxech a jejich dostatečná nelinearita. Poukazovali na to, že veřejný standard by měl mít i veřejná návrhová kritéria. V opačném případě to budí nedůvěru. Těm, kdo návrhová kritéria znají, umožňuje využít jejich slabosti k luštění, a jsou tedy proti ostatním zvýhodněni.

Mohlo by jít buď o osoby, které se k těmto tajným informacím dostanou neoprávněně (např. cizí tajné služby), nebo o samotné ochránce (IBM, NSA). To by u veřejného standardu nemělo být. Účastníci studia se domnívali, že NSA si ve skutečnosti nepřeje silný standard, aby jej mohla snadněji luštit. Nesouhlasili s tím, že by DES měl být odolný jen proti „luštění se znalostí otevřeného textu", což NBS do požadavků dal, ale i proti „luštění s možností volby otevřeného textu", což původně požadováno nebylo.

Druhá konference NBS

Nato NBS zorganizoval pracovní setkání matematiků v září 1976 k analýze kvality DES a možnosti, že by mohl obsahovat „trojského koně" („*Report of the workshop on cryptography on support of computer security*").

Přestože jiní účastníci na různé nedostatky také upozorňovali, fakticky nabyla předložena žádná konkrétní metoda na jejich přímé využití. Představitel IBM na konferenci k problému „trojského koně" dokonce prohlásil: „**Musíte nám důvěřovat, my všichni jsme dobří skauti**". Také NBS a NSA prohlásily, že neznají žádné metody, jak využít objevené kvazilinearity.

Bezprostředně po skončení konferencí se NBS rozhodl ponechat DES beze změn. Toto rozhodnutí způsobil zejména tlak průmyslu, který už potřeboval konečný verdikt, aby mohl vyrábět čipy. Změny v DES by tento proces nejméně o rok oddálily. Aféra kolem utajovaní však zanechala na NSA nesmazatelný stín podezření. Zvláště když vyšetřovací komise Senátu USA potvrdila, že NSA přesvědčila IBM o vhodnosti redukce délky klíče. Původně totiž IBM pro své potřeby používala 128-bitový klíč.

V roce 1978 prohlásil Davis na podporu DES následující:

> „ Každý, kdo si zakoupí kryptografické zařízení pracující na základě DES, může být ujištěn o specifické úrovni bezpečnosti dat: totiž je potřeba použít 2^{55} pokusů a metodu prověření všech možností, aby bylo možno získat klíč pro použitý šifrovací algoritmus."

Komplementárnost

Objevená vlastnost komplementárnosti spočívá v tom, že ze vztahu

$$C = E_K(M) \quad \text{plyne} \quad \text{non } C = E_{\text{non } K}(\text{non } M), \tag{8.6}$$

kde non označuje negaci bit po bitu. To je pravidelnost, která by se v takovémto případě rozhodně neměla objevit. Autoři rozboru její odhalení považovali diplomaticky řečeno za velmi znepokojující. Tvůrcům schématu tato vlastnost pravděpodobně unikla. Je umožněna tím, že negace klíče i vstupu se při jejich součtu mod 2 ve funkci f zruší:

$$f(R, K) = f(\text{non } R, \text{non } K). \tag{8.7}$$

Totiž označíme-li $R'_0 = \text{non } R_0$, $L'_0 = \text{non } L_0$ a šifrujeme-li zprávu $M' = (L'_0, R'_0)$ klíčem $K' = \text{non } K$ obdržíme postupně: $M' = \text{non } M$, platí-li, že $(L'_i, R'_i) = \text{non } (L_i, R_i)$, je i $(L'_{i+1}, R'_{i+1}) = \text{non } (L_{i+1}, R_{+1}i)$, protože

$$\begin{aligned}
(L'_{i+1}, R'_{i+1}) &= (R'_i, L'_i \oplus f(R'_i, K'_i)) = (\text{non } R_i, (\text{non } L_i) \oplus f(\text{non } R_i, \text{non } K_i)) \\
&= (\text{non } R_i, (\text{non } L_i) \oplus f(R_i, K_i)) = (\text{non } R_i, \text{non } R_{i+1}) \\
&= \text{non } (L_{i+1}, R_{+1}i).
\end{aligned}$$

Speciálně tedy $(L'_{16}, R'_{16}) = \text{non } (L_{16}, R_{16})$ tj. $\text{non } E_K(M) = E_{\text{non } K}(\text{non } M)$.

To se dá pak využít i k luštění v případě, že hledáme klíč K a máme k dispozici dvojice (M, C_1) a $(\text{non } M, C_2)$, které vznikly při šifrování tímto neznámým klíčem.

Proveďme zašifrování $E_K(M)$. Není-li tento výraz roven C_1, vylučujeme klíč K. Kdyby byl při šifrování non M použít klíč non K, obdrželi bychom

$$C_2 = E_{\text{non } K}(\text{non } M) = \text{non } E_K(M). \tag{8.8}$$

Nejsou-li si oba krajní výrazy, které máme k dispozici, rovny, můžeme vyloučit i klíč non K. Tím jsme nahradili jedno šifrování (klíčem non K) za pohé porovnávání výrazů, což je proti šifrování časově zanedbatelné. Prostor klíčů je tedy de facto poloviční a hledání 2-krát rychlejší.

Vlastnost komplementarity by navíc mohla ve špatně navržených protokolech způsobit i nežádoucí odhalení chráněných informací. Při všech možných aplikacích na ni musí být dáván pozor. Je smutné, že mnozí kryptologové (zejména z NBS) tuto vlastnost nepovažují za podstatnou. Ostatně v oficiálním popisu DES vypracovaném NBS se také tvrdí že klíč je 64-bitový. Rozhodně nejde o chybu, ale o vytvoření dojmu, že tomu tak je. S „dobrými skauty“ tedy není asi všechno v pořádku.

Nevhodný návrh S-boxů

Kromě tajných návrhových kritérií, která mohla obsahovat další skryté vady, v studiu bylo poukázáno na velkou korelovanost a nedostatečnou nelinearitu výstupních bitů S-boxů. Například box S4 má pětasedmdesátiprocentní nadbytečnost.

S-box S4	$(\mathbf{x_1}, \mathbf{x_6})$			
$(\mathbf{x_2}, \mathbf{x_3}, \mathbf{x_4}, \mathbf{x_5})$	**0 0**	**0 1**	**1 0**	**1 1**
0 0 0 0	0 1 1 1	1 1 0 1	1 0 1 0	0 0 1 1
0 0 0 1	1 1 0 1	1 0 0 0	0 1 1 0	1 1 1 1
0 0 1 0	1 1 1 0	1 0 1 1	1 0 0 1	0 0 0 0
0 0 1 1	0 0 1 1	0 1 0 1	0 0 0 0	0 1 1 0
0 1 0 0	0 0 0 0	0 1 1 0	1 1 0 0	1 0 1 0
0 1 0 1	0 1 1 0	1 1 1 1	1 0 1 1	0 0 0 0
0 1 1 0	1 0 0 1	0 0 0 0	0 1 1 1	1 1 0 1
0 1 1 1	1 0 1 0	0 0 1 1	1 1 0 1	1 0 0 0
1 0 0 0	0 0 0 1	0 1 0 0	1 1 1 1	1 0 0 1
1 0 0 1	0 0 1 0	0 1 1 1	0 0 0 1	0 1 0 0
1 0 1 0	1 0 0 0	0 0 1 0	0 0 1 1	0 1 0 1
1 0 0 1	0 1 0 1	1 1 0 0	1 1 1 0	1 0 1 1
1 1 0 0	1 0 1 1	0 0 0 1	0 1 0 1	1 1 0 0
1 1 0 1	1 1 0 0	1 0 1 0	0 0 1 0	0 1 1 1
1 1 1 0	0 1 0 0	1 1 1 0	1 0 0 0	0 0 1 0
1 1 1 1	1 1 1 1	1 0 0 1	0 1 0 1	1 1 1 0

Jeho 4 menší boxy 4×4 (při hodnotách krajních bitů $x_1x_6 = 00, 01, 10, 11$) jsou snadno odvoditelné pouze od prvního z nich (00). Platí dokonce vztah

$$S4(x \oplus 000001) = (12)(34)S4(x) \oplus (x_6, \text{non } x_6, \text{non } x_6, x_6), \tag{8.9}$$

kde $x = (x_1, x_2, x_3, x_4, x_5, x_6)$ je vstup a symbolický zápis (1,2) znamená výměnu 1. a 2. bitu. Označíme-li $y = (y_1, y_2, y_3, y_4)$ jako výstup, vyplývá odud, že součet $(y_1 \oplus y_2)$ je na x_6 závislý pouze lineárně a $(y_1 \oplus y_2 \oplus y_3 \oplus y_4)$ na něm nezávisí vůbec. Taková pravděpodobnost je nežádoucí.

Slabé a poloslabé klíče

Hodně pozornosti bylo věnováno studiu klíčů. Budou-li registry C a D na počátku obsahovat konstantní bity (buď nuly nebo jedničky), pak je operace SHL a PC2 je nijak nezmění a klíče K_i si budou rovny. Tím nastane i nežádoucí rovnost $E_K = D_K$. Celkem existují 4 tyto klíče (podle toho, zda registry C nebo D obsahují nuly nebo jedničky). Podobnou vlastnost má 6 dvojic tzv. poloslabých klíčů K_1 a K_2, pro něž platí

$$E_{K_1}E_{K_2} = Id \quad \text{neboli} \quad E_{K_1} = D_{K_2}.$$

Ty vznikají tak, že jejich klíče jsou shodné, ale v opačném pořadí jejich použití. Tím vzniká efekt, že při šifrování druhým klíčem probíhá postupné dešifrování klíčem prvním. To nastane např. u této dvojice klíčů

$$(01FE01FE01FE01FE, FE01FE01FE01FE01).$$

Později bylo identifikováno mnoho dalších tříd různě „slabých“ klíčů.

Hellmanův časopaměťový útok

Východisko k luštění lz nalézt i v efektivním využití času a paměti. Vychází se ze znalosti konkrétního otevřeného textu, který se velmi často objevuje ve zprávách, např. osm za sebou jdoucích mezer. Při zachycení jemu odpovídajícímu šifrového textu můžeme klíč luštit vyzkoušením všech jeho možností (čas t=256 šifrování, paměť m=1 slovo) nebo tím, že si předem připravíme tabulku všech možných dvojic (klíč, šifrový text) (čas t=1, paměť m=256 slov) a porovnáním šifrového textu. V obou případech je celková spotřeba „časopaměti“ t·m=256 – spotřebu času lze přelít do paměti a naopak. Hellman v roce 1980 přišel na to, jak část tabulky vypočítat předem a ve velmi zhuštěné podobě ji uložit do paměti. Luštění potom probíhá částečně jako šifrování a částečně jako porovnávání. Při optimalizaci požadavků na čas, paměť a cenu dospěl k návrhu, který zahrnoval 10 000 DES čipů, paměť 1000 GB a přípravné výpočty tabulek trvající 1 až 2 roky. To vše v ceně asi 3,6 miliónu \$. S těmito prostředky byl schopen určit až 1000 klíčů denně (v důsledku paralelismu) s průměrným čekáním na řešení 1 den. Při plné amortizaci zařízení do 5 let by cena za vyluštění jednoho klíče byla 1-100 \$. Nevýhoda Hellmanova útoku spočívá v tom, že vše se počítá pro konkrétní otevřený text. Pro jiný otevřený text musíme všechny předběžné výpočty provést znovu.

Další zajímavé vlastnosti DES

Při studiu DES do r. 1990 bylo popsáno několik dalších vlastností DES. Zmiňme se o odhalení kratší periody v jednom z modů činnosti DES. Při vazbě $k = 64$ u OFB je průměrná délka cyklu produkovaného hesla přibližně 2^{63} bloků, avšak při jiné délce k je pouze kolem 2^{31} bloků! Toto množství hesla při rychlosti šifrování 2^{16} bitů za sekundu (např. pro digitalizovanou řeč) je vyčerpáno asi za 18 hodin! Poté by došlo k dvojímu použití hesla, což je pro luštitele známá a jednoduchá úloha. Musí být proto používaná pouze plná vazba $k = 64$, při které je průměrná délka periody hesla dostatečná.

Mnohonásobné použití DES

K odstranění nedostatku malého objemu klíče bylo doporučováno několikanásobné šifrování DES. Při dvojnásobném šifrování algoritmem DES s použitím dvou různých klíčů, tj.

$$C = E_{K_2}(E_{K_1}(M)) \tag{8.10}$$

by klíč vzrostl na 112 bitů, což se zdá být dostatečné. Avšak byla nalezena technika, která by de facto redukovala klíč na pouhých 57 bitů. Spočívá v útoku se znalostí otevřeného textu. Známý otevřený text je zašifrován všemi možnými klíči (vzniká množina $E_{K_1}(M)$ o 2^{56} prvcích) a známý šifrový text je dešifrován všemi možnými klíči (vzniká množina $D_{K_2}(C)$ o 2^{56} prvcích). Průnik obou vzniklých množin obsahuje pravé řešení. Kromě toho vzniká asi 2^{48} falešných párů klíčů. Ty lze snadno vyloučit kontrolním zašifrováním na dalším známém páru (M, C). Tato metoda se nazývá „meet in the middle“. IBM doporučení Diffieho a Hellmana přijala, vylepšila a k šifrování významných informací (jako jsou například klíče nižších úrovní) používá dvou klíčů následným způsobem

$$C = E_{K_1}(D_{K_2}(E_{K_1}(M))). \tag{8.11}$$

Tato metoda trojnásobného šifrování (s dvojnásobným klíčem) má výhodu v kompatibilitě se systémy s jedním klíčem (původního šifrování se dosáhne volbou $K_1 = K_2$).

První vážný analytický útok

Obránci DES, kteří argumentovali tím, že doposud nebyl předložen žádný vážnější útok proti DES (a přehlíželi tak výše uvedené skutečnosti), mají od 11.8. 1990 ztíženou argumentaci. Tento den předložili Eli Biham a Adi Shamir z Weizmanova institutu v Izraeli metodu luštění, nazývanou diferenciální kryptoanalýza (DK). Je analytickým útokem proti DES. Snadno s ní **rozbili osmikrokovou DES, japonskou blokovou šifru FEAL a předchůdce DES LUCIFERa**. U patnáctikrokové verze DES s ní dosáhli lepších výsledků, než je zkoušení všech možných klíčů. Po dvou letech pak metodu zdokonalili i pro plně šestnáctikrokovou verzi DES a tím vyvrátili tvrzení Davise.

Podstata diferenciální kryptografie spočívá ve využití nevhodných S-boxů a slabého zpracování klíče. Vliv klíče K_i se dá totiž určitým způsobem „vyblokovat“. Uvažujme v jednom kroku DES při výpočtu $f(R, K)$ zpracování dvou vstupů: R_1 a R_2. První úvaha spočívá v tom, že diference R_1 a R_2 ($=R_1 \oplus R_2$) zůstává nezměněna, když R_1 a R_2 projdou rozšířením (expanzí) E a operací $\oplus$ s klíčem K. *Vliv klíče se v této diferenci eliminuje:*

$$(E(R_1) \oplus K) \oplus (E(R_2) \oplus K) = E(R_1) \oplus E(R_2) = E(R_1 \oplus R_2)! \tag{8.12}$$

Druhá myšlenka spočívá v tom, jak obejít nelineární S-boxy. *Pro některé diference vstupů jsou mnohé diference výstupů z S-boxů málo pravděpodobné a* *jiné velmi pravděpodobné*. Výstupní diference se ovšem v dalším kroku schématu stávají vstupními diferencemi! Permutace P nám v tom vůbec nevadí. Tímto postupným zřetězením vstupně-výstupních diferencí dostáváme nakonec pravděpodobnostní vztah mezi diferencemi otevřeného textu a diferencemi šifrového textu. Nalezneme-li páry (M, C), kde M je otevřený text, C je šifrový text, které vyhovují našemu modelu, máme o průběhu šifrování už dost informací. Určení klíčů K_{16} až K_1 je pak jen složitou technickou záležitostí. S každým krokem se ovšem příslušné pravděpodobnosti násobí, takže čím více kroků má schéma, tím je účinnost metody horší. Například na **rozbití osmikrokové verze DES stačily pouze 2 minuty**, na šestnáctikrokovou verzi je to již 2^{37} operací. Dosud je také k tomu potřeba velké množství odpovídajících si dvojic (M, C). Na druhé straně se metoda neustále zdokonaluje, takže nelze odhadnout, kde se zastaví její účinnost.

Na obranu proti ní v dnešní podobě zatím postačuje trojnásobné šifrování. Avšak důvěra v DES jako celek dostala vážnou trhlinu. Záplata v podobě mnohonásobného šifrování může brzy povolit. Kromě toho existují oblasti, u nichž přepracování z jednoduchého šifrování vyjde stejně jako zavedení jiného kryptosystému (platební karta, klíčová hospodářství, ...).

Další analytické útoky

Ve dnech 23.-27.května 1993 se v Lofthusu, malé norské vesničce, konala další ze série konferencí Mezinárodní asociace pro kryptologický výzkum - EUROCRYPT'93. K DES se vztahovalo několik příspěvků. Uveďme dva nejdůležitější.

První z příspěvků přednesl Eli Biham, jeden ze známých izraelských „rozbíječů“ DES. Svůj příspěvek s názvem „Nové typy kryptoanalytických útoků na bázi příbuzných klíčů“ uvedl předvedením čersvého výtisku knihy o diferenciální kryptoanalýze DES; napsal ji společně s Adi Shamirem. Biham si všiml, že některé rodiny kryptoschémat využívají poměrně jednoduché tvorby rundovských klíčů (u DES viz K_i), čímž mezi nimi vznikají zřejmé vztahy, a ty pak využil ke dvěma útokům.

Prvním útokem *„s možností volby otevřeného textu“* dosáhl novou redukci složitosti při luštění poměrně široké třídy kryptoschémat a předvedl rychlejší variantu tohoto útoku s využitím **vlastnosti komplementárnosti**.

Druhý útok je nový a autor ho pojmenoval „útok s možností volby vztahů v klíči s nízkou složitostí“. Připomeňme rozlišení typů útoků na šifrovací systém

- Luštění pouze ze šifrového textu. Kryptoanalytik má k dispozici libovolné množství kryptogramů, neví ovšem, jaký otevřený text byl zašifrován.

- Luštění při znalosti otevřeného a šifrového textu. Kryptoanalytik má k dispozici libovolné množství dvojic otevřeného textu a jemu příslušného kryptogramu.

- Luštění s možností volby. Kryptoanalytik má možnost vnutit šifrovacímu systému svůj otevřený text a obdržet od něj odpovídající kryptogram.

Jaký útok kryptoanalytik zvolí, záleží vždy na konkrétní úloze, před kterou je postaven. V tomto typu útoku bylo využito toho, že určité jednoduché vztahy mezi klíči mají za následek určité jednoduché vztahy mezi otevřenými a šifrovými texty. Bihamovy útoky jsou aplikovatelné na rodinu schémat LOKI a LUCIFERA. Mohly by být aplikovatelné i na DES, kdyby posuny registrů C a D v algoritmu přípravy klíčů K_i byly stejné .... Jak křehká je bezpečnost DES! Ukazuje se, že **nové výsledky útoků** nepadají z nebe, ale **jsou pouze výsledkem nového soustředěného úsilí** kryptoanalytiků. Bihamův útok je totiž zlepšením, rozšířením a zobecněním Knudsenova útoku na australský algoritmus LOKI91 z roku 1992, ale vznikl nezávisle na něm.

Druhý příspěvek byl ještě zajímavější. Přednesl ho Japonec **M. Matsui** a byl nazván „*Lineární kryptoanalytická metoda pro šifru DES*“. Nová metoda kryptoanalýzy je útokem „se znalostí otevřeného textu“. Její podstata spočívá v **objevu lineárních vztahů v kryptosystému DES (s využitelnou pravděpodobností)**. To se mu podařilo díky **slabým nelinearitám** S-boxů.

Následující výsledky kryptoanalýzy byly získány na pracovní stanici Hewlett-Packard HP 9750 (PA-RISC/66MHz) s programy vytvořenými v jazyku C.

1. Výsledky experimentů při útoku se znalostí otevřeného textu:

    - osmikroková DES je rozluštitelná s 2^{21} otevřenými texty za 40 sekund,
    - dvanáctikroková DES je rozluštitelná s 2^{33} otevřenými texty za 50 hodin,
    - šestnáctikroková DES je rozluštitelná s 2^{47} známými otevřenými texty rychleji, než je vyčepávající hledání 56-bitového klíče.

    V některých případech (není-li otevřený text náhodný) lze klíč luštit přímo ze šifrového textu. To je novinka, která přímo ohromuje. Přitom Matsuiho metoda luštění je myšlenkovitě přímočará.

2. Zde jsou výsledky experimentů pro luštění pouze ze šifrového textu:

    - je-li otevřený text tvořen anglickými texty ve znacích ASCII, osmikroková DES je rozluštitelná s 2^{29} šifrovými texty.
    - jsou-li otevřené texty náhodné znaky ASCII, je potřeba 2^{37} šifrových textů,
    - jsou-li otevřené texty tvořeny zcela náhodnými znaky, existují situace, kdy je Matsuiho metoda rychlejší než vyzkoušení všech hodnot klíče.

Z příspěvku je zřejmé, že poslední slovo ve využití této myšlenky ještě nepadlo. Dává zcela konkrétní metodu pro luštění algoritmu DES, ale má zejména význam vědecko-metodologický. Příspěvek obsahuje mnoho nových myšlenek, které budou studovány a použity jak pro kryptoanalýzu, tak pro tvorbu nových kryptoschémat. Je téměř neuvěřitelné, že celý útok je založen na tom, že „S-box S5 dává s velkou pravděpodobností lineární vztah mezi některými vstupy a některými výstupy". Je to další útok se znalostí otevřeného textu a první útok se znalostí šifrového textu na DES.

Původní varování Diffieho a Hellmana

Vraťme se nyní k původním varováním Diffieho a Hellmana z roku 1976. Jak víme, brali v úvahu 1 milión čipů provádějících asi 800 000 zkoušek klíčů za sekundu. Z konference NBS vyplynul závěr, že takový čip nebude možno vyrobit dříve než v roce 1990. Podívejme se na následující tabulku

Rok ohlášení	**Forma**	**Počet zkoušek DES klíčů/s**	**Zdroj informace**	**Uvedená cena**
1984	vyrobeno	2^{18} (256 k)	Proc. CRYPTO84	
1987	vyvinuto	2^{19} (512 k)	Proceedings of	
1987	možné vyrobit	2^{20} (1 M)	EUROCRYPT 87	40 $
1991	možné vyrobit	2^{22} (4 M)	DES Watch	
1992	možné vyrobit	2^{24} (16 M)	Proc. CRYPTO92	300 $
1995	možné vyrobit	2^{26} (64 M)	odhad	
1999	možné vyrobit	2^{28} (256 M)	odhad	

Z tabulky je vidět, že konference vývoj mírně podcenila a požadovaných parametrů bylo dosaženo už v roce 1987! Dnes je možno za 300 $ pořídit čip s rychlostí zkoušek 16 M klíčů/s. Pokud chceme v průměru luštit jeden klíč za den, potřebujeme $2^{55}/(86400 \cdot 2^{24}) = 24855$ čipů. Za ně zaplatíme $24855 \cdot 300 = 7\ 456\ 500$ $. Uvažujeme-li o stejné ceně za běžnou technologii v daném roce, zaplatíme za totéž v roce 1995 pouze 1 864 125 $ a v roce 1999 jen 466 031 $. **Cena za jeden vyluštěný klíč by tak byla v roce 1992 asi 4000 $, v roce 1995 potom jen 1000 $ a v roce 1999 pouze 250 $**. To už je báječná investice pro vysoce kvalifikované zloděje!!! Uvědomme si, že za 5 let projdou pod ochranou DES v severoamerických bankách finanční transakce v hodnotě

$$1826 \cdot 400 \cdot 10^9 \approx 700\ 000\ 000\ 000\ 000\ \$..$$

V tomto světle vypadá argumentace lidí od NBS, že „nejutajovanější věci mohou být vyzrazeny za mnohem méně než desítky miliónů dolarů, nutných k rozbití DES ..." jako omezená a účelová. Stejně účelové je i chování některých bank, které prostě uvedenou situaci neberou na vědomí (pokud o ní vůbec vědí).

Klasické argumenty jsou: „V praxi jsou tyto ceny za hardware dostatečně odstrašující, další výdaje jsou nutné na zaplacení expertů – kryptoanalytiků, úroveň znalostí o bankovních systémech je nízká, hlavní hrozba je spíše uvnitř banky než drahý a nejistý matematický útok, snadnější je klíče ukrást atd.“ Kde budou tyto argumenty banky, až jednoho dne pan prezident banky zjistí, že neznámý poberta odeslal aktiva banky někam na druhý konec světa? Tvářit se, že hrozba neexistuje, je pštrosí politika. A tu nelze dělat v oblastech týkajících se bezpečnosti. Rozbití DES je rok od roku atraktivnější a hrozba se stává hrozivější. Útočníkem nemusí být „obyčejní“ zloději, ale třeba některá z tajných služeb či mafií.

Dnes by Hellmanův časopaměťový útok mohl být proveden s 1024 DES čipy, 32 GB paměti a pár měsíci předběžných výpočtů. To vše v ceně do 3,6 miliónu \$ a s dobou čekání na řešení jeden den. I to je dost nepříjemné.

Softwarová hrozba

Softwarový útok je snadno a všem dostupný. Softwarová verze DES běží na pracovních stanicích dostatečně rychle, aby hravě rozluštila nekvalitní hesla (zašifrovaná v DES). Ukazuje se, že na novějších počítačích může software nahradit průměrně rychlé čipy DES. Současný experiment využívající nečinnost v síti 40 pracovních stanic dosáhl 2^{34} zkoušek klíčů DES během jednoho dne. Uvažujeme-li, že heslo obsahuje pouze malá písmena, je to 26^8 možností. V daném experimentu by nám k úspěchu stačilo zkoušet klíč po víkendech necelý měsíc.

Představíme-li si síť o 512 nejmodernějších pracovních stanicích, které pracují po dobu jednoho měsíce v mimopracovní době, pak jsou tyto stanice schopny provést

$$512(\text{poč.}) \cdot (22 \cdot 15 + 8 \cdot 24)(\text{hod.}) \cdot 3600(\text{sec.}) \cdot 32000(\text{klíčů/s}) \approx 2^{45} \text{ zkoušek klíčů.}$$

Pravý klíč bude tedy za tuto dobu nalezen s pravděpodobností 1:2000, tedy větší, než je pro činitele odpovědné za bezpečnost zdrávo.

Závěr

Velmi brzy budeme postaveni před kolaps dnešní bezpečnostní technologie postavené na bázi DES. Reálné luštění hrozí ze tří stran: sestrojením speciálního stroje na hledání klíčů, analytickým útokem a softwarovým útokem. Cena za jeden vyluštěný klíč bude v roce 1999 jen 250 \$. Analytický útok s využitím diferenciální kryptoanalýzy a lineární kryptoanalytické metody pro DES se bude zdokonalovat. Možnosti softwarové hrozby porostou s rozšiřováním počítačových sítí. To vše si vynutí náhradu DES. Bude to velmi drahý a náročný proces. Do té doby tam, kde je to možné, bude nutné jednoduché šifrování DES nahradit za několikanásobné. Časem se bude muset vyměnit množství různých platebních a jiných karet pracujících pod DES. Pravděpodobně se změní i politika „veřejnosti“ šifrových schémat. Budou muset být zahájeny nové mezinárodní standardizační procesy. Vnitřní ochranu některých národních komerčních systémů bude vhodnější zabezpečit po vzoru švýcarských bank národními nebo firemními neveřejnými kryptoschématy. Jedinými možnostmi jak nahradit DES jsou algoritmus RSA a diskrétní mocnění. Rozhodně bychom se z vývoje DES měli poučit.

8.4 Použití NP-těžkých problémů v šifrovacích systémech

V současné době neexistuje rychlý (tj. pracující v polynomiální době) algoritmus pro žádný NP-těžký problém, a pokud NP≠P, takový algoritmus neexistuje. Je tedy přirozený nápad založit kryptosystém na NP-těžkém problému, aby rozbití šifrovacího systému bylo ekvivalentní nalezení rychlého algoritmu, který by řešil NP-těžký problém.

To je mj. základní idea pro DES. Uvažme následující výpočetní problém.

Algebraické rovnice nad $\mathbf{Z}_2$

Vstup: Polynomy $p_1, \ldots, p_k$ v proměnných $x_1, \ldots, x_n$ nad $\mathbf{Z}_2$.

Otázka: Mají polynomy společný nulový bod $(x_1, \ldots . x_n)$ v $\mathbf{Z}_2^n$?

Například následující tři rovnice

$$
\begin{aligned}
x_1x_4x_6 + x_2x_4x_5 - 1 &= 0\\
x_1x_2 + x_2x_3 + x_3x_4 - 1 &= 0\\
x_1x_3 + x_4x_5 + x_1x_6 - 1 &= 0
\end{aligned}
$$

mají společné řešení $(1, 0, 1, 1, 1, 1)$. Ačkoliv výše uvedený problém je lehký pro malá k a n, se zvětšováním těchto parametrů se obtížnost vyřešení neustále zvyšuje. Lze navíc dokázat, že platí

Věta 8.4.1 *Problém rozhodnout, zda algebraický systém rovnic modulo 2 má řešení, je NP-těžký.*

Výše uvedená věta byla použita při tvorbě šifrovacího systému LUCIFER navrženého IBM a popsaného H. Feistelem v roce 1973. Systém pracuje následovně. Bez újmy na obecnosti lze předpokládat, že délka zprávy M je $2n$ (pokud by byla větší, použijeme rozdělení do bloků délky $2n$ a na každý blok aplikujeme šifrovací proceduru). Zprávu M rozdělíme na dva stejné bloky délky n, tj. $M = (M_0, M_1)$.

Klíč K bude vektor $\mathbf{k}$, který určuje podklíče $\mathbf{k_1}, \mathbf{k_2}, \ldots, \mathbf{k_{d-1}}$. Pro $2 \leq i \leq d$, rekurzivně definujeme

$$M_i = M_{i-2} + f(\mathbf{k_{i-1}}, M_{i-1}), \tag{8.13}$$

kde f je nějaká libovolná pevně zvolená nelineární transformace a počítáme modulo 2.

Závěrečný kryptogram C je určen vztahem

$$C = e(M, K) = (M_{d-1}, M_d). \tag{8.14}$$

Zprávu M lze získat z kryptogramu C opačným postupem – všechno, co musí příjemce zprávy provést, je obrácení pořadí 8.13. Tedy

$$M_{i-2} = M_i + f(\mathbf{k_{i-1}}, M_{i-1}), \tag{8.15}$$

$d \geq i \geq 2$, až dospějeme k původní zprávě $M = (M_0, M_1)$.

Za předpokladu, že příjemce zná tvar funkce f a hodnoty šifrovacích klíčů, je dešifrování kryptogramu právě tak těžké jako jeho zašifrování. Poznamenejme, že díky 8.15 funkce f nemusí být invertibilní.

V typické situaci, Mr. X může znát tvar funkce f, ale nesmí znát klíče. Zda je pak schopen získat klíče pomocí luštění s pomocí volby, závisí na tvaru f. Je-li f lineární transformace, je určení klíče výpočetně snadné. Avšak, v případě, že f je nelineární transformace, např.

$$f(x_1, \ldots, x_n; y_1, \ldots, y_n) = (p_1, \ldots, p_n),$$

kde každé p_i je polynom v proměnných $(x_1, \ldots, x_n; y_1, \ldots, y_n)$, pak určení klíče ze znalosti M a C se redukuje na problém vyřešení algebraických rovnic pro klíčové proměnné. Jak jsme viděli, to je NP-těžký problém.

Dosáhli jsme tedy našeho cíle: nalezení kryptosystému, který má lehký proces šifrování a dešifrování, ale který zároveň vynucuje po Mr. X, aby při hledání klíče pomocí luštění s pomocí volby řešil NP-těžký problém. Za předpokladu, že NP-těžké problémy jsou nezvládnutelné, je bezpečnost systému zajištěna.

Je zde však následující problém. Klasická teorie složitosti se téměř úplně zabývá s nejhorším možným chováním. Ačkoliv tedy určení klíče je těžký problém, nemáme jistotu, že neexistuje velký počet „snadných vstupů“.

8.5 GOST aneb GOsudarstvennyj STandart

Šifrovací algoritmus GOST

- vznikl roku 1989,
- je první a zatím poslední sovětská veřejná šifra,

- vládní standard pro kryptografickou ochranu systémů zpracování dat,
- určen pro SW i HW implementaci,
- neklade žádné omezení na stupeň utajení chráněných informací.

Klíče GOST

- GOST je klasická bloková symetrická šifra s tajným klíčem a délkou bloku 64 bitů (na první pohled velmi podobná DES),
- má 256 bitový klíč W (DES má pouze 56b),
- má 8 substitučních tabulek, každá o 64 bitech (vstup 4b, výstup 4b). Dohromady tvoří tzv. substituční blok K, který je součástí tajného klíče. Tuto část klíče lze považovat za známou, pokud je konstantní ve větší síti (bankovnictví). Na 8 substitučních funkcí $j \to K[i][j]$ nejsou ve standardu kladeny žádné požadavky; vhodné jsou permutace čísel 0–15. Singularity: $K[i][j] = const.$, $K[i][j] = j$. Přestože existují tyto singularity, nelze blok K považovat za odepsaný a klíč zužovat pouze na W.

Princip šifrování GOST (v základním režimu ECB)

- v podstatě převod jednoho bloku otevřeného textu (OT) na odpovídající blok šifrového textu (ŠT),
- klíč W rozdělen do osmi (32 bitových) slov $X[0]$ až $X[7]$
- tato jsou rozvinuta na další, tzv. rundovní klíče takto:
  - for$(i = 8; i < 24; i++)$ $X[i] = X[i-8]$;
  - for$(i = 24; i < 32; i++)$ $X[i] = X[31-i]$

  Tj. posloupnost rundovních klíčů $X[0]$ až $X[31]$ nabývá hodnot

  - $X[0], \ldots, X[7], X[0], \ldots, X[7], X[0], \ldots, X[7], X[7], \ldots, X[0]$.
- vstupní 64 bitový blok $OT = (in[1], in[0])$ rozdělen na slova $N_1 = in[0]$, $N_2 = in[1]$
- N_1 a N_2 jsou ve 32 rundách promíchána s rundovními klíči Feistelovým principem následovně:
  - for$(i = 0; i < 32; i++)$ $\{tmp = N_1; N_1 = F(N_1 + X[i])\hat{\ } N_2; N_2 = tmp\}$

- nakonec proběhne záměna slov N_1 a N_2, tj. výsledný šifrový text je:
  - $ŠT = (out[1], out[0]) = (N_1, N_2)$
- Zbývá dodefinovat funkci F: její 32 bitový vstup je rozdělen na 8 čtyřbitových slov a každé z nich projde odpovídající substituční tabulkou $K[i]$. Výsledek (8×4 bity) spojený do 32 bitového slova se poté cyklicky rotuje doleva o 11 bitů.
- Dešifrování: stejný postup, ale rundovní klíče se použijí v opačném pořadí (díky Feistelovu principu).

Kvalita šifrování

- kvalitu zajišťují
  - ◇ dostatečná délka klíče
  - ◇ dvě nelineární operace — sčítání v modulu 2^{32} a substituční boxy $K[i]$
- funkce substitučních boxů: vliv každého vstupního bitu OT (i rundovního klíče) vmíchávají v každé rundě šifrování do 4 bitů výstupu. Rotace o 11 bitů způsobí, že tyto výstupní bity v každé následující rundě ovlivňují o 4 výstupní bity více.
- další (dopředný) vliv má sčítání mezivýsledku (N_1, N_2) s $X[i]$. Teoreticky tedy může díky aritmetickému přenosu od nejnižšího řádu k nejvyššímu dojít k ovlivnění všech 8 boxů najednou během jedné rundy (záleží na počtu a vzájemných polohách jedniček).
- Nevýhoda oproti DES: cyklická rotace o 11 bitů místo promyšlené permutace používané v DES. GOST tuto nevýhodu kompenzuje dvojnásobkem počtu rund.

Kryptografická betonáž

- 32 rund je značně předimenzovaný počet, nicméně je velkou překážkou pro diferenciální a lineární kryptoanalýzu
- sloitost šifry roste s každou další rundou exponenciálně, kdežto čas lineárně (32 rund je tedy 2^{16} krát složitější než 16 rund, ale trvá jen dvakrát déle)
- počet rund byl navržen s ohledem na omezení složitosti šifry shora součtem bitů OT a klíče

Další režimy činnosti

- Režim zpětné vazby ze šifrového textu (CFB) je definován stejně jako u DES:
  - ŠT$[i] = OT[i]$ˆGOST(ŠT$[i-1]$)
- Režim „gammirovanija" se sice překládá jako OFB, ale je odlišný od OFB u DES. Transformuje blokovou šifru na proudovou. Základem je 64b čítač. Nejprve se zvolí inicializační vektor IV a po zašifrování se naplní do čítače: $(c[1], c[0]) = \mathrm{GOST}(IV[1], IV[0])$.

  Poté se vždy k $c[0]$ přičte konstanta 0x01010101 v modulu 2^{32} a k $c[1]$ 0x01010104 v modulu $(2^{32}-1)$. Zašifrováním takto vzniklé hodnoty čítače vziká heslo. Pak ŠT$[i] = OT[i]$ˆheslo$[i]$. V dalším kroku se k čítači opět přičtou příslušné konstanty a celý postup se opakuje, až je vygenerováno tolik bitů hesla, kolik má OT.
- Zabezpečovací kód zprávy (MAC) je rovněž odlišný od DES. Na tvorbu MAC je použit stejný klíč jako na data, ale jiná šifra (prvních 16 rund GOST). Tvorba MACu probíhá v režimu CBC a jeho délku $L \leq 32$b si volí uživatel.
- Režim řetězení šifrového textu (CBC) není definován.

Slabiny standardu GOST

- Existují slabé klíče $X[i] = X[7-i]$, kdy zašifrování je rovno dešifrování, ale jejich hustota v prostoru všech klíčů $(1/2^{128})$ je oproti DES $(1/2^{54})$ mizivá. Navíc GOST nemá vlastnost komplementárnosti.
- Problémy se substitučními boxy: pokud jsou (nesmyslně) zvoleny jako konstanta, GOST vůbec nešifruje. Zvolí-li se jako identity, jako by ve schématu nebyly.

Závěrem

- GOST používá několik kryptografických novinek (gammirovanije),
- nepodporuje jeden z mezinárodně standardizovaných režimů (CBC),
- předpokládá se jeho masové nasazení v Rusku.

sectionŠifrovací algoritmus Skipjack

Skipjack je nový šifrovací algoritmus vyvinutý NSA (National Security Agency) v rámci jejího nového návrhu na podmíněné sdílení klíčů. Tento nový šifrovací standard má nahradit dnes již zastarávající DES. (Zastarávající i v tom směru, že na útok hrubou silou — tj. vyzkoušení všech klíčů — dnes stačí jedna velká firma a 300 000 USD!) Proto NSA přišla s novým nápadem nazvaným Key Escrow System (podmíněné sdílení klíčů) a novým algoritmem Skipjack. Tento algoritmus je tajný, je ukryt v čipu Clipper, který je chráněn specialním pouzdrem. Při porušení tohoto pouzdra dojde k fyzikálně-chemické destrukci vnitřního zapojení a uložených klíčů.

„Popis“ algoritmu Skipjack

Skipjack pracuje podobně jako DES, který má nahradit. Je to bloková šifra, má stejné režimy činnosti jako DES a stejnou délku bloku (64 bitů). Liší se "podstatně" složitějším vnitřkem (32 rund proti 16 u DES...), délkou klíče (80 bitů proti 56 u DES) a hlavně tím, že ho zná jen NSA. Podle expertů, jimž NSA dovolila nahlédnout do tajné dokumentace, je Skipjack velmi kvalitní šifrovací algoritmus. Nevidí u něj jinou možnost luštění, než vyzkoušení všech klíčů. (Což podle nich v následujících 30 – 40 letech nehrozí.) Algoritmus se tají mimo jiné i proto, aby nemohl být realizován softwarově, protože by tím občané dostali do rukou kvalitní šifru, aniž by ji vláda mohla luštit.

Podmíněné sdílení klíčů

Tento nápad jako by vypadl z Orwellova románu: Chcete kvalitně šifrovat? Ano, máme pro vás velmi kvalitní algoritmus od NSA. Váš šifrovací klíč si ale s dovolením ponecháme. Bude uložen u vládou pověřené organizace ve dvou částech a dohromady ho lze dát jedině na základě soudního povolení.

Zajímavá je i technická realizace této myšlenky: Programování čipů provádí utajované vládní zařízení (**ZPČ**). Toto zařízení má v sobě uloženy tři tajné vládní klíče — $K1$, $K2$, FK (Family Key). Při výrobě asistují určené organizace **ORG1** a **ORG2**, u nichž budou uloženy ony dvě půlky klíče nového čipu. **ORG1** a **ORG2** generují náhodné vstupy do **ZPČ**, které k nim přidává identifikační číslo nového čipu $IČČ$ (tj. identifikační číslo nového uživatele čipu) a z těchto 3 hodnot vygeneruje 80 bitové subklíče $KU1$ pro **ORG1** a $KU2$ pro **ORG2**. Pak se spočte klíč uživatele $KU = KU1 \operatorname{xor} KU2$ a zapíše se do čipu, společně s $IČČ$ a FK. Do databáze **ORG1** však **ZPČ** zašle společně s $IČČ$ nikoli $KU1$, ale $KU1$ zašifrované klíčem $K1$, tj. hodnota $E(KU1, K1)$ a do **ORG2** obdobně $E(KU2, K2)$. Tyto dvě organizace tedy dostávají do úschovy černou skříňku, od níž nemají klíče, neví, co je uvnitř, ale mají ji bedlivě hlídat.

Postup vlády při dešifrování

V "případě potřeby" — tj. na základě soudního povolení — má vláda možnost dešifrovat komunikaci mezi dvěma uživateli **A** a **B**. Klíč, který šifruje vlastní data se nazývá relační klíč (RK). Clipper sám o sobě neřeší problém volby RK. Povinností každého výrobce šifrovacího zařízení, které obsahuje Clipper, však je, aby RK byl před vlastní komunikací "sdělen vládě". Šifrátory to provedou automaticky tak, že během vzájemné synchronizace každý zašle svému protějšku pole dat $LEAF$, které nese požadovaný RK. $LEAF$ (Law Enforcement Access Field) je definován takto:

$$LEAF = E((25\text{b } I\check{C}\check{C}, 80\text{b hodnota } E(RK, UK), 23\text{b autentizátor } A), FK).$$

Je-li zachycena podezřelá komunikace, vláda (má k dispozici FK) nejprve odšifruje $LEAF$ a dostane $I\check{C}\check{C}$, $E(RK, UK)$ a **A**. Po získání soudního povolení k vydání klíčů uživatele s číslem $I\check{C}\check{C}$ obdrží od **ORG1** $E(KU1, K1)$ a od **ORG2** $E(KU2, K2)$. Po odšifrování (má k dispozici $K1$, $K2$) z nich zjistí $KU1$, $KU2$ a určí $KU = KU1 \text{xor} KU2$. Pomocí KU odšifruje zbylou část pole $LEAF$, tj. $E(RK, UK)$, a získá konečně relační klíč RK. S pomocí Skipjacku už může dešifrovat zachycenou komunikaci (ale i tu co byla předtím i tu co bude)...

Porovnání algoritmů DES a Skipjack

	DES	Skipjack
typ šifry	bloková	bloková
vývoj algoritmu	IBM	NSA
délka bloku dat	64 bitů	64 bitů
inicializační vektor	64 bitů	64 bitů
délka uživatelského klíče	56 bitů	80 bitů
přídavné klíče	ne	ne
popis algoritmu	veřejný	státem utajovaný
možnost použití dif. kryptoanalýzy	ano	ne
výroba čipů	desítky firem	vládou USA určení výrobci
možnost přeprogramování čipů	většinou ne	ne

8.6 Proudová šifra RC4

Po mnoha letech boje s DESem a dalšími šiframi se zdá že zvítězila šifra společnosti RSA Data Security jménem RC4. Tato šifra je však vzhledem k sídlu firmy "americkým produktem" a jako taková je chráněna předpisy proti vyvážení zbraní. Její kvalita tedy byla v programech vyvážených mimo USA značně omezena, a to vedlo k jejímu rozluštění (pomocí Internetu).

Popis RC4

RC4 je proudová šifra. Uživatelem zadaný klíč se použije k tvorbě 256bajtové šifrovací tabulky. Potom se v každém kroku šifrování jednak modifikuje obsah tabulky, a jednak se z ní vygeneruje jeden bajt hesla. Heslo se použije jako základ generátoru pseudonáhodných čísel; dvěma z nich se mění sama tabulka (permutuje se) a třetí se posílá na výstup. To se pak "přixoruje" na právě zpracovávaný bajt otevřeného textu. Jako šifra proudová má RC4 mnoho výhod i nevýhod. K výhodě patří především rychlost zpracování, která z ní dělá nástroj přitažlivý pro mnoho komerčních produktů. Proudové šifry kódují jen tolik bajtů, kolik jich má otevřený text, délka dat se tedy neprodlužuje. Je také možno dešifrovat libovolný bajt, který je třeba uprostřed zprávy. A zde začínají nevýhody. Na proudové šifry lze obecně lépe útočit. Stačí změnit např. jediný bajt, pokud známe formát zprávy, neboť:

nový bajt = (*původní bajt* `XOR` *heslo*) `XOR` *diference*.

Pokud tohle bude dešifrováno operací `XOR` *heslo*, dostaneme (*původní bajt* `XOR` *diference*).

Víme-li, kde stojí v textu jednička na pozici desetitisíců, např. v textu "Proveďte převod 10 000 USD z konta x na konto y." můžeme volbou (*1* `XOR` *9*) snadno získat 80 000 dolarů. Přitom:

- neznáme klíč uživatele
- neznáme heslo
- neznáme konkrétní typ proudové šifry

Rozluštění

Šifru RC4 vyvinula firma RSA, která zaměstnává schopné kryptology, např. prof. Ronalda Rivesta a má patent na další šifry jako RSA, MD a řadu RC. Kombinací těchto prostředků ovládla RSA trh, neboť asymetrickou RSA šifru použila pro distribuci klíčů, MD pro digitální podpisy a RC4 pro vlastní rychlé šifrování. Klíč šifer byl pochopitelně omezen na 40 bitů, což se jim patrně stalo osudným. Prezident firmy Jim Bidzon se sice dal zaregistrovat jako vývozce zbraní a podařilo se mu dosáhnout podstatného zkrácení času při vládním schvalování šifer, ale zrušit omezení na délku klíče nedokázal. Firma tedy alespoň přísně tajila i zdrojový kód šifry RC (jako by si v některé z prvních lekcí Kryptologie nemohla přečíst, že nemá cenu tajit algoritmus), a tento byl v roce 1994 disassemblován patrně někým z tzv. cypherpunkers v rámci boje na právo se bránit, které tito šifrovací nadšenci interpretují jako právo zjistit kód šifry, které se má svěřit ochrana jejich dat. Výsledný zdrojový kód šifry v jazyce C byl pomocí anonymní pošty oznámen 13. července 1994 na Internetu. V hutném perlovském formátu vypadá takto:

```
#!/usr/local/bin/perl -0777-- -export-a-crypto-system-sig -RC4-3-lines-PERL
@k=unpack('C*',pack('H*',shift));for(@t=@s=0..255){$y=($k[$_%@k]+$s[$x=$_
]+$y)%256;&S;}$x=$y=0;for(unpack('C*',<>)){$x++;$y=($s[$x%=256]+$y)%256;
&S;print pack(C,$_^=$s[($s[$x]+$s[$y])%256]);}sub S{@s[$x,$y]=@s[$y,$x];}
```

Dnes již je možné si stáhnout řadu utilitek pro komfortní používání RC4. Šifropankři se domnívají, že rozlousknutím kódu ztratila firma RSA na algoritmus právo a umožňují s ním volně pracovat, pouze nedoporučují komerční využití (koupě šifry u RSA vyjde levněji, než soudní proces s ní).

Šifropankři a jejich zábavy

RC4 však pro své klady pronikla nejen do mnoha komerčních produktů (jak je zmíněno výše), ale také na Internet. Pro bezpečný přenos např. finančních dat se na síti používá protokol SSL (Secure Sockets Layer), který pracuje s RC4. Tento protokol používá např. Netscape, a právě prostřednictvím šifrované objednávky zboží Netscapem bylo loni ukázáno, jak je komunikace na Internetu "bezpečná". Vzhledem k tomu, že šifra RC4 je známa a byl použit pouze 40 bitový klíč, zbývalo jediné: vyzkoušet všechny možnosti. To provedly nezávisle na sobě dva týmy, trvalo to zhruba 100 MIPS let (30 Pentií za 8 dní). Tím bylo jasně řečeno, že je to právě délka klíče, která umožňuje každému provést tento nejjednodušší typ útoku.

Později Hal (původní zadavatel šifrové úlohy) vyhlásil další kód. Nyní bylo cílem zvládnout prohledání prostoru všech klíčů v co nejkratším čase. Přitom bylo pomocí Internetu zapojeno několik desítek lidí a výsledek se dostavil: informace o tom, že Hall si objednal tričko NetScape za 12 dolarů, byla známa za 31h 47m 36s. Třetí Halova výzva se chystá, do projektů se zapojuje řada. Pokud chcete na něco takového věnovat strojový čas, jste vítáni, je dokonce možnost získat symbolickou finanční odměnu, pokud se zrovna vám podaří získat klíč.

Závěr

Tato situace v oblasti bezpečnosti komunikace na Internetu je tedy velice neuspokojivá. Svět obecně předpokládá, že Netscape si najme programátory mimo USA (zajímavé je prohlášení firmy z 18. července, tedy tři dny po prolomení jejich systému o tom, že protokol SSL má být otevřenou platformou pro vytvoření bezpečného produktu pro výměnu důvěrných informací). Cypher pankeři předpokládají, že časem zlomí vládní dle nich nesmyslná omezení na délku klíče. A Jelcin v Rusku zavádí nutnost používat klíče o délce 256 bitů.

Přílohy aneb hrátky s RC4:

- RC4 in 3 lines of perl: `http://dcs.ex.ac.uk/ãba/rc4.html`,

- RC4 v céčku: `http://dcs.ex.ac.uk/ãba/rc4.c` ,
- Lámací projekt (mimo jiných): `http://dcs.ex.ac.uk/ aba/brute-rc4.html`.

8.7 Šifrovací algoritmus IDEA

První verze tohoto algoritmu byla poprvé prezentována v roce 1990. Jeho autory jsou Xuejia Lai a James Massey. Autoři dali tomuto algoritmu název PES (Proposed Encryption Standard). V příštím roce, když Biham a Shamir přišli s diferenciální kryptoanalýzou, autoři zesílili svůj algoritmus a nazvali ho IPES (Improved Proposed Encryption Standard), později (1992) byl název změněn na **IDEA** (International Data Encryption Algorithm).

Algoritmus IDEA se zdá být v dnešní době nejlepším a nejbezpečnějším veřejně dostupným blokovým šifrovačem. Ale jeho budoucnost není úplně jasná. Není zde moc velká snaha převzít ho jako náhradu za DES. Zčásti proto, že je patentovaný a také proto, že veřejnost stále čeká na jeho odolnost proti kryptoanalýze příštích let. Velkým plusem pro IDEA je to, že je součástí šifrovacího systému PGP.

8.7.1 Základní vlastnosti šifrovacího systému IDEA

IDEA je blokový šifrovací algoritmus. Pracuje se 64-bitovými bloky vstupního textu. Klíč je přitom dlouhý 128 bitů. Pro dešifrování je použit ten samý algoritmus jako pro šifrování. Myšlenka celého algoritmu je postavena na „míchání“ tří algebraických operací. Všechny mohou být snadno implementovány jak hardwarově tak i softwarově. Jsou to:

- **XOR**
- sčítání modulo 2^{16}
- násobení modulo 2^{16} (na tuto operaci můžeme pohlížet jako na S-box z DESu).

Všechny tyto operace pracují na 16-bitových blocích.

8.7.2 Popis systému IDEA

Obrázek 1 v příloze je přehled algoritmu IDEA. 64-bitový blok se rozdělí na čtyři 16-bitové podbloky: X1, X2, X3 a X4. Tyto podbloky jsou potom vstupem pro první kolo algoritmu. Algoritmus celkem obsahuje osm kol. V každém kole se tyto podbloky XORují, sčítají a násobí mezi sebou navzájem a s šesti 16-bitovými podklíči. Po každém kole se prohodí druhý a třetí podblok. Nakonec se na těchto čtyřech podblocích provede výstupní transformace zkombinováním se čtyřmi podklíči. V každém kole algoritmu probíhají tyto operace:

1. Vynásobení X1 a prvního podklíče. 2. Sečtení X2 a druhého podklíče. 3. Sečtení X3 a třetího podklíče. 4. Vynásobení X4 a čtvrtého podklíče. 5. XOR výsledků kroků 1. a 3. 6. XOR výsledků kroků 2. a 4. 7. Vynásobení výsledku kroku 5. s pátým podklíčem. 8. Sečtení výsledků kroků 6. a 7. 9. Vynásobení výsledku kroku 8. a šestého podklíče. 10. Sečtení výsledků kroků 7. a 9. 11. XOR výsledků kroků 1. a 9. 12. XOR výsledků kroků 3. a 9. 13. XOR výsledků kroků 2. a 10. 14. XOR výsledků kroků 4. a 10.

Výstupem kola jsou čtyři podbloky, které jsou výsledkem kroků 11., 12., 13. a 14. Prohozením dvou prostředních podbloků dostaneme vstup pro další kolo. Po osmém kole následuje finální výstupní transformace:

1. Vynásobení X1 a prvního podklíče 2. Sečtení X2 a druhého podklíče 3. Sečtení X3 a třetího podklíče 4. Vynásobení X4 a čtvrtého podklíče

Nakonec spojením těchto čtyř výstupních podbloků dostaneme šifrovaný text.

Vytváření podklíčů je také snadné. Algoritmus používá celkem 52 takových podklíčů (šest pro každé kolo a čtyři pro výstupní transformaci). 128-bitový klíč se nejprve rozdělí na 16-bitové podklíče. Toto je prvních osm podklíčů pro algoritmus (šest pro první kolo a dva pro druhé kolo). Potom klíč rotuje o 25 bitů doleva a znovu se rozdělí na osm podklíčů. A tak dále až do konce algoritmu. Dešifrování je v podstatě stejné jako šifrování až na to, že podklíče jsou reverzní a mírně odlišné. Dešifrovací klíče jsou buď aditivní nebo multiplikativní inverze šifrovacích klíčů. (Pro účely algoritmu IDEA bereme, že podblok složený ze samých nul reprezentuje $2^{16} = -1$ pro násobení modulo $2^{16}+1$; tedy multiplikativní inverze nuly je nula.) Tyto výpočty zaberou nějaký čas, ale stačí je vždy spočítta pouze jednou pro každý klíč. Tabulka ukazuje šifrovací podklíče a jim příslušné dešifrovací podklíče:

Kolo	Šifrovací podklíče	Dešifrovací podklíče
1.	$Z^{(1)}{}_1, Z^{(1)}{}_2, Z^{(1)}{}_3, Z^{(1)}{}_4, Z^{(1)}{}_5, Z^{(1)}{}_6$	$Z^{(9)}{}_1^{-1}, -Z^{(9)}{}_2, -Z^{(9)}{}_3, Z^{(9)}{}_4^{-1}, Z^{(8)}{}_5, Z^{(8)}{}_6$
2.	$Z^{(2)}{}_1, Z^{(2)}{}_2, Z^{(2)}{}_3, Z^{(2)}{}_4, Z^{(2)}{}_5, Z^{(2)}{}_6$	$Z^{(8)}{}_1^{-1}, -Z^{(8)}{}_3, -Z^{(8)}{}_2, Z^{(8)}{}_4^{-1}, Z^{(7)}{}_5, Z^{(7)}{}_6$
3.	$Z^{(3)}{}_1, Z^{(3)}{}_2, Z^{(3)}{}_3, Z^{(3)}{}_4, Z^{(3)}{}_5, Z^{(3)}{}_6$	$Z^{(7)}{}_1^{-1}, -Z^{(7)}{}_3, -Z^{(7)}{}_2, Z^{(7)}{}_4^{-1}, Z^{(6)}{}_5, Z^{(6)}{}_6$
4.	$Z^{(4)}{}_1, Z^{(4)}{}_2, Z^{(4)}{}_3, Z^{(4)}{}_4, Z^{(4)}{}_5, Z^{(4)}{}_6$	$Z^{(6)}{}_1^{-1}, -Z^{(6)}{}_3, -Z^{(6)}{}_2, Z^{(6)}{}_4^{-1}, Z^{(5)}{}_5, Z^{(5)}{}_6$
5.	$Z^{(5)}{}_1, Z^{(5)}{}_2, Z^{(5)}{}_3, Z^{(5)}{}_4, Z^{(5)}{}_5, Z^{(5)}{}_6$	$Z^{(5)}{}_1^{-1}, -Z^{(5)}{}_3, -Z^{(5)}{}_2, Z^{(5)}{}_4^{-1}, Z^{(4)}{}_5, Z^{(4)}{}_6$
6.	$Z^{(6)}{}_1, Z^{(6)}{}_2, Z^{(6)}{}_3, Z^{(6)}{}_4, Z^{(6)}{}_5, Z^{(6)}{}_6$	$Z^{(4)}{}_1^{-1}, -Z^{(4)}{}_3, -Z^{(4)}{}_2, Z^{(4)}{}_4^{-1}, Z^{(3)}{}_5, Z^{(3)}{}_6$

Rychlost algoritmu IDEA

Současné softwarové implementace jsou asi dvakrát rychlejší než DES. IDEA na stroji 386, 33 MHz šifruje data rychlostí 880 Kbit/s a na stroji 486, 66 MHz rychlostí 2.4 Mbit/s. Možná se to zdá málo, ale násobení není levná operace. Na vynásobení dvou 32-bitových čísel potřebuje procesor 486 čtyřicet cyklů (pentium 10). Hardwarové implementace pak dosahují rychlostí až 177 Mbit/s.

8.7.3 Kryptoanalýza systému IDEA

Délka klíče algoritmu IDEA je 128 bitů, to je více než dvakrát tolik co u DES. Pokud bychom chtěli provést útok hrubou silou, potřebujeme 2128 (1038)-krát šifrovat. Představme si čip, který otestuje miliardu klíčů za sekundu a nechme miliardu takových čipů luštit klíč. Potom stejně bude rozlomení klíče trvat 1013 let, tedy vice než je stáří vesmíru. Pole 1024 takových čipů by klíč rozlomilo za jeden den, ale ve vesmíru ani není dostatek atomů křemíku k sestrojení takového stroje.

Tedy útok hrubou silou není zrovna nejlepší cesta, jak na IDEU zaútočit. Ale na definitivní kryptoanalytické výsledky je algoritmus stále ještě moc nový. Návrháři se snažili, aby algoritmus byl odolný vůči diferenciální kryptoanalýze. Definovali koncept Markovova šifrovače a ukázali, že odolnost vůči diferenciální kryptoanalýze lze modelovat a kvantifikovat. (Na obrázku 2 je pro porovnání uveden původní PES algoritmus. Ten byl později zesílen proti diferenciální kryptoanalýze. Je zajímavé, jak málo někdy stačí.). Lai tvrdí, že IDEA je odolná vůči diferenciání kryptoanalýze už po čtyřech kolech z osmi. Willi Meier zkoumal tři algebraické operace použité v algoritmu. Ačkoli jsou nekompatibilní, existují případy, kdy mohou být zjednodušeny tak, že kryptoanalýza zabere pár procent původního času. Jeho útok je efektivnější než útok hrubou silou pouze pro dvoukolový algoritmus. Pro tři a více kol už je útok hrubou silou efektivnější. Osmi-kolový algoritmus je tedy bezpečný. John Daemen objevil třídu slabých klíčů. Tyto klíče nejsou slabé ve smyslu slabých klíčů pro DES. To, že jsou slabé znamená, že pokud jsou použity, útočník je dokáže rozpoznat pomocí útoku z vybraného textu. Takový slabý klíč je např.:

$$0000,0000,0x00,0000,0000,000x,xxxx,x000 \text{ (hex)}$$

Na pozici, kde se vyskytuje x lze dosadit libovolné číslo. V každém případě šance na náhodné vygenerování nějakého takového klíče je velmi malá. Navíc je snadné modifikovat algoritmus tak, aby neměl žádné slabé klíče - použít operaci XOR na každý podklíč s hodnotou $0x$0DAE.

Pracovní módy a varianty

Zapojení dvou šifrovačů za sebe (double-IDEA) bude stejně jako double-DES náchylné na „meet-in the middle“ útok. Ale, protože klíč algoritmu IDEA je více než dvakrát delší než klíč DES, je tento útok prakticky neproveditelný. Útočník by potřeboval paměť 1039 bytů. Pokud to stále někomu nestačí, lze použít tripple-IDEA:

$$C = E_{K_3}(D_{K_2}(E_{K_1}(P)))$$

Další možností je použít nezávislé podklíče. IDEA potřebuje 52 16-bitových klíčů, tedy celkem 832 bitů. Tato varianta je jistě bezpečnější, ale nikdo neví o kolik. Pokus o variaci algoritmu by mohlo být zdvojnásobení velikosti bloku. Algorimus by pracoval namísto s 16-bitovými podbloky s 32-bitovými podbloky a s 256-bitovým klíčem. Kódování by bylo rychlejší a bezpečnost by vzrostla 232-krát. Ale – bylo by to opravdu tak? Teorie v pozadí algoritmu se opírá o fakt, že 216+1 je prvočíslo. Ale 232+1 prvočíslo není. Algoritmus by se snad dal modifikovat tak, aby fungoval, ale měl by velmi odlišné vlastnosti z hlediska bezpečnosti. Lai tvrdí, že by bylo obtížné přinutit takovýto algoritmus pracovat. I když vypadá IDEA mnohem bezpečnější než DES, není vždy snadné v už existujících aplikacích DES nahradit. Pokud jsou databáze a formuláře navrženy pro 64-bitový klíč, je téměř nemožné předělat je na 128-bitový klíč. Toto je možno obejít zdvojením 64-bitového klíče - ovšem za cenu snížení bezpečnosti systému.

Pokud někomu záleží více na rychlosti než na bezpečnosti, je možné snížit počet kol v algoritmu. V současné době je nejlepší útok rychlejší než útok hrubou silou pouze pro méně než 2,5 kolový algoritmus. 4-kolový algoritmus bude tedy dvakrát rychlejší a – podle dnešních znalostí – i stejně bezpečný.

Bez záruky

IDEA je relativně nový algoritmus a mnoho otázek o něm zůstává neodpovězených. Je IDEA grupa? (Lai si myslí, že ne.) Existují nějaké dosud neobjevené cesty k rozbití tohoto algoritmu? IDEA má pevné teoretické základy, ale čas od času i bezpečně vypadající algoritmus může podlehnout nějaké nové formě kryptoanalýzy. Různé akademické a vojenské skupiny se snažily kryptoanalyzovat IDEU. Zatím však bez úspěchu.

8.8 PGP – Pretty Good Privacy

Úvod

PGP je kryptografický aplikační software autora Philipa R. Zimmermanna. Využívá systém veřejného klíče RSA a algoritmus symetrického blokového šifrování IDEA. Má promyšlený systém správy klíčů a používá se především k autentizaci a šifrování zpráv pro elektronickou poštu. Přesto je PGP pouze samostatným prostředkem a není součástí žádného standartního maileru. PGP je v současnosti k dispozici prakticky na všech platformách od MS-DOS, OS/2, LINUX, SCO a mnoha dalších typech operačních systémů. Pro území Spojených států a Kanadu se používá pro komerční účely legální verze PGP pod názvem ViaCrypt PGP verze 2.7.1. Pro území mimo USA a Kanadu je pouze pro nekomerční využití dostupný balík PGP verze 2.6.2i z května roku 1995, který umožnuje maximální délku veřejného klíče až na 2048 bitů. Tato verze PGP je v současnosti snadno přístupná na mnoha známých ftp serverech v Evropě jako jsou třeba : ftp.funet.fi nebo ftp.sunet.se.

Jak pracuje PGP

PGP je hybridní šifrovací systém, který využívá princip šifrování RSA s veřejným klíčem. Každý uživatel si generuje jeden nebo více párů tajného a veřejného klíče, přičemž se veřejné klíče zveřejňují. Tyto seznamy veřejných klíčů se dají najít na mnoha serverech na Internetu. V případě,že hodláme poslat někomu zprávu šifrovanou pomocí PGP, postupujeme takto :

1. najdeme si veřejný klíč adresáta na Internetu,
2. PGP vygeneruje náhodný konvenční 128-bitový symetrický šifrovací klíč,
3. pomocí šifry RSA a veřejného klíče adresáta se tento nový 128-bitový klíč zašifruje do tvaru, který je pak zaslán v hlavičce zprávy
4. tělo zprávy je zašifrováno pomocí toho 128-bitového klíče prostřednictvím symetrické blokové šifry IDEA švýcarského typu, která je dosud považována dokonce za silnější prostředek než RSA (šifrování pomocí IDEI používáme proto, že je o hodně rychlejší),
5. nakonec se spojí obě části a zpráva se vyšle.

Systém RSA tedy slouží pouze jako zabezpečení přenosu konvenčního klíče. Mimo jiné PGP také zprávu před zašifrováním komprimuje prostřednictvím kompresních podprogramů ZIP. PGP navíc umožňuje opatřit šifrovanou zprávu digitální signaturou, která pak dovolí příjemci ujistit se o integritě a autenticitě této zprávy. Pro digitální signaturu se používá volně dostupná funkce jednosměrného vzorkování zprávy MD5 (Message Digest Function). Ta vyprodukuje z původní zprávy kryptograficky silné 128 bitové číslo, které co nejlépe zaručuje integritu celé původní zprávy. Tento „MD5 kód“ je posléze zašifrován tajným klíčem uživatele. Příjemce po dešifrování této signatury pomocí veřejného klíče odesílatele zjistí zda je zpráva originální.

Systém správy klíčů

Systém správy klíčů je důležitou součástí PGP. Jednotlivý pár tajného a veřejného klíče pro RSA se vytváří na základě dialogu s programem PGP. Programu můžeme říct jakou délku má mít klíč - od 512 bitů až do délky 2048 bitů včetně. Program se také táže na jakousi „frázi“ hesla, která může být libovolně dlouhá a má být snadno zapamatovatelná. Tato fráze je velice důležitá, protože jsou podle ní tajná hesla šifrována pomocí konvenčního šifrovacího algoritmu IDEA. Při každém použití svého tajného klíče se pak program PGP na tuto frázi ptá. Je to jakási pojistka pro PGP. Program PGP nám tak vytvoří 2 soubory, ve kterých uchovává všechna hesla, která bychom mohli později potřebovat. Soubor 'pubring.pgp' obsahuje všechny veřejné klíče uživatele + veřejné klíče vybraných adresátů a tento soubor neustále roste. Druhý soubor 'secring.pgp' obsahuje tajné klíče uživatele, které jsou zakódovány pomocí fráze. Je třeba velice pečlivě chránit soubor tajných klíčů a také frázi hesla.

Závěr

Dá se předpokládat, že bezpečnost PGP je pro nejbližší dobu zajištěna velmi dobře. Nabourat PGP lze snad jen tak, že se podaří rozluštit konvenční symetrickou šifru IDEA, nebo faktorizovat šifrovací modul šifry RSA. IDEA používá 128-bitový klíč a zatím nebyly publikovány žádné zásadní slabiny. Jelikož lze zvolit délku klíčů pro RSA až do 2048 bitů, lze předpokládat i zde, že PGP obstojí.

Kapitola 9

Náhodné šifrování

Se zavedením náhodných nebo pravděpodobnostních kryptografických technik bylo umožněno navržení vhodné definice matematické bezpečnosti kryptografického systému a dokázání toho, že jisté kryptosystémy jsou bezpečné podle těchto definic (za vhodných předpokladů z teorie složitosti). Tyto pravděpodobnostní kryptosystémy používají jednosměrné funkce a funkce s vlastností padacích dveří v daleko komplexnější míře než tomu bylo u deterministických šifrovacích systémů. Ačkoliv použití pravděpodobnostních technik nebylo v kryptografii nic nového, novinkou byla *dokazatelná úroveň* bezpečnosti.

Dokazujeme-li, že kryptografický systém je bezpečný, je důležité opatrně specifikovat, které způsoby útoku může nepřítel Mr. X podniknout. Např. systém může být bezpečný proti slabému útoku (pasivní odposlech), ale není už bezpečný proti silnějšímu útoku (aktivní odposlech a manipulace s komunikační linkou).

Nejjednodušší formou útoku je, že pasivní nepřítel pouze sleduje legitimní uživatele kryptografického systému. Může znát pouze kryptogramy nebo nějaké dvojice otevřeného a mu odpovídajícího šifrového textu (a *known-plaintext* attack). V případě použití veřejného klíče pak může generovat polynomiální počet dvojic otevřeného a mu odpovídajícího šifrového textu.

Potencionálně nebezpečnější útok, který je považován za realistický v praktickém použití, je útok nepřítele, který je legitimním oprávněným uživatelem systému. Nepřítel pak může podniknout všechny akce dovolené legitimnímu uživateli předtím, než se pokusí dešifrovat šifrový text zaslaný mezi dvojicí uživatelů.

Můžeme obecně předpokládat, že nepřítel může manipulovat komunikačním kanálem mezi každou dvojicí legitimních uživatelů a dokonce může používat jejich kryptografické zařízení (ale nemůže získat jejich klíče). Např. pomocí tzv. *chosen-ciphertext attack* může nepřítel získat otevřený text pro kryptogram dle svého výběru.

Nepřítelův „úspěch“ by měl být definován co možná nejvelkomyslněji – totiž důkaz bezpečnosti systému by měl zamezit i tu nejslabší formu nepřítelova úspěchu.

Skromný cíl nutný pro tvorbu kryptosystému je, že většinu zpráv nemůže nepřítel odvodit jenom z kryptogramu. To je však až příliš skromný cíl, protože neřeší následující problémy:

- kryptosystém není bezpečný pro jistá pravděpodobnostní rozložení prostoru zpráv (např. prostor zpráv sestává pouze z polynomiálního počtu zpráv, které nepřítel zná),
- parciální informace o zprávách lze snadno spočítat z kryptogramu,
- lze snadno určit jednoduché, ale užitečné fakty o transmisi zprávy (např. když je jedna a tatáž zpráva odeslána vícekrát).

9.1 Náhodnost v šifrovacím procesu

V tomto odstavci se budeme věnovat použití náhodnosti v šifrovacím procesu. Prvním diskutovaným pojmem je nalezení šifrovacích schémat, která jsou *sémanticky bezpečná*. To jsou taková schémata, která zajišťují bezpečnost všech částečných informací o přenášených zprávách (uveďme např. Goldwasserův a Micaliův kryptosystém). Toto je pak těsně spjato k nevyřešeným problémům nalezení „kryptograficky bezpečných“ pseudonáhodných čísel. To jsou posloupnosti, v kterých nemůže nepřítel Mr. X v reálném čase předpovědět s úspěchem větším než je průměrný úspěch, s jakým bude vysláno další číslo.

Totéž schéma znáhodnění šifrovacího procesu spočívá na praktickém řešení fundamentálního problému vyřešeného A.D. Wynerem, když v roce 1975 spojil teorii informace, kódování a pseudonáhodná čísla ve své práci o tzv. Wynerově odposlouchávacím kanálu.

Homofonická substituce

Nejjednodušší náhodné šifrovací schéma je tzv. *homofonicka substituční šifra.* Zpráva M je řetězec z abecedy Σ_1. M je zašifrována do náhodného šifrového textu C nad abecedou Σ_2, kde $|\Sigma_1| \leq |\Sigma_2|$.

Má-li abeceda Σ_1 t symbolů, lze Σ_2 rozložit na disjunktní neprázdné podmnožiny A_i, $(1 \leq i \leq t)$ tak, že symbol s_j je náhodně zašifrován do prvku z podmnožiny A_j. Klíč K systému je uspořádaný rozklad $(A_1, \ldots, A_t)$.

Cesta, pomocí které náhodnost zvýší úroveň bezpečnosti, je jasná. Zvětšení počtu klíčů nám zvýší úroveň dvojsmyslnosti textu. Platba za náhodné šifrování spočívá v tom, že pro každý klíč každá zpráva odpovídá několika šifrovým textům, tj. množina kryptogramů bude větší než množina zpráv. Abychom toto vyrovnali, musíme mít zajištěno jisté rozšíření v komunikačním kanálu, protože je potřeba více informace na určení kryptogramu než na určení otevřeného textu. Mluvíme pak o pásové expanzi. Tento postup se nutně objeví v každém náhodném šifrovacím schématu.

Tato myšlenka není vůbec nová. Např. Gauss se domníval, že objevil nenapadnutelnou šifru zavedením homofonů. Dalším příkladem homofonní šifry byla Jefferson-Bazierova kolová šifra a její variace používané ve Spojených státech v obou světových válkách.

9.2 Sémantická bezpečnost a Goldwasser-Micaliho schéma

Šifrovací schémata, která zajišťují bezpečnost všech částečných informací o zprávách, jsou zřejmě důležité cíle kryptografického zkoumání. Takováto schémata se nazývají *sémanticky bezpečná.*

Problémem sémantické bezpečnosti jsme se v předchozím nezabývali. Např., ta bezpečnost, o které jsme předtím mluvili, nám nezajistila, že některé z bitů zakódovaných RSA-algoritmem nebo DES-algoritmem jsou snadněji určitelné než jiné. Pokusíme se neformálně popsat řešení takovýchto problémů.

Příklad. Uvažme systém s veřejným klíčem založený na faktorizaci přirozeného čísla N, které je součinem dvou prvočísel p a q. Obecný problém určení p a q z N je velmi obtížný, ale pokud poslední číslice N je 3, snadno získáme částečnou informaci, že poslední číslice p a q jsou buď 1 a 3 nebo 7 a 9.

Zvláštní případ sémantické bezpečnosti je *bitová bezpečnost*, která zajišťuje, že nejenom celou zprávu nelze dešifrovat, ale že nelze získat i jednotlivé bity. Poprvé se tímto problémem do hloubky zabývali Goldwasser a Micali v roce 1982. Ti pak navrhli nové sémanticky bezpečné schéma založené na náhodném šifrování.

Toto schéma je rozšíření myšlenky veřejného klíče, ale zabývá se zakódováním zprávy bit po bitu. Aktuální zašifrování jednoho bitu závisí na zprávě a na výsledku posloupnosti vrhů mincí. Je pak zcela bezpečné vzhledem k pojmům bitové a sémantické bezpečnosti (za předpokladu že číselně-teoretický problém) rozhodnutí, zda číslo je kvadratický zbytek, je „těžký“.

Popišme nyní aktuální šifrovací proceduru. Uvažme zprávu M jako posloupnost bitů $M_1M_2 \dots M_n$ a zašifrováním každého bitu jako řetězce. Předpokládejme, že A si přeje odeslat zprávu M příjemci B. Nejprve si B vybere svůj veřejný/soukromý klíč pomocí následujícího postupu.

1. Náhodně vybere dvě k-bitová prvočísla p a q a nechť $N = p \cdot q$. Faktorizace (p, q) čísla N bude soukromý klíč příjemce.

2. Vybere náhodně takové číslo y z čísel nesoudělných s číslem N tak, že y je kvadratický nezbytek vzhledem k N.

3. Zveřejní jako svůj *veřejný klíč* uspořádanou dvojici (N, y).

Odesílatel A zakóduje svoji zprávu $M = M_1 M_2 \dots M_n$ podle následujícího algoritmu.

Šifrovací algoritmus

Pro každý bit M_i A vybere náhodně číslo x z množiny čísel $N^* = \{z : 1 \leq z \leq N,\ (N, z) = 1\}$. M_i pak zakóduje jako

$$e(M_i) = \begin{cases} yx^2 \bmod N & \text{pokud } M_i = 1, \\ x^2 \bmod N & \text{pokud } M_i = 0. \end{cases} \tag{9.1}$$

Kryptogram C má pak tvar

$$C = e(M_1)e(M_2)\dots e(M_n). \tag{9.2}$$

Protože N je $2k$-bitové přirozené číslo, je jasné, že kryptogram je mnohonásobně delší než M.

Pro okamžik předpokládejme, že šifrovací proces a vytvoření registru veřejných klíčů lze provést v polynomiálním čase (ve skutečnosti v čase $O(nk^2)$).

Věnujme se nyní dešifrovacímu procesu prováděném příjemcem B.

Dešifrovací algoritmus

Označme přirozená čísla $e(M_i)$ jako e_i, $1 \leq i \leq n$. Pak procedura pro dešifrování je snadná:
je-li e_i kvadratický zbytek modulo N, je $M_i = 0$,
není-li e_i kvadratický zbytek modulo N, je $M_i = 1$.

Dodatečná (trapdoor) informace příjemce B je, že, protože zná faktorizaci M na prvočísla p a q, je schopen velice rychle rozhodnout z Eulerova kritéria, zda je či není e_i kvadratický zbytek modulo p a q. Protože to nastává právě tehdy, když e_i je kvadratický zbytek modulo N, jsme hotovi.

Příklad. Uvažme Goldwasser-Micaliho systém s s bezpečnostním parametrem k. Pak šifrovací algoritmus e zašle bit $M_i = 1$ do jednoho z čísel

$$yx_1^2, yx_2^2, \dots, yx_{\varphi(N)}^2 \pmod N,$$

kde $x_1, x_2, \dots, x_{\varphi(N)} \pmod N$ leží v N^* a (N, y) je příjemcův veřejný klíč.

Šifrovací algoritmus e zašle bit $M_j = 0$ do jednoho z čísel

$$x_1^2, x_2^2, \dots, x_{\varphi(N)}^2 \pmod N.$$

Aby bylo šifrování bezpečné, musí být N dostatečně velké, aby bylo prakticky nemožné rozhodnout, zda se jedná o kvadratický zbytek či ne. Protože to závisí na faktorizaci N, musí mít N alespoň 200 číslic. Tedy jedná se o velmi neefektivní algoritmus a míra komunikace drasticky klesá.

Popišme nyní typ nepřítele, se kterým jsme připraveni se utkat. Předpokládejme, že máme co činit s pasivním odposloucháváčem přenosové linky, který zná prostor zpráv, pravděpodobnostní rozdělení zpráv a šifrovací algoritmus. Nechť zná dále kryptogram a pokouší se z něho získat zprávu.

Pokud dovolíme nepříteli neohraničenou dobu a výpočetní kapacity, bude schopen rozluštit zprávu použitím úplného prohledávání. Co ale skutečně chceme, je situace, ve které nepřítel nemůže spoléhat na to, že rozšifruje kryptogram v rozumné době.

Pokud budeme uvažovat Goldwasser-Micaliho schéma, poznamenejme následující:

(a) Každý *bit* je zašifrován jako přirozené číslo modulo N a tedy je transformován do $2k$-bitového řetězce.

(b) Šifrovací funkce e je náhodná v tom smyslu, že tentýž bit může být transformován do různých řetězců v závislosti na výběru náhodné proměnné x. Proto mluvíme o *pravděpodobnostním šifrování*.

Bezpečnost systému závisí na předpokládané nemožnosti rozhodnutí nepřítelem, zda se jedná o kvadratický zbytek modulo N a toto pak závisí na délce N, tj. na $2k$. Lze tedy považovat k za *bezpečnostní parametr* našeho šifrovacího systému.

Jakmile budeme považovat k za náš parametr, ptáme se nyní, zda nikdo omezený časem ohraničeným polynomem v k nemůže prolomit šifrovací systém. Goldwasser a Micali dokázali, že jejich systém má tuto vlastnost v extrémně silném smyslu.

Předpokládejme, že si můžeme představit nepřítel Mr. X jako prodavače zařízení na prolomení kódů. Aby byl schopen demonstrovat svoje zboží, spustí svůj polynomiálně omezený přístroj T na odposlech linky. Pak použije *hledač zpráv* F s výpočetními zdroji omezenými polynomem v k, aby prohledal prostor zpráv M_1 a M_2 tak, že zašifrování M_1 a M_2 jsou rozlišitelná přístrojem T.

Řekneme, že systém je *polynomiálně bezpečný*, jestliže polynomiálně omezený hledač zpráv F nemůže najít dvě zprávy M_1 a M_2, které jsou rozlišitelné polynomiálně omezeným přístrojem T na odposlech linky. To znamená, že je-li dáno C (kryptogram vzniklý buď z M_1 nebo z M_2), T by neměl získat žádnou výhodu v porozumění toho, která ze dvou zpráv M_1 a M_2 byla zašifrována do kryptogramu C.

Goldwasserovi a Micalimu se podařilo dokázat o svém šifrovacím schématu, že je sémanticky bezpečné proti každé pohrůžce polynomiálně omezeného útočníka za předpokladu, že problém kvadratického zbytku je obtížný v následujícím smyslu. Pro kladné přirozené číslo k je množina H_k *obtížných složených přirozených čísel* definována jako

$$H_k = \{N : N = p \cdot q, \text{kde } p \text{ a } q \text{ jsou } k\text{−bitová prvočísla}\}.$$

Předpoklad, že problém kvadratického zbytku je nezvládnutelný

Buď P a Q pevné polynomy. Pro každé přirozené číslo k, buď C_k booleovský obvod se dvěma $2k-$bitovými vstupy a jedním booleovským výstupem s následujícími vlastnostmi:

(a) Je-li dán vstup (n, x), pak C_k správně rozhodne pro alespoň $\frac{1}{P(k)}$ čísel n z H_k, zda je či není x kvadratický zbytek modulo n pro každé přirozené číslo x menší a nesoudělné s n;

(b) C_k má ze všech takovýchto booleovských obvodů splňujících (a) minimální počet vnitřních vrcholů (bran).

Pak, pro dostatečně veliké k, je počet vnitřních vrcholů obvodu C_k alespoň $Q(k)$.

Na základě výsledků získaných Goldwasserem a Micalim vyvstává přirozená otázka: jak bezpečné jsou bity podstatně praktičtějších šifrovacích schémat jako jsou RSA a Rabinova schémata s veřejným klíčem?

Goldwasser, Micali a Tong dokázali, že specifické bity jak RSA tak Rabinova schématu byly bezpečné v tom smyslu, že nalezení nejmenšího signifikantního bitu otevřeného textu je ekvivalentní narušení celého RSA systému. Tento výsledek byl zesílen Ben-Orem, Chorem a Shamirem v roce 1983, kde dokázali, že každá metoda, která určí nejméně podstatný bit musí buď mít pravděpodobnost chyby $\frac{1}{4} - \varepsilon$ nebo musí být schopná prolomit zcela RSA-algoritmus. Alexi, Chor, Goldreich a Schnorr v roce 1984 ukázali, že získání jakékoliv výhody ve zjištění nejméně podstatného bitu je ekvivalentní rozluštění celé zprávy.

Goldwasser-Micaliho schéma lze přirozeným způsobem zobecnit. Poznamenejme, že *predikátem s vlastností padacích dveří* je booleovská funkce $B : \{0,1\}^* \to \{0,1\}$ taková, že je snadné v očekávaném polynomiálním čase za vstup délky k a výstup v vybrat náhodně x tak, že $B(x) = v$ a $|x| \leq k$; a žádný nepřítel pracující v polynomiálním čase nemůže, pro náhodně vybrané $x \in X$ tak, že $|x| \leq k$, nemůže spočítat $B(x)$ s pravděpodobností větší než $\frac{1}{2} + \frac{1}{k^c}$, pro všechna $c > 0$ a pro dostatečně velké k; je-li však známa informace o padacích dveřích, lze snadno spočítat $B(x)$.

Buď dále B predikát s vlastností padacích dveří a k buď bezpečnostní parametr systému. Veřejný klíč uživatele A obsahuje popis B a jeho tajný klíč obsahuje informaci o padacích dveřích. Nyní může uživatel B odeslat uživateli A soukromou zprávu $M = m_1 m_2 \dots m_k$ (to je M v binární notaci) následovně: pro $i = 1, \dots, k$ B

1. náhodně vybere binární řetězec x_i délky nejvýše k tak, že $B(x_i) = m_i$;

2. odešle x_i k A.

Pro dešifrování zprávy pak uživatel A, který zná informaci o padacích dveřích, vypočte $B(x_i) = m_i$ pro všechna $i = 1, \ldots, k$.

Platí pak následující věta.

Věta 9.2.1 Goldwasser-Micali *Existuje-li predikát s vlastností padacích dveří, je výše uvedené pravděpodobnostní šifrovací schéma s veřejným klíčem bezpečné v polynomiální době.*

9.3 Kryptograficky bezpečná pseudonáhodná čísla

Problém nalezení „nepředpověditelných“ pseudonáhodných čísel je těsně svázán s náhodným zašifrováním. Je-li dána metoda pro generování takovýchto posloupností, lze ji užít jako zdroj přibližného one-time pad systému, který je bezpečný, zatímco lineární posouvací registry bezpečné nejsou.

Bezpečné pravděpodobnostní šifrovací schéma založené na myšlenkách Goldwassera a Micaliho, ale podstatně efektivněji implementovatelné, je schéma Bluma a Goldwassera (1985). Základní idea schématu je zašifrování zprávy tím, že ji sečteme modulo 2 s nepředpověditelnou pseudonáhodnou posloupností.

Předpokládejme, že A a B si přejí tajně komunikovat a proto sdílí společný generátor náhodných čísel a společnou tajnou informaci s. Aby mohl odeslat A blok otevřeného textu $M_1 M_2 \ldots$ k uživateli B, odesílatel A použije s pro vygenerování posloupnosti pseudonáhodných čísel $X_1, X_2, \ldots,$ a vyvoří kryptogram $C = C_1 C_2 \ldots$, kde

$$C_i = M_i + X_i \pmod 2. \tag{9.3}$$

Nebezpečí pro tento systém spočívá v tom, že nepřítel může mít nějako dodatečnou informaci, která mu umožní najít některé X_i. Na základě těchto znalostí by pak mohl být schopný vygenerovat zbytek posloupnosti a tedy prolomit kryptogram.

Příklad 9.3.1 Jedna z nejpopulárnějších a nejrychlejších metod získání pseudonáhodných posloupností je použití metody *lineárních kongruenčních generátorů*. Ty sestávají z *modulu* N, *násobku* a nesoudělným s N a *přírůstku* c. Začneme z náhodné tajné informace X_1. Posloupnost $(X_n : 1 \leq n < \infty)$ je určena vztahem

$$X_{n+1} = aX_n + c \pmod N. \tag{9.4}$$

Jsou tedy všechna X_i přirozená čísla mezi 0 a $N-1$. Lze dokázat, že vytvořené posloupnosti splňují mnoho různých testů náhodnosti pro vhodný výběr parametrů a, c a N. Proto je tato *lineární kongruenční metoda* široce používána.

Tato metoda není však kryptogragraficky bezpečná. Ze znalosti všech bitů několika po sobě jdoucích X_k lze odvodit parametry a, c a N. Lze dokonce dokázat, že zbytek posloupnosti je téměř všude předpověditelný v polynomiálním čase za předpokladu, že jsou dány alespoň $\frac{2}{3}$ vedoucích bitů prvních málo čísel.

Neformální popis *kryptograficky bezpečného* generátoru náhodných čísel (RNG) je následující. Začneme s tajnou informací s s n pravdivými náhodnými bity. Generátor pak s musí použít, aby vytvořil v čase polynomiálním v n, posloupnost bitů

$$X_1, X_2, \ldots, X_{n^k}$$

s vlastností, že pro daný generátor a prvních t bitů X_i, ale bez možnosti tajné informace s, je výpočetně nemožné předpovědět X_{t+1} s pravděpodobností větší než $\frac{1}{2}$. O takovémto generátoru řekneme, že prošel *testem následujícího bitu.*

Blum a Micali vytvořili první metodu pro tvorbu dokazatelně bezpečných generátorů pseudonáhodných bitových posloupností založenou na použití jednocestných predikátů. Nechť tedy D značí konečnou množinu, $f : D \to D$ je permutace na D a nechť B je funkce z D do $\{0, 1\}$ taková, že

- je-li dáno $x = f^{-1}(y)$, lze snadno vypočíst $B(y)$,
- je-li dáno pouze y, lze $B(y)$ obtížně vypočítat.

Je-li dána tajná informace $x_0 \in D$, můžeme vytvořit posloupnost $x_0, x_1, \ldots, x_n$ použitím rekurentního předpisu $x_{i+1} = f(x_i)$. Abychom vytvořili binární posloupnost $b_0, \ldots, b_{n-1}$ délky n, definujme $b_i = B(x_{n-i})$. Uvědomme si obrácené uspořádání vzhledem k posloupnosti $x_0, x_1, \ldots, x_n$; musíme nejprve vypočíst všechna $x_0, x_1, \ldots, x_n$ a teprve pak spočítat b_0 z x_n, $\ldots$, b_{n-1} z x_1.

Ukažme, že tento generátor splňuje test následujícího bitu. Předpokládejme opak. Pak získáme spor tím, že ukážeme, jak lze spočítat $B(y)$ z y. Protože f je permutace, existuje x_0 tak, že $y = x_{n-i-1}$ v posloupnosti generované z x_0. Pro dané $y = x_{n-i-1}$ můžeme spočítat x_{n-i},$x_{n-i+1}, \ldots, x_n$ pomocí f a pak vypočteme $b_0, \ldots, b_i$ z y. Můžeme-li pak určit efektivně $b_{i+1} = B(x_{n-i-1}) = B(y)$, získáme spor s tím, že je obtížné spočítat $B(y)$ pouze z y.

V roce 1982, Blum a Micali navrhli generátor splňující test následujícího bitu za předpokladu, že problém vypočtení diskrétního algoritmu je „obtížný“ v tom smyslu, že jakmile budou vstupy dostatečně velké, každá efektivní procedura pro výpočet diskrétního logaritmu by nefungovala alespoň pro jistou pevnou část vstupů. Definujme $B(x) = B_{g,p}(x) = 1$, pokud $\log_g x \bmod p \leq \frac{p-1}{2}$ a $B(x) = B_{g,p}(x) = 0$ jinak, kde p je prvočíslo, g je generátor multiplikativní grupy $\mathbf{Z}_p^*$ a $x \in \mathbf{Z}_p^*$. Dále klademe $f(x) = f_{g,p}(x) = g^x \bmod p$. Je-li výpočet diskrétního logaritmu skutečně obtížný, budou vytvořené posloupnosti nepředpovídatelné. Ačkoliv je Blum-Micaliho generátor mezník v historii problému, nelze jej snadno používat či popsat a navíc byl předstihnut jinými praktičtějšími a efektivnějšími systémy.

L. Blum, M. Blum and Shub (1983) navrhli tzv. $x^2 \bmod N$ *generátor*, který lze jednodušeji implementovat a lze dokázat, že je bezpečný vzhledem k testu následujícího bitu za předpokladu, že problém kvadratického zbytku je obtížný.

Tento generátor funguje stejně jako obecná Blum-Micaliho metoda s volbou $f(x) = x^2 \bmod N$ a $B(x) = 1$, pokud je x liché, a $B(x) = 0$ jinak. Přitom počáteční tajná informace x_0 je kvadratický zbytek modulo N, N je obvykle součin dvou prvočísel stejné délky, přičemž každé z těchto prvočísel je kongruentní se 3 modulo 4. Alexi, Chor, Goldreich a Schnorr (1984) dokázali, že předpoklad, že problém kvadratického zbytku těžký lze nahradit předpokladem, že faktorizování je těžké.

Zaveďme nyní pojem *statistického testu v polynomiálním čase*. Buď G generátor pseudonáhodných čísel, který roztáhne $k-$bitovou vstupní tajnou informaci do $p(k)-$bitové posloupnosti, kde p je nějaký pevný polynom. Buď dále S_k množina všech $p(k)-$bitových posloupností vytvořených generátorem G z $k-$bitových vstupních tajných informací. *Statistický test v polynomiálním čase* je náhodný algoritmus $\mathcal{A}$ který se, pro daný vstupní řetězec délky n, zastaví v polynomiálním čase v n a dá na výstup 0 nebo 1. Generátor G *projde testem* $\mathcal{A}$, jestliže, pro každé přirozené číslo t a všechna dostatečně velká k, máme

$$|P_k^S - P_k^R| < \frac{1}{k^t}, \tag{9.5}$$

kde P_k^S je pravděpodobnost, že algoritmus $\mathcal{A}$ dá na výstup 1 pro náhodně vybraný řetězec z S_k a P_k^R je pravděpodobnost, že algoritmus $\mathcal{A}$ dá na výstup 1 pro skutečně náhodně vybraný řetězec téže délky.

Řekneme, že generátor G je *kryptograficky bezpečný*, jestliže projde všemi statistickými testy v polynomiálním čase. Jinak řečeno, generátor pseudonáhodných čísel je kryptograficky bezpečný, jestliže neexistuje žádný efektivní algoritmus, který je schopen v polynomiálním čase oddělit výstup generátoru G od výstupu vytvořeném opravdu náhodnou posloupností.

Hlavní výsledek získaný Yaem (1982) je následující. Přeformulujme definici *testu následujícího bitu*. Rozumíme jím náhodný polynomiální algoritmus $\mathcal{T}$, který se vstupy k a prvními i bity řetězce s náhodně vybraného z S_k nám dá na výstup bit b. Buď p_k^s pravděpodobnost, že bit b je roven $(i+1)-$nímu bitu s. Řekneme, že generátor G *projde testem následujícího bitu* $\mathcal{T}$, jestliže pro každé $t > 0$ a dostatečně velká k platí

$$|p_k^S - \frac{1}{2}| < \frac{1}{k^t}. \tag{9.6}$$

Věta 9.3.2 *Generátor G projde všemi statistickými testy v polynomiálním čase tehdy a jen tehdy, když projde všemi testy následujícího bitu v polynomiálním čase.*

Zůstává otázka existence takovýchto kryptograficky bezpečných generátorů. Doposud však závisí na předpokladu obtížnosti nějakého problému z teorie čísel.

9.4 Wynerův kanál

Kódování a kryptografie

Uvažme následující kryptosystém $(\mathbf{K}, \mathbf{M}, \mathbf{C}, \mathbf{T})$, kde

- ◇ prostor klíčů $\mathbf{K}$ se bude skládat ze všech matic G typu $k \times n$ tak, že

$$G = \begin{pmatrix} 1 & 0 & \dots & & & \dots & 0 & 0 & g_{1,k+1} & \dots & \dots & g_{1,n} \\ 0 & 1 & \dots & & & \dots & 0 & 0 & g_{2,k+1} & \dots & \dots & g_{2,n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & v\dots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & & & \dots & 1 & 0 & g_{k-1,k+1} & \dots & \dots & g_{k-1,n} \\ 0 & 0 & \dots & & & \dots & 0 & 1 & g_{k,k+1} & \dots & \dots & g_{k,n} \end{pmatrix},$$

  spolu s kontrolní maticí H typu $k \times n$ tak, že $G \cdot H^\top = 0$ a

$$H = \begin{pmatrix} h_{1,1} & \dots & \dots & h_{1,k} & 1 & 0 & \dots & & & \dots & 0 & 0 \\ h_{2,1} & \dots & \dots & h_{2,k} & 0 & 1 & \dots & & & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & v\dots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ h_{k-1,1} & \dots & \dots & h_{k-1,k} & 0 & 0 & \dots & & & \dots & 1 & 0 \\ h_{k,1} & \dots & \dots & h_{k,k} & 0 & 0 & \dots & & & \dots & 0 & 1 \end{pmatrix},$$

- ◇ prostor zpráv $\mathbf{M} = \{\mathbf{s}_0, \dots, \mathbf{s}^k\}$ je totožný s množinou syndromů,
- ◇ prostor $\mathbf{C}$ zašifrovaných textů je množina všech n-tic $\mathbf{V}_n$,
- ◇ zobrazení $\mathbf{T} : \mathbf{K} \times \mathbf{M} \to \mathbf{C}$ je definované následovně: $\mathbf{T}(G, \mathbf{s}^i) = \mathbf{c} = (c_1, \dots, c_n)$, kde $\mathbf{c}$ je libovolný vektor z příslušné třídy rozkladu, tj. množiny $\{\mathbf{v} : \mathbf{v} \cdot H^\top = \mathbf{s}\}$.

Adresát obdrží nezkreslenou informaci, protože jejich spojení je bez šumu. Pomocí vektoru $\mathbf{c}$ vypočítá syndromový vektor $\mathbf{s}^i = \mathbf{c} \cdot H^\top$. Funkce $\mathbf{T}$ nabývá svých hodnot náhodně. Proto se v tomto případě mluví o nedeterministickém resp. **náhodném šifrovacím systému**.

Přesněji řečeno, náhodný šifrovací systém $\prod$ je podmnožina kartézského součinu $\mathbf{M} \times \mathbf{K} \times \mathbf{C}$ tak, že pro každý klíč $K \in \mathbf{K}$ a každý zašifrovaný text $C \in \mathbf{C}$ existuje nejvíc jedna zpráva $M \in \mathbf{M}$ tak, že $(M, K, C) \in \prod$. Zároveň pro každé $M \in \mathbf{M}$ a každé $K \in \mathbf{K}$ existuje aspoň jeden kryptogram $C \in \mathbf{C}$ tak, že $(M, K, C) \in \prod$.

Zejména platí následující tvrzení

Tvrzení 9.4.1 *V syndromovém kryptosystému je přenosový poměr* $R \leq \frac{m}{m+1}$.

Proof. Předpokládejme kvůli jednoduchosti, že prostor zpráv M obsahuje 2^m prvků. To znamená, že nejúspornější rovnoměrný kód bude používat $m-$tice $\mathbf{u} = (u_1, \ldots, u_m)$. Pokud použijeme (n, l)-grupový lineární kód, tj. jeho příslušná matice G je typu $l \times n$ a jeho kontrolní matice H je typu $(n-l) \times n$, pak syndromový vektor $\mathbf{s} = \mathbf{v} \cdot H^\top$ bude typu $1 \times (n-l) = 1 \times m$. Proto přenosový poměr bude

$$R = \frac{n-l}{n} = \frac{m}{n} = \frac{m}{m+l} \leq \frac{m}{m+1}.$$

Při rostoucím l bude $\lim R = 0$. V případě, že počet zpráv nebude mocninou čísla 2, použije ohraničení

$$2^m \leq |M| < 2^{m+1}$$

a postup důkazu zopakujeme.

Podívejme se nyní na naši situaci z pohledu nepřítele. Počet tříd rozkladu s rostoucím l bude menší a výrazně bude narůstat počet možných slov, která můžeme vyslat pro jedno konkrétní zdrojové slovo tj. v každé třídě rozkladu bude 2^l vektorů. Navíc vlivem poruch v binárním symetrickém kanále bude vektor $\mathbf{z}$, který zachytí nepřítel různý od kryptogramu, který obdrží povolaný příjemce. Nepřítel tedy bude neustále maten. Za to ale zaplatíme rychlostí přenosu, neboť na přenos m-bitové zprávy budeme muset vyslat $n = m + l$-bitovou šifru.

Připomeňme si v dalším zhruba obsah fundamentální práce A.D. Wynera, ve kterém jsou spojeny problémy kryptografie, kódování a teorie informace do opravdu atraktivního a praktického tvaru. Zdroj $\mathcal{S}$ vysílá informaci konstantní rychlostí a tato se přenáší pokud možno bez chyby k příjemci, zatímco nechává odposloucháváče bez povšimnutí. Nejprve předpokládejme speciální situaci, kdy hlavní kanál je binární kanál bez paměti a bez šumu a odposlouchávací kanál je binární symetrický kanál s pravděpodobností chyby $p < 1$.

První řešení tohoto problému je velmi jednoduché:

Šifrovací algoritmus

Zašifrujte 0 jakožto náhodný binární řetězec délky N obsahující sudý počet jedniček. Zašifrujte 1 jakožto náhodný binární řetězec délky N obsahující lichý počet jedniček.

Je jasné, že dešifrovací proces je snadný, příjemce musí pouze zjistit paritu každého bloku délky N. Přitom nejsou žádné požadavky na paměť či CPU jednotku.

Případný odposloucháváč bude však více než zmaten. Totiž, připomeňme, že $p < 1$ je pravděpodobnost toho, že je špatně přenesen přes odposlouchávací kanál. Pak pravděpodobnost P_c správně obdrženého symbolu odposloucháváčem je určena vztahem

$$\begin{aligned} P_c &= P[\text{nastane právě sudý počet chyb v } N \text{ symbolech}] \\ &= q^N + \binom{N}{2} q^{N-2}p^2 + \dots \qquad (q = 1-p) \qquad (9.7) \\ &= \tfrac{1}{2}[(q+p)^N + (q-p)^N] = \tfrac{1}{2}[1 + (1-2p)^N]. \end{aligned}$$

Zejména tedy $P_c \simeq \frac{1}{2}$ pro velká N. Evidentně pak při přenosu binární zprávy zachytí odposlouchávač text téměř zcela plný poruch a šumů. Za toto pak ale platíme velkou cenu – totiž přenosový poměr, což je počet bitů původního textu, které přeneseme jedním bitem šifry, se nám redukoval na číslo $\frac{1}{N}$.

Předpokládejme místo toho, že jsme rozdělili proudový zdroj do bloků $\mathbf{S}$ délky k a že použijeme kód na opravování chyb, abychom zakódovali každý takovýto blok n-bitovým vektorem $\mathbf{X}$ tak, že přenosová rychlost R kódu je $\frac{k}{n}$. Je-li $\mathbf{Z}$ n-bitová posloupnost zachycená odposlouchávačem, označíme odposlouchávačovu ekvivokaci $H(\mathbf{X}|\mathbf{Z})$ a tedy

$$d = \frac{H(\mathbf{X}|\mathbf{Z})}{k}$$

je odposlouchávačova ekvivokace na 1 bit vysílané zprávy neboli míra dvojsmyslnosti připadající na jeden bit vysílané zprávy.

Z teorie informace víme, že $H(\mathbf{X}|\mathbf{Z}) = H(\mathbf{S}|\mathbf{Z}) \leq H(\mathbf{S}) \leq k$ a tedy $d \leq 1$. Speciálně, pokud odposlouchávač pozná klíč, pomocí něhož se vysílá, tj. hodnotami náhodného vektoru $\mathbf{Z}$ je jednoznačně určena hodnota náhodného vektoru $\mathbf{S}$, pak je $H(\mathbf{S}|\mathbf{Z}) = 0$ a $d = 0$. Naopak, pokud odposlouchávač nepostřehne žádnou závislost mezi $\mathbf{S}$ a $\mathbf{Z}$, tj. zprávy $\mathbf{S}$ a $\mathbf{Z}$ jsou nezávislé, je pak $d = \dfrac{H(\mathbf{S})}{k}$ Jsou-li navíc všechny vysílané zprávy stejně pravděpodobné, tj. $P(\mathbf{S} = (s_1, \dots, s_k)) = 2^{-k}$ pro každou zprávu $(s_1, \dots, s_k)$, je $H(\mathbf{S}) = k$, $d = 1$.

Cílem je navrhnout takový šifrovací systém, který by měl maximální míru dvojsmyslnosti tj. zajistit věci tak, aby odposlouchávač nezískal žádné množství informací.

Evidentně, parametry R a d působí proti sobě tj. pokud R roste, d klesá a obráceně. Wyner dokázal analogie Shannonovy věty o kódování se šumem. Řekneme, že dvojice (R, d) je *dosáhnutelná*, jestliže, pro všechna $\varepsilon > 0$, existuje kódovací a dekódovací algoritmus s parametry n a k tak, že

$$\frac{k}{n} \geq R - \varepsilon, \quad H(\mathbf{X}|\mathbf{Z}) \geq \frac{d-\varepsilon}{k}, \quad P_e \leq \varepsilon, \qquad (9.8)$$

kde P_e je rychlost chybového dekódování na bit u legitimního příjemce.

Dále platí

$$\begin{aligned} H(\mathbf{S}|\mathbf{Z}) &= H(\mathbf{S},\mathbf{Z}) - H(\mathbf{Z}) \\ &= H(\mathbf{S},\mathbf{C},\mathbf{Z}) - H(\mathbf{C}|\mathbf{S},\mathbf{Z}) - H(\mathbf{Z}) \\ &= H(\mathbf{Z}|\mathbf{S},\mathbf{C}) + H(\mathbf{S},\mathbf{C}) - H(\mathbf{C}|\mathbf{S},\mathbf{Z}) - H(\mathbf{Z}) \qquad (9.9) \\ &= H(\mathbf{Z}|\mathbf{S},\mathbf{C}) + H(\mathbf{S}|\mathbf{C}) - H(\mathbf{Z}) + H(C) - H(\mathbf{C}|\mathbf{S},\mathbf{Z}) \\ &= H(\mathbf{Z}|\mathbf{S},\mathbf{C}) + H(\mathbf{S}|\mathbf{C}) + [H(C) - H(\mathbf{Z})] - H(\mathbf{C}|\mathbf{S},\mathbf{Z}). \end{aligned}$$

Přitom náhodný vektor $\mathbf{C}$ modeluje zašifrovanou zprávu, tj. $T(G, \mathbf{s}) = \mathbf{c}$, která je vyslaná při použití syndromového vektoru $\mathbf{s}$. Pokud víme hodnotu $\mathbf{C} = \mathbf{c}$, pak známe i hodnotu vektoru $\mathbf{S} = \mathbf{s}$ a naopak. Proto pak máme $H(\mathbf{Z}|\mathbf{C}, \mathbf{S}) = H(\mathbf{Z}|\mathbf{C})$. Abychom mohli 9.9 upravit, musíme udělat doplňující předpoklady o binárním symetrickém kanálu, přes který nepřítel odposlouchává přenášené zprávy.

Jak vyslanou šifru $\mathbf{C}$ tak odposlechnutou zprávu $\mathbf{Z}$ můžeme považovat za náhodné vektory $\mathbf{C} = (C_1, \ldots, C_n)$, resp. $\mathbf{Z} = (Z_1, \ldots, Z_n)$. Přitom náhodné veličiny C_i, resp. Z_i nabývají hodnot 0 a 1 s příslušnými pravděpodobnostmi. Připomeňme, že binární symetrický kanál se nazývá *bez paměti*, jestliže

$$P(\mathbf{Z} = \mathbf{z}|\mathbf{C} = \mathbf{c}) = \prod_{i=1}^{n} P(Z_i = z_i|C_i = c_i). \tag{9.10}$$

Označme $h(x) = -x \log x - (1-x) \log (1-x)$. Připomeňme, že pravděpodobnost vstupu znaku 1 do binárního symetrického kanálu bez paměti je $P(C_i = 1) = p' = 1 - P(C_i = 0)$ a chybovost kanálu, tj. $P(Z_i = 1|C_i = 0) = P(Z_i = 0|C_i = 1) = \varepsilon$. Pak entropie vstupu bude $H(C_i) = h(p')$ a entropie zachycených zpráv bude $H(Z_i) = h(p' + p - 2pp')$. Dále bude platit pro podmíněné entropie $H(Z_i|C_i = 0) = H(Z|C_i = 1) = H(Z_i|C_i) = h(p)$. Odtud pak

$$H(\mathbf{Z}|\mathbf{C}) = \sum_{i=1}^{n} H(Z_i|C_i) = n \cdot h(p). \tag{9.11}$$

Zároveň lze z teorie informace dokázat, že entropie šifrového vektoru $\mathbf{C}$ je menší nebo rovna entropii odposlechnutého vektoru $\mathbf{Z}$. Protože vždy $H(\mathbf{C}|\mathbf{S}, \mathbf{Z}) \geq 0$ a $H(\mathbf{S}|\mathbf{C}) = 0$, je

$$H(\mathbf{S}|\mathbf{Z}) \leq H(\mathbf{Z}|\mathbf{C}) = n \cdot h(p). \tag{9.12}$$

Zejména tedy pro příslušnou míru dvojsmyslnosti obdržíme

$$k \cdot d \leq n \cdot h(p) \text{ tj. } R \cdot d \leq h(p). \tag{9.13}$$

Dokázali jsme tedy následující tvrzení:

Věta 9.4.2 *Při použití syndromového kryptosystému o přenosovém poměru R, míře dvojsmyslnosti d a chybovosti symetrického binárního kanálu bez paměti, přes který nepřítel odposlouchává zprávy platí nerovnost $R \cdot d \leq h(p)$. V případě, že přenos zpráv mezi odesílatelem a adresátem není bezšumový a pravděpodobnost chyby na jeden vyslaný znak je $P_p = \frac{1}{k} P(\mathbf{S} \neq \mathbf{S'})$, je $R \cdot [d - h(P_p)] \leq h(p)$.*

Výše uvedené tvrzení je pak ekvivalentní s tvrzením:

Věta 9.4.3 *Je-li hlavní kanál bez šumu a odposloucháváčův binární symetrický kanál má pravděpodobnost chyby p, je dvojice (R, d) dosažitelná, $0 < R, d < 1$, právě tehdy, když $R \cdot d \leq h(p)$.*

Popišme dále obecnější Wynerův výsledek. Naším úkolem je maximalizace přenosu zpráv k legitimnímu příjemci a zároveň udržení nejistoty odposlouchávače nad jistou hodnotou.

Pomocí entropie to lze vyjádřit následovně: maximalizovat

$$I(\mathbf{X}|\mathbf{Y}) = H(\mathbf{X}) - H(\mathbf{X}|\mathbf{Y})$$

a zároveň minimalizovat informaci

$$I(\mathbf{X}|\mathbf{Z}) = H(\mathbf{X}) - H(\mathbf{X}|\mathbf{Z}).$$

Pro každý poměr $R > 0$ definujme $\rho(R)$ jakožto množinu těch rozdělení $\mathbf{p}$ pravděpodobnosti na vstupním vektoru $\mathbf{X}$ hlavního kanálu tak, že

$$I(\mathbf{X}|\mathbf{Y}) \geq R.$$

Buď C_M buď kapacita hlavního kanálu a nechť

$$\Gamma(R) = \sup_{\mathrm{p}\in\rho(R)}[I(\mathbf{X}|\mathbf{Y}) - I(\mathbf{X}|\mathbf{Z})].$$

Pak platí

Věta 9.4.4 *Množina $\mathcal{R}^*$ všech dosažitelných dvojic (R, d) je určena vztahem*

$$\mathcal{R}^* = \{(R, d) : 0 \leq R \leq C_M, 0 \leq d \leq 1, R \cdot d \leq \Gamma(R)\}.$$

9.5 Pravděpodobnostní podpisovací schémata

Pravděpodobnostní techniky byly rovněž aplikovány na vytvoření digitálních podpisů. Tento přístup byl započat Goldwasserem, Micalim a A.C. Yaoem (1984), kteří představili podpisovací schéma založené na obtížnosti faktorizace a na obtížnosti invertibility RSA funkce, což jsou dokazatelně obtížné problémy pro útok s použitím známého podpisu a to pro nepřítele, který uspěje s paděláním podpisu jedné zprávy, ale ne dle jeho výběru. Goldwasser, Micali a Rivest zesílili tento výsledek navržením podpisovacího schénatu, které je nenapadnutelné výše uvedeným způsobem. Toto schéma je založeno na obtížnosti faktorizování (daleko obecněji na existenci tzv. *permutací s vlastností padacích dveří* – tj. takové dvojice permutací f_0, f_1 majících společný definiční obor, pro který je obtížné najít x, y tak, že $f_0(x) = f_1(y)$).

Popišme jejich schéma. Buď (f_0, f_1) a (g_0, g_1) dvě dvojice takovýchto permutací. Buď dále $b = b_1 \dots b_k$ binární řetězec a definujme $F_b^{-1}(y) = f_{b_k}^{-1}(\dots(f_{b_{k_2}}^{-1}((f_{b_{k_1}}^{-1}(y))))$ a $G_b^{-1}(y) = g_{b_k}^{-1}(\dots(g_{b_{k_2}}^{-1}((g_{b_{k_1}}^{-1}(y))))$.

Podpisující zveřejní (x, f_0, f_1, g_0, g_1), kde x je náhodně vybraný prvek z definičního oboru f_0, f_1 a utají informaci typu padacích dveří dovolující mu vypočíst F_b^{-1} a G_b^{-1}. Zároveň si uchová obsah proměnné *history* (ne nutně utajeno). Pro jednoduchost budeme předpokládat bezprefixový prostor zpráv.

Podpis $i-$té zprávy m_i se vytvoří následovně. Podpisující

1. vybere náhodně r_i z definičního oboru g_0, g_1 a položí $history = history \cdot r_i$, kde $\cdot$ označuje zřetězení;
2. vypočte $l_i = F^{-1}_{history}(x)$ a $t_i = G^{-1}_{m_i}(r_i)$;
3. vytvoří podpis zprávy m_i jako trojici $(history, l_i, t_i)$.

Abychom zkontrolovali platnost podpisu (h, l, t) zprávy m, každý s přístupem k veřejnému klíči podpisujícího může prověřit, že

1. $F_h(l) = x$, kde x je uloženo ve veřejně přístupném souboru;
2. $G_m(t) = r$, kde r je suffix h.

Pokud obě podmínky platí, je podpis platný.

Výše popsané schéma, ačkoliv teoreticky velmi atraktivní, je zcela neefektivní. Lze ho však modifikovat do daleko kompaktnějších podpisů bez použití paměti pro jiné věci než pro veřejné a tajné klíče.

Příloha A

Obecná pravidla kryptologie

- nevysílat ten samý text zašifrovaný různými klíči
- zabránit používání stereotypních a velmi pravděpodobných slov a frází
- omezit používání typických kombinací písmen, interpunkčních znamének, mezer mezi slovy

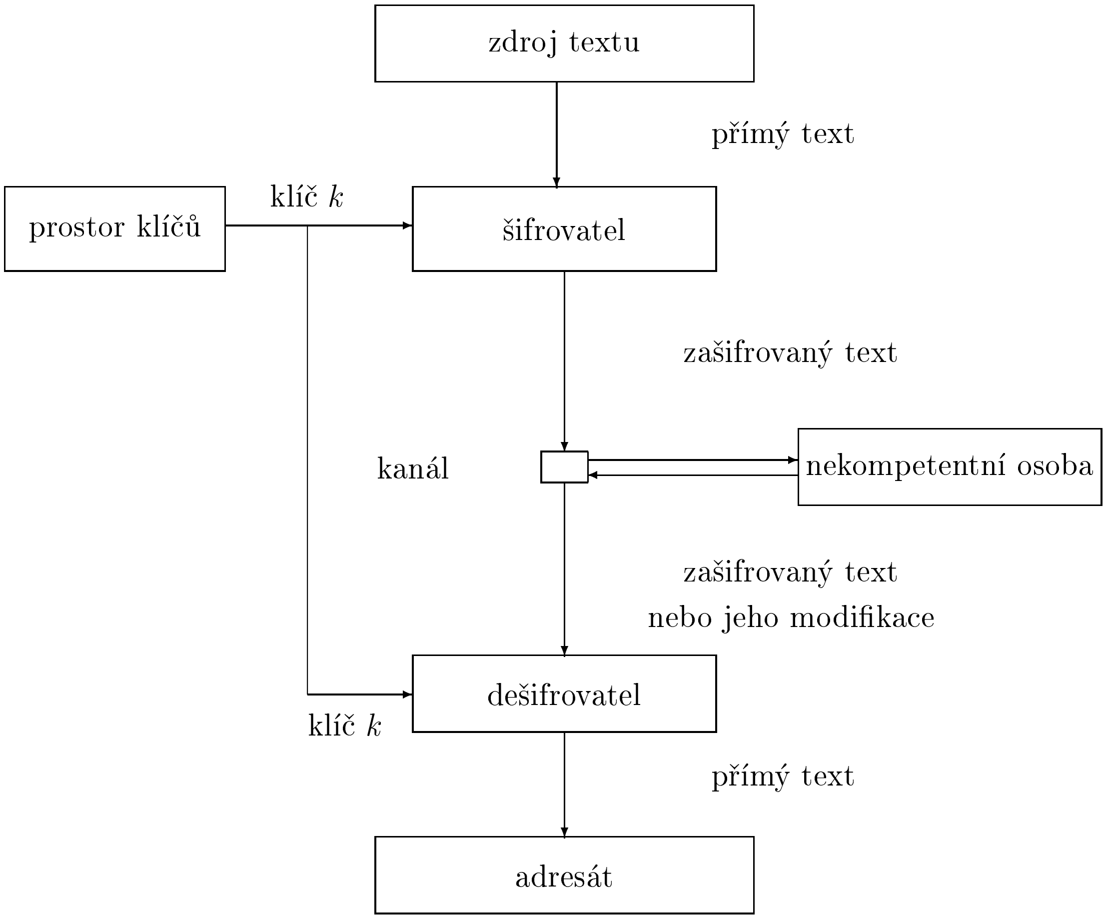
Kritéria posouzení kvality šifrovacího systému:

- spolehlivost - ve smyslu odolnosti proti rozluštění
- objem klíče: Vzhledem k tomu, že se klíč (v klasických systémech) musí mezi komunikujícím stranami vyměnit, je snaha, aby byl co „nejmenší“.
- složitost operace šifrování a dešifrování: Míra složitosti je relativní a její hodnocení závisí v prvé řadě na tom, za jakých okolností šifrujeme a jaký stupeň bezpečnosti přitom chceme zajistit.
- šíření chyb: Existují šifrovací metody, ve kterých šifrování znaku a nebo skupiny znaků závisí na zašifrování některých předcházejících znaků. V těchto systémech může mít potom chyba vážné důsledky při dešifrování textu.
- zvětšení zašifrovaného textu: Existují šifrovací systémy, ve kterých se text po zašifrování prodlouží. Jedním z motivů pro takovou metodu může být cíl porušit statistickou charakteristiku zašifrovaného textu. Prodlužování textu je však na druhé straně nežádoucí z jiných důvodů, například z důvodu omezené propustnosti přenosového kanálu.

Rozhodující faktory při výběru šifry:

- počet a délka vysílaných zpráv

- doba aktuálnosti zprávy
- odhad možností a odbornosti potenciálních zájemců o zprávu
- jednoduchost



Obrázek A.1: Schéma kryptografického systému

Příloha B

Charakteristika českého jazyka

FREKVENCE v %			
písmeno	relativní	písmeno	relativní
A	8.6	N	6.8
B	1.7	O	8.0
C	3.3	P	3.2
D	3.6	Q	0.0
E	10.5	R	4.9
F	0.2	S	6.3
G	0.2	T	5.1
H	2.2	U	4.0
I	7.5	V	4.3
J	2.2	W	0.0
K	3.6	X	0.1
L	4.2	Y	2.8
M	3.5	Z	3.2

Tabulka B.1: Četnosti písmen českého jazyka (mezinárodní abeceda bez mezery)

Nejčetnější české bigramy:

ST TE NE NI EN LE PR AN RA EL VE PO JE OV RO RE NA VA TO TR EC CH ES ET AL

Nejčetnější české trigramy:

PRO IST ANI STA OVE OST OVA EST ADA NEM ENI EHO NIS SKE EDN ATI ENE ITI RAN JEN INE SPO LAD STI ENA PRI AKO AVA DNI INI TER BYL MIN OJE RED UJE BUD PRA AJI EME MAV SKY ABY

- Ačkoli písmeno P tvoří jen něco přes 3% textu, patří bigramy PR a PO k nejčetnějším bigramům.

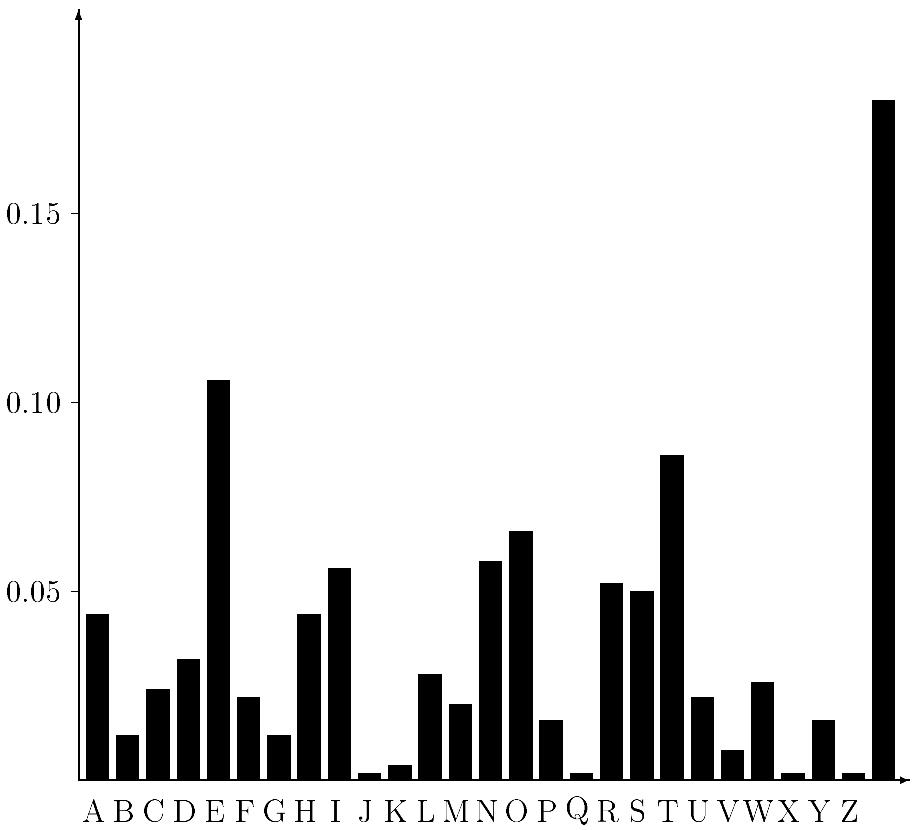
FREKVENCE v %			
písmeno	relativní	písmeno	relativní
A	6.5	O	6.7
B	1.2	P	1.6
C	2.4	Q	0.1
D	3.1	R	5.2
E	10.7	S	5.0
F	2.3	T	8.6
G	1.3	U	2.1
H	4.3	V	0.8
I	5.6	W	1.6
J	0.1	X	0.1
K	0.3	Y	1.6
L	2.8	Z	0.1
M	2.0	-	18.2
N	5.8		

Tabulka B.2: Četnosti písmen českého jazyka (mezinárodní abeceda s mezerou)

- Bigramy ST a PR jsou nejčetnějšími souhláskovými bigramy.
- V českém jazyce se s vyjímkou bigramu OU téměř nevyskytují bigramy samohláskové.
- Protože v mezinárodní abecedě neexistuje písmeno **ch** jako znak, je bigram CH čtvrtým nejčastějším souhláskovým bigramem. Přitom písmeno C zabírá 3.4% textu a písmeno H pouhých 1.9% textu.
- Nejčastější trigramy jsou převážně takové, u nichž se střídají samohlásky a souhlásky. Nejčastějším souhláskovým trigramem je STR.

FREKVENCE v %					
písmeno	relativní	písmeno	relativní	písmeno	relativní
A	6.5	Í	6.7	Š	0.7
Á	1.2	J	1.6	T	3.9
B	2.4	K	0.1	Ť	0.7
C	3.1	L	5.2	U	3.0
Č	10.7	M	5.0	Ú	0.5
D	2.3	N	8.6	V	3.9
Ď	1.3	Ň	2.1	W	0.0
E	4.3	O	0.8	X	0.1
É	5.6	Ó	1.6	Y	1.6
Ě	0.1	P	0.1	Ý	0.8
F	0.3	Q	1.6	Z	1.9
G	2.8	R	0.1	Ž	0.9
H	2.0	Ř	18.2	-	16.2
I	5.8	S	4.0		

Tabulka B.3: Četnosti písmen českého jazyka (úplná abeceda s mezerou)



Obrázek B.1: Histogram četností písmen českého jazyka (s mezerou)

Příloha C

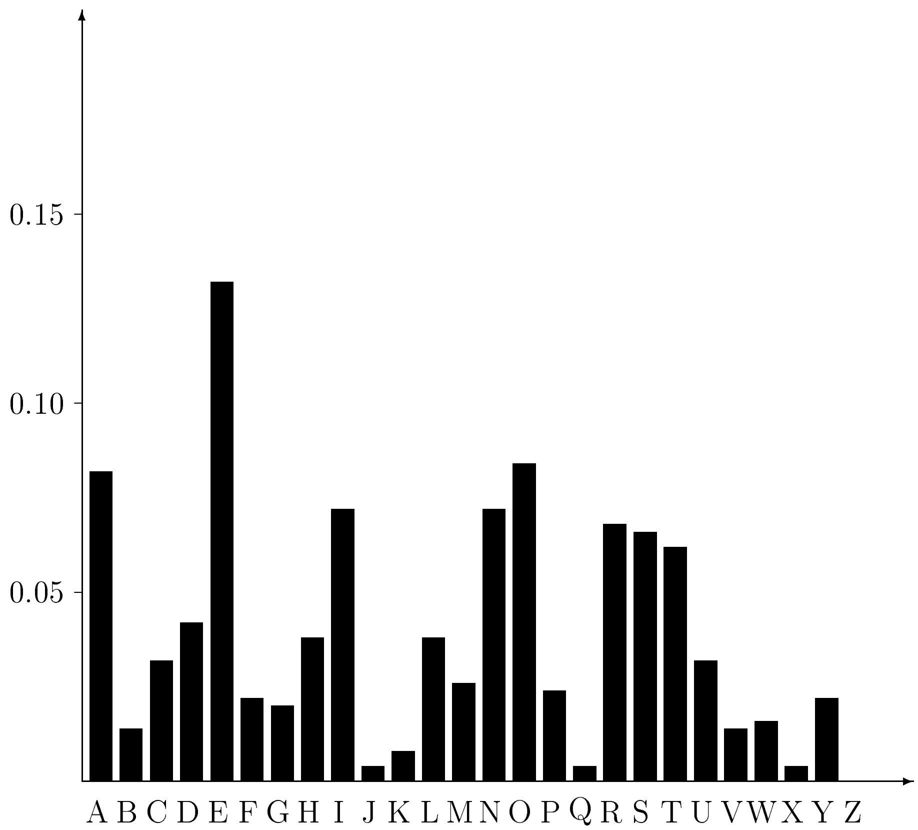
Charakteristika anglického jazyka

FREKVENCE v %			
písmeno	relativní	písmeno	relativní
A	8.1	N	7.2
B	1.5	O	8.4
C	3.1	P	2.4
D	4.2	Q	0.4
E	13.1	R	6.8
F	2.1	S	6.6
G	2.0	T	6.3
H	3.9	U	3.2
I	7.1	V	1.4
J	0.4	W	1.7
K	0.9	X	0.5
L	3.9	Y	2.1
M	2.7	Z	0.0

Tabulka C.1: Četnosti písmen anglického jazyka (bez mezery)

A E I O U Y tvoří 42% textu
C D H L N R S T tvoří 42% textu
B F G J K M P Q V W X Z tvoří 16% textu
Nejčetnější bigramy:
TH ER ON AN RE HE IN ED ND HA AT EN ES OF OR NT EA TI TO IT ST IO LE IS OU AR AS DE RT VE
Nejčetnější trigramy:
THE AND ING ENT ION NTH THA FOR NDE HAS NCE EDT TIS OFT STH MAN

- Nenastává žádné zkreslení vlivem anulování diakritických znamének (narozdíl od češtiny zde každé písmeno představuje pouze samo sebe).
- Téměř vůbec se nevyskytuje písmeno Z. Nejčastějším slovem, kde se písmeno Z vyskytuje, je ZERO.
- Angličtina má nejkratší průměrnou délku slova. Již některé nejčastější trigramy jsou úplnými slovy.



Obrázek C.1: Histogram četností písmen anglického jazyka

Příloha D

Charakteristika německého jazyka

FREKVENCE v %			
písmeno	relativní	písmeno	relativní
A	5.5	N	10.2
B	1.6	O	2.1
C	2.9	P	0.5
D	4.9	Q	0.0
E	19.2	R	7.0
F	2.0	S	7.1
G	3.6	T	5.9
H	5.0	U	4.2
I	8.2	V	0.8
J	0.2	W	1.4
K	1.3	X	0.0
L	3.5	Y	0.0
M	1.7	Z	1.2

Tabulka D.1: Četnosti písmen německého jazyka (bez mezery)

A E I O U Y tvoří 39.2% textu
D N R S T tvoří 35.1% textu
B C F G H L M W Z tvoří 22.9% textu
J K P Q V X tvoří 2.8% textu
Nejčetnější bigramy:
EN ER CH EI DE ND IN GE IE TE ES NE UN RE HE ST SE SC BE NG IC DI NS AN SS UE SI EH NI EG IS HA

Nejčetnější trigramy:
EIN SCH DER ICH NDE CHE DIE INE DEN GEN END UND TEN CHT ENS NGE UNG ERE ERS STE SEI EGE IES IND NER
Nejčetnější tetragramy:
ICHT KEIT HEIT CHON CHEN CHER URCH EICH

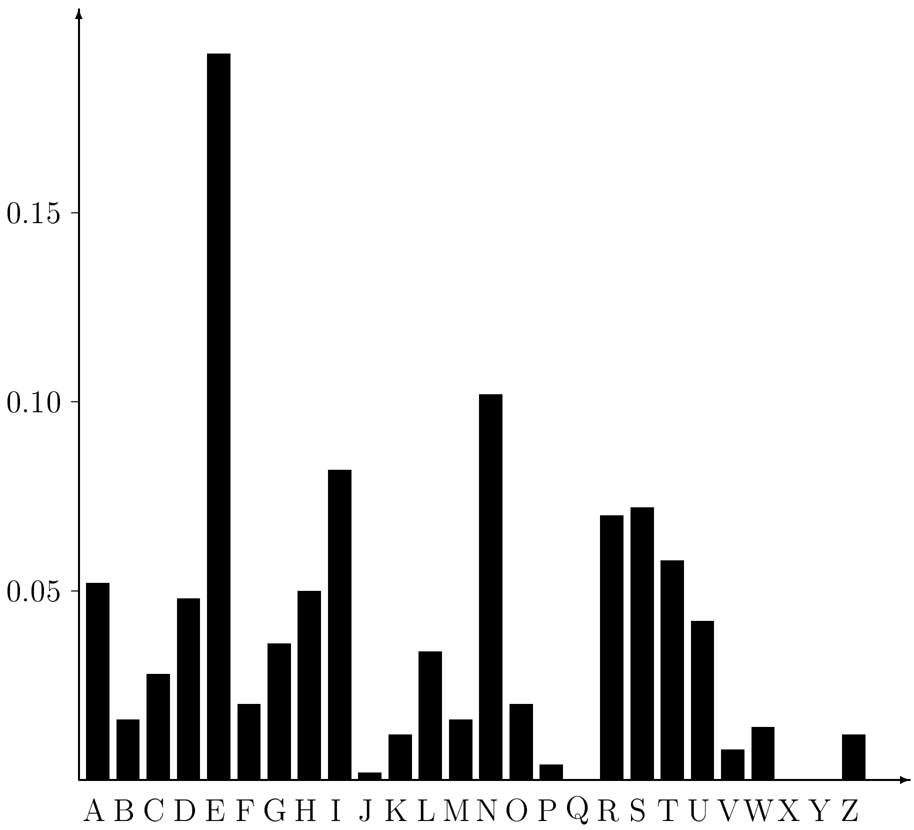
- Nevyskytují se téměř písmena Q, X, Y.
- Charakteristická je vysoká četnost souhlásky N.
- Vysoká četnost samohlásky E je dána také opisen přehlasů písmen A, O, U, které mají tvar AE, OE, UE.
- Téměř všechny nejčetnější tetragramy obsahují CH.

	GBR	BRD	FRA	ITA	ESP	PTG	SLO
A	7.88	5.26	8.54	11.42	12.79	13.50	11.24
B	1.44	2.03	0.91	1.00	1.22	0.50	1.74
C	2.89	2.22	2.98	4.30	4.18	3.50	3.42
D	4.06	4.96	3.49	3.64	5.13	5.00	3.77
E	12.94	18.80	16.80	11.70	13.60	13.00	9.48
F	2.75	1.73	1.01	1.05	0.58	1.00	0.17
G	1.69	3.80	1.07	1.68	1.06	1.00	0.06
H	5.61	4.51	0.71	1.19	1.08	1.00	2.91
I	7.26	8.11	7.81	11.66	6.62	6.00	6.21
J	0.20	0.08	0.54	0.00	0.40	0.50	2.27
K	0.42	1.17	0.00	0.00	0.00	0.00	3.75
L	3.56	3.24	5.62	6.23	5.73	3.50	3.93
M	2.53	2.10	2.98	2.58	2.60	4.50	2.75
N	7.40	10.85	7.38	7.28	6.58	5.50	6.05
O	7.41	2.82	5.24	9.38	9.15	11.50	9.80
P	1.98	0.52	3.05	2.85	2.84	3.00	3.43
Q	0.16	0.01	1.20	0.54	1.35	1.50	0.00
R	6.74	7.01	6.64	6.47	6.60	7.50	4.46
S	6.53	7.04	8.06	4.90	7.62	7.50	6.02
T	9.36	5.43	7.28	6.34	4.13	4.50	4.91
U	2.63	4.61	6.15	3.05	4.31	4.00	3.75
V	1.08	0.92	1.71	1.88	0.87	1.50	4.69
W	1.65	1.44	0.00	0.00	0.00	0.00	0.01
X	0.24	0.00	0.42	0.00	0.10	0.20	0.02
Y	1.52	0.00	0.22	0.00	1.06	0.00	2.43
Z	0.07	1.34	0.19	0.86	0.40	0.30	2.75

Tabulka D.2: Četnosti písmen (v %) několika evropských jazyků (mezinárodní abeceda s mezerou)

jazyk	I
angličtina	0.066895
čeština	0.058258
dánština	0.070731
finština	0.073796
francouzština	0.074604
holandština	0.079805
italština	0.073294
němčina	0.076667
ruština	0.056074
slovenština	0.060270
španělština	0.076613

Tabulka D.3: Závislost indexu koincidence na jazyku



Obrázek D.1: Histogram četností písmen německého jazyka

Literatura

[1] J. Adámek: *Stochastické procesy a teorie informace - úlohy.* ČVUT, Praha 1989.

[2] J. L. Balcázar, J. Díaz, J. Gabarró: *Structural Complexity I.* Springer-Verlag, Heidelberg 1988.

[3] H. Beker and F. Piper: *Cipher Systems. The Protection of Communication.* Northwood, London 1982.

[4] A. Beutelspacher: *Kryptologie.* Vieweg, Braunschweig 1991.

[5] E. Biham and A. Shamir: *Differential Cryptoanalasys of the Data Encryption Standard.* Springer-Verlag, New York 1993.

[6] V. Klíma: *Kódy, komprimace a šifrování.* Chip 2/1993, str. 24-28.

[7] V. Klíma: *Kritika šifrového standardu.* Chip 5/1993 str. 52-58, Chip 6/1993, str. 50-55, 7/1994 str. 50-51.

[8] V. Klíma: *Šifry a kryptéři.* Chip 2/1994, str. 206-207.

[9] V. Klíma: *Utajené komunikace.* Chip 5/1994, Chip 6/1994, str. 184-188, 7/1994 str. 138-141 a 8/1994 str. 118-121.

[10] V. Klíma: *Utajené komunikace.* Chip 2/1995 str. 126-128, Chip 3/1995, str. 136-141, 4/1995 str. 136-138, Chip 5/1995 str. 166-168, Chip 6/1995, str. 174-175, 7/1995 str. 130-131, 8/1995 str. 142-143, 9/1995 str. 206-208, 10/1995 str. 226-228, 11/1995 str. 170-172 a 12/1995 str. 164-166.

[11] J. Kodl, V. Smejkal, T. Sokol: *Smíme šifrovat.* Chip 5/1995, str. 30-32.

[12] J. Kodl, V. Smejkal, T. Sokol: *Šifry, státní zájmy a lidská práva.* Chip 4/1995, str. 34-37.

[13] L. Kučera a J. Nešetřil: *Algebraické metody diskrétní matematiky.* SNTL, Praha 1989.

[14] K. Kukura: *Pretty Good Privacy.* Elektronika 12/1995, str. 18-19.

[15] Š. Porubský a O. Grošek: *Šifrovanie. Algoritmy, Metódy, Prax.* Grada, Praha 1992.

[16] B. Preneel, R. Govaerts and J. Vandewalle: *Information Authentication: Hash Functions and Digital Signatures.* preprint 1993.

[17] B. Preneel: *Standardization of Cryptographic Techniques.* preprint 1993.

[18] A. Salomaa: *Public-Key Cryptography.* Springer, Berlin 1996.

[19] B. Schneier: *Angewandte Kryptographie.* Addison-Wesley, Bonn 1996.

[20] B. Schneier: *Digitální podpisy.* Bajt 11/93, str.79-82.

[21] National Bureau of Standards: *Data Encryption Standard.* U.S. Department of Commerce, FIPS pub. 46, December 1977.

[22] National Bureau of Standards: *Data Encryption Standard.* U.S. Department of Commerce, FIPS pub. 81, December 1977.

[23] D. Welsh: *Codes and Cryptography.* Oxford University Press, New York 1989.