Základy kryptológie

Martin Stanek

verzia 0.16
12. decembra 2004

Obsah

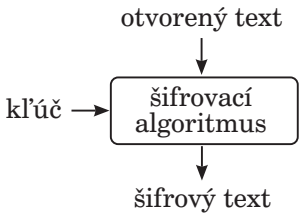
1 Úvod

Kryptológia je vedná oblasť zaoberajúca sa skúmaním bezpečnostných aspektov komunikácie. Pôvodne sa kryptológia venovala najmä metódam ochrany dôvernosti prenášaných informácií (dát), teda šifrovaniu. Rozvoj technológie podnietil rozšírenie poľa pôsobnosti aj na ďalšie bezpečnostné požiadavky (napr. integrita alebo nepopretie autorstva) a aplikačné oblasti (protokoly pre autentifikáciu, digitálne podpisy, elektronické voľby, atď.).

Kryptológia sa delí na dve časti – kryptografiu a kryptoanalýzu. Kryptografia sa zaoberá návrhom konštrukcií (algoritmov, protokolov) napĺňajúcich konkrétne bezpečnostné požiadavky. Úlohou kryptoanalýzy je skúmať možnosti útokov na tieto konštrukcie. Cieľom tejto časti je zoznámiť čitateľa so základnými pojmami v kryptológii.

1.1 Šifrovanie

Šifrovanie si kladie za cieľ transformovať vstupné dáta do podoby, v ktorej sú pre potenciálneho útočníka nezrozumiteľné a nie je schopný rekonštruovať ich pôvodný tvar. Zároveň požadujeme, aby oprávnené subjekty (používatelia) mohli pôvodné dáta rekonštruovať. Vstupné dáta v ich pôvodnej podobe budeme nazývať otvorený text (príp. správa). Proces ich transformácie sa nazýva šifrovanie a je realizované šifrovacím algoritmom (funkciou). Výsledok šifrovania je šifrový text (príp. zašifrovaná správa). Šifrovací algoritmus je parametrizovaný ďalším vstupom – kľúčom, ktorý nezávisí na otvorenom texte. Popisovaná situácia je zobrazená na obrázku 1.1.



Obrázok 1.1: Šifrovanie

Proces inverznej transformácie, keď zo šifrového textu dostaneme opäť pôvodný otvorený text, sa nazýva dešifrovanie a je realizovaný dešifrovacím algoritmom (funkciou). Dešifrovací algoritmus je taktiež parametrizovaný kľúčom.

Popíšme šifrovanie formálnejšie. Nech P, C, K sú konečné množiny:

P – množina otvorených textov,
C – množina šifrových textov,
K – množina kľúčov.

Hovoríme, že funkcia $E : P \times K \to C$ je šifrovacou funkciou, ak k nej existuje funkcia $D : C \times K \to P$ taká, že platí:

$$\forall k \in K \, \forall p \in P : \; D(E(p,k),k) = p.$$

Takúto funkciu D potom nazývame dešifrovacou funkciou k E a dvojica $\langle E, D\rangle$ tvorí šifrovací systém. Vzhľadom na použitie rovnakého kľúča pre šifrovanie a dešifrovanie sa takéto systémy nazývajú aj symetrické. Šifrovanie budeme niekedy namiesto $E(p,k)$ označovať $E_k(p)$ a dešifrovanie budeme namiesto $D(c,k)$ označovať aj $D_k(c)$.

Najdôležitejšou otázkou pri skúmaní šifrovacích systémov je to, ako a za akých podmienok môžu zabezpečiť dôvernosť šifrovaných dát.

Poznamenajme, že popísané šifrovacie systémy sú systémy „klasickej“ kryptografie (systémy s tajným kľúčom), kde sa vyžaduje utajenie kľúča pred útočníkom. Od roku 1976 sa začala rozvíjať aj oblasť skúmajúca šifrovacie systémy s verejným kľúčom. V nich nemusí byť šifrovací kľúč utajený a požiadavka utajenosti sa týka len dešifrovacieho kľúča (z toho vyplýva, že kľúče sú rôzne). Šifrovacími systémami s verejným kľúčom sa budeme zaoberať neskôr.

Príklad. Uvažujme abecedu $\{\mathrm{A}, \mathrm{B}, \ldots, \mathrm{Z}\}$, obsahujúcu písmená anglickej abecedy. Nech P aj C sú rovné tejto množine. Nech $K = \{0, \ldots, 25\}$. Pre potreby šifrovania priraďme každému písmenu abecedy číslo prislúchajúce jeho poradiu, začínajúc od nuly: A = 0, B = 1, …, Z = 25. Šifrovacia transformácia pripočíta k znaku kľúč modulo 26, pri dešifrovaní sa musí postupovať opačne:

$$E(p,k) = (p+k) \bmod 26,$$
$$D(c,k) = (c-k) \bmod 26.$$

Pre šifrovanie dlhších textov sa transformácia aplikuje na každý znak samostatne. Šifrujme otvorený text „ABRAKA“ s použitím kľúča 11:

$$\begin{aligned} \text{ABRAKA} &= 0,\ 1,\ 17,\ 0,\ 10,\ 0 \\ &\quad\ \downarrow \quad +11 \\ &= 11,\ 12,\ 2,\ 11,\ 21,\ 11 \\ &= \text{LMCLVL} \end{aligned}$$

Šifrový text je teda „LMCLVL“. Pre dešifrovanie stačí kľúč odpočítať.

V minulosti sa uplatňovali dva základné princípy konštrukcie symetrických šifier: substitúcia a permutácia. Substitučné šifry nahrádzajú znaky otvoreného textu inými znakmi. Permutačné šifry nemenia znaky v otvorenom texte, ale menia ich poradie. Ilustrujme tieto princípy na dvoch príkladoch.

Jednoduchá substitučná šifra

Označme A abecedu jazyka otvoreného textu. Nech $P = C = A$. Množina kľúčov K je množina všetkých permutácií nad A. Šifrovacia a dešifrovacia funkcia sú definované takto:

$$E(p,k) = k(p),$$
$$D(c,k) = k^{-1}(c).$$

Ak chceme šifrovať texty dlhšie ako jeden znak, tak aplikujeme E postupne na všetky znaky. Pri dešifrovaní postupujeme analogicky, len použijeme inverznú permutáciu.

Permutačná šifra

Dané je prirodzené číslo $m \geq 1$. Označme A abecedu jazyka otvoreného textu. Nech $P = C = A^m$, t.j. m-tice znakov abecedy. Množina kľúčov K je množina permutácií množiny $\{1, 2, \ldots, m\}$. Šifrovacia a dešifrovacia funkcia sú definované takto:

$$E(p_1 p_2 \ldots p_m, k) = p_{k(1)} p_{k(2)} \cdots p_{k(m)},$$
$$D(c_1 c_2 \ldots c_m, k) = c_{k^{-1}(1)} c_{k^{-1}(2)} \cdots c_{k^{-1}(m)}.$$

Dlhší text rozdelíme na m-tice, ktoré šifrujeme zvlášť.

Vernamova šifra

Dané je prirodzené číslo $m \geq 1$ a nech $A = \{0, 1\}$ je abeceda otvoreného textu. Nech $P = C = \{0, 1\}^m$. Pri šifrovaní volíme kľúč z množiny $K = \{0, 1\}^m$. Šifrovanie spočíva v pripočítaní kľúča po bitoch k otvorenému textu modulo 2:

$$E(p, k) = p \oplus k = p_1 \oplus k_1, \ldots, p_m \oplus k_m.$$

Pri dešifrovaní využijeme fakt, že pre všetky $x \in \{0, 1\}$ platí: $x \oplus x = 0$. Preto stačí k šifrovému textu opäť pripočítať kľúč:

$$D(c, k) = c \oplus k = (p \oplus k) \oplus k = p.$$

Vernamova šifra (niekedy nazývaná one-time pad) je príkladom absolútne bezpečnej šifry, ak sú splnené tieto predpoklady:

1. Kľúče sú volené (vyberané) z množiny K náhodne, nezávisle a s rovnakou pravdepodobnosťou.
2. Pri šifrovaní ďalšieho otvoreného textu zvolíme vždy nový kľúč z K.

Ak by sme použili rovnaký kľúč, povedzme k, na šifrovanie otvorených textov p_1 a p_2, tak vieme sčítaním šifrových textov eliminovať vplyv kľúča a získať

$$(p_1 \oplus k) \oplus (p_2 \oplus k) = p_1 \oplus p_2.$$

Zo súčtu dvoch otvorených textov, pokiaľ sú redundantné, je možné oba texty určiť. Popis postupu presahuje rozsah nášho úvodu do kryptológie.

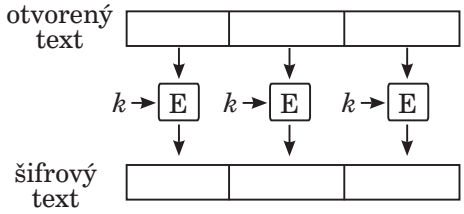
Absolútna (nepodmienená) bezpečnosť Vernamovej šifry znamená neformálne to, že útočník nedokáže zo šifrového textu určiť otvorený text, bez ohľadu na výpočtovú kapacitu, ktorou disponuje. Každý otvorený text môže byť tým pravým a útočník je pri jeho určovaní v rovnakej situácii, ako keby žiadny šifrový text nemal. Šifrový text c môže byť šifrovanou podobou otvoreného textu p_1 aj p_2, ak bol na šifrovanie použitý kľúč $k_1 = c \oplus p_1$ (v prvom prípade) alebo $k_2 = c \oplus p_2$ (v druhom prípade). Útočník nepozná kľúč, ktorý bol zvolený náhodne a nezávisle, preto môže byť otvorený text akýkoľvek.

Napriek bezpečnostným kvalitám Vernamovej šifry, má táto šifra aj dôležitý nedostatok. Ak chceme preniesť m bitov dlhú správu, potrebujeme vygenerovať a bezpečne doručiť adresátovi m bitov dlhý kľúč. Dôverný prenos správy sme vymenili za dôverný prenos kľúča. V situácii, keď náhodné bity vygenerujeme v dostatočnom množstve dopredu a neskôr postupne používame, nám však tento problém nemusí prekážať.

1.2 Blokové a prúdové šifry

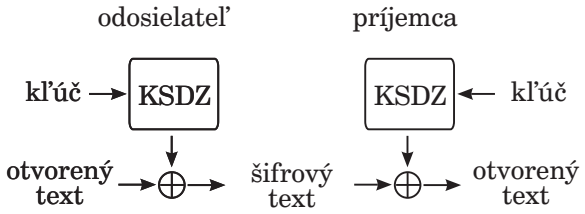
Moderné symetrické šifrovacie systémy používajú kľúče pevnej dĺžky (napr. 128 bitov), ktoré môžeme použiť aj na šifrovanie podstatne dlhších otvorených textov. Teda na umožnenie dôverného prenosu dát (v podstate) ľubovoľnej dĺžky stačí dôverne preniesť adresátovi relatívne krátky tajný kľúč.

Symetrické šifrovacie systémy môžeme rozdeliť na blokové a prúdové. Blokové šifry spracúvajú vstupný text po blokoch pevnej dĺžky, pričom šifrovacia funkcia je definovaná ako transformácia jedného bloku dát. Dešifrovanie je tiež definované nad jednotlivými blokmi.



Obrázok 1.2: Bloková šifra

Prúdové šifry napodobňujú Vernamovu šifru s použitím kratšieho kľúča. Kľúč je použitý na inicializáciu konečnostavového deterministického zariadenia (KSDZ), ktoré produkuje prúd bitov.



Obrázok 1.3: Prúdová šifra

Prúd bitov je sčítavaný modulo 2 s bitmi otvoreného textu. Príjemca šifrového textu použije kľúč a vygeneruje rovnaký prúd bitov na sčítanie so šifrovým textom.

Blokovým aj prúdovým šifrám sa budeme venovať podrobnejšie v kapitolách 3 a **??**.

1.3 Kryptoanalýza

V kryptoanalýze sa snažíme získať otvorený text, šifrovací kľúč, prípadne zašifrovať vlastný otvorený text. Podľa predpokladov, ktoré kryptoanalytické útoky vyžadujú, môžeme rozdeliť útoky na niekoľko základných typov:

1. *Útok len so znalosťou šifrového textu* (COA – ciphertext only attack). Útočník má k dispozícii množinu šifrových textov:

   $$\{E_k(p_1), E_k(p_2), \ldots, E_k(p_n)\},$$

pričom nepozná zodpovedajúce otvorené texty. Cieľom je získať hodnotu k, prípadne určiť niektoré p_i alebo vytvoriť $E_k(p)$ pre zvolený otvorený text p.

2. *Útok so znalosťou otvoreného textu* (KPA – known plaintext attack). Útočník má k dispozícii množinu dvojíc otvorených a šifrových textov:

$$\{\langle p_1, E_k(p_1)\rangle, \langle p_2, E_k(p_2)\rangle, \ldots, \langle p_n, E_k(p_n)\rangle\}.$$

Cieľom je získať hodnotu k, prípadne určiť niektoré p_i alebo vytvoriť $E_k(p)$ pre zvolený otvorený text p.

3. *Útok s možnosťou voľby otvoreného textu* (CPA – chosen plaintext attack). Útočník má možnosť zvoliť si niekoľko otvorených textov, ku ktorým získa zodpovedajúce šifrové texty, pri použití rovnakého kľúča k. Ciele útoku môžu byť rovnaké ako pri KPA.
4. *Útok s možnosťou voľby šifrového textu* (CCA – chosen ciphertext attack). Útočník má možnosť zvoliť si niekoľko šifrových textov, ku ktorým získa zodpovedajúce otvorené texty, pri použití rovnakého kľúča k. Ciele útoku môžu byť rovnaké ako pri KPA.

Pri útokoch CPA a CCA je možné uvažovať aj s ich adaptívnymi variantmi, keď útočník môže voľbu textov opakovať po analýze získaných dát (prípadne aj opakovane niekoľko krát). Od moderných šifrovacích systémov sa očakáva, že sú odolné voči útokom všetkých spomenutých typov.

Prirodzene, pri všetkých útokoch predpokladáme, že útočník pozná šifrovaciu funkciu E (a tiež aj dešifrovaciu funkciu D). Predpokladá sa, že algoritmy sú používané dlhodobo a útočník sa k nim dostane. Bezpečnosť šifrovacieho systému má preto závisieť výlučne na bezpečnosti (utajení) kľúča. Toto pravidlo sa nazýva Kerckhoffov princíp.

Všimnime si, že typy útokov sú zoradené podľa rastúcej sily. To si môžeme demonštrovať napríklad na jednoduchej substitučnej šifre (pozri časť 1.1). Pri COA máme k dispozícii len šifrový text a na jeho dešifrovanie (a určenie kľúča) potrebujeme, ako uvidíme v časti 2.1, vykonať kryptoanalýzu s využitím frekvenčných charakteristík textu. Jednoduchšiu situáciu máme pri KPA, keď priamo získavame niektoré časti kľúča (permutácie) z dvojíc otvorených a šifrových textov. Ak máme šťastie a dvojíc je dostatočne veľa, môžu tieto „pokryť“ celú abecedu a my získame kľúč. V prípade CPA máme situáciu ešte jednoduchšiu. Zvolíme si ako otvorené texty celú abecedu a získané šifrové texty sú priamo kľúčom (permutáciou). Pre CCA je situácia podobná a dostaneme inverznú permutáciu.

Veľkosť priestoru kľúčov

Dôležitou podmienkou bezpečnosti šifrovacieho systému je dostatočná veľkosť priestoru kľúčov, vzhľadom na počet otvorených textov šifrovaných jedným kľúčom. Predpokladáme, že jeden kľúč je použitý na šifrovanie viacerých otvorených textov. Bez ohľadu na použitý šifrovací systém môže útočník skúsiť útok úplným preberaním. Postupne skúša všetky kľúče $k \in K$. Pri COA dešifruje texty a testuje ich „zmysluplnosť“, pri KPA a ďalších porovnáva korektnosť výsledkov so známymi údajmi. Najneskôr po prejdení celého priestoru kľúčov získa správny kľúč. Na zamedzenie

takémuto útoku by mala byť mohutnosť množiny kľúčov (t.j. $|K|$) dostatočne veľká. Veľkosť $|K|$, ktorú možno považovať za dostatočnú, ovplyvňuje aj požiadavka na dĺžku doby, počas ktorej sa útočník nemá dostať k otvorenému textu.

Samozrejme, dostatočne veľké K nie je postačujúcou podmienkou bezpečnosti, ako uvidíme na príklade kryptoanalýzy jednoduchej substitučnej šifry, pre ktorú v prípade anglickej abecedy $\{A, B, \ldots, Z\}$ máme $|K| = 26! \approx 2^{88,4}$.

2 Kryptoanalýza jednoduchých šifier***

... niečo o obsahu, čo bude v tejto časti ...

2.1 Jednoduchá substitučná šifra

Základom kryptoanalýzy jednoduchej substitučnej šifry je analýza frekvencií znakov šifrového textu. V prirodzených jazykoch, ako sú napríklad slovenčina alebo angličtina, je pravdepodobnosť výskytu jednotlivých znakov v texte rôzna. Pravdepodobnosti výskytu znakov slovenčiny a angličtiny v abecede A,...,Z (bez diakritiky a bez medzery) uvádza tabuľka 2.1. Prirodzene, hodnoty pre konkrétny text sa môžu od uvedených líšiť, v závislosti na štýle, téme a najmä dĺžke textu.

znak	%	**znak**	%
A	10,88	C	3,58
O	9,39	U	3,13
E	8,49	P	3,03
S	6,17	Z	3,00
N	5,99	Y	2,70
I	5,79	H	2,51
T	5,76	J	2,19
R	4,74	B	1,81
V	4,63	G	0,21
L	4,41	F	0,20
K	3,99		
D	3,79		
M	3,61		

(a) slovenčina

znak	%	**znak**	%
E	12,70	M	2,41
T	9,06	W	2,36
A	8,17	F	2,23
O	7,51	G	2,02
I	6,97	Y	1,97
N	6,75	P	1,93
S	6,33	B	1,49
H	6,09	V	0,98
R	5,99	K	0,77
D	4,25	J	0,15
L	4,03	X	0,15
C	2,78	Q	0,10
U	2,76	Z	0,07

(b) angličtina

Tabuľka 2.1: Pravdepodobnosť výskytu znakov v prirodzenom jazyku

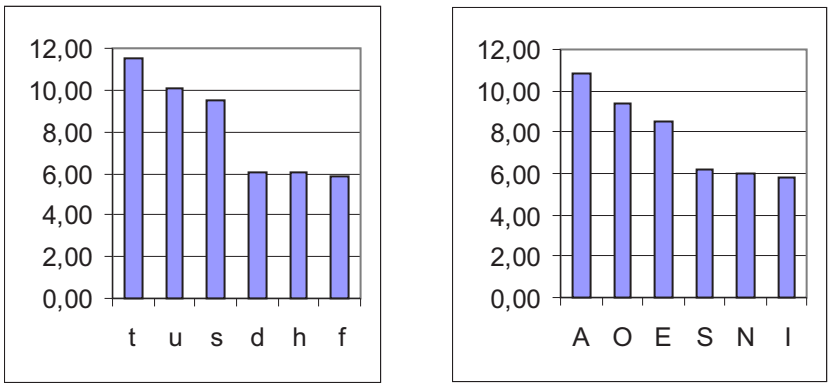
Jednoduchá substitučná šifra nemení frekvenčné charakteristiky znakov textu, preto môžeme očakávať, že najfrekventovanejšie znaky jazyka otvoreného textu zodpovedajú najfrekventovanejším znakom v šifrovom texte. Postup kryptoanalýzy ilustrujeme na konkrétnom príklade. Nasledujúci šifrový text vznikol z otvoreného textu v slovenskom jazyku, pričom bola použitá abeceda s medzerou a bez diakritiky. Medzery boli v šifrovom texte zachované.

```
pishzf ksj lxsis ju juxshty ptqty rifjty ksj cjrc katloc u
xhqcpfzf lt hcritydse piuktxds hclkc ue quqt yfshqdthq otzu
ltrtduzs clijfuxudu du xmzshqsdty pfhukty hqtzs dsotzt udf
jidrt piukac tyfsqdcqs hqsdm otzf ksihqxt xmofszsds hfitru
ptatxru pif juldse hqsds otzu toqfuadcqu dulasidmy qyuxtjszsdmy
oitruqty pt josjdse pisazfulrs uikafxc hchslfuksat h piuktxdtc
xmhzf du licam lxti j rqtisat hu xkauljuzt lt pifefyukse huzm
hquim pfhui dsixtjds toeuhdfz js pifefyukfu huzu hu dsptcjfxuzu
tl tlkatlc xmhsqitxuqszu es lthq ytjds js dseuru hqtzfkru kf
hqtz dshqteu du ptxtldty yfshqs
```

Na začiatok zostrojíme tabuľku frekvencií jednotlivých znakov. Najviac nás zaujímajú znaky s najväčšou frekvenciou výskytu, v uvedenom šifrovom texte sú to:

znak	t	u	s	d	h	f	i	q	...
%	11,48	10,12	9,53	6,03	6,03	5,84	5,06	5,06	...

Pokiaľ je text dostatočne dlhý, možeme očakávať, že najviac frekventované znaky v šifrovom texte zodpovedajú najviac frekvetovaným znakom jazyka. Porovnanie šiestich najfrekventovanejších znakov šifrového textu a slovenčiny je na obrázku 2.1.



(a) šifrový text (b) slovenčina

Obrázok 2.1: Porovnanie najfrekventovanejších znakov šifrového textu a slovenčiny

Na začiatok zvolíme hypotézu, že znaky `t`, `u` a `s` zodpovedajú v otvorenom texte samohláskam `A`, `O` a `E`, hoci nie nutne v tomto poradí. Zároveň budeme predpokladať, že ďalšia dvojica znakov `d` a `h` šifrového textu zodpovedá najčastejším spoluhláskam slovenčiny `S` a `N`, opäť bez určenia poradia. Na zistenie správneho priradenia možno využiť dvojice znakov (digramy). Počet výskytov všetkých dvojpísmenových kombinácii predpokladanej spoluhlásky (`d`/`h`) a predpokladanej samohlásky (`t`/`u`/`s`) je v nasledujúcej tabuľke:

digram	**početnosť**	**digram**	**početnosť**
`dt`	4	`ht`	1
`du`	6	`hu`	6
`ds`	13	`hs`	2

Keďže v slovenčine sú digramy `NA`/`SA` častejšie ako digramy `NO`/`SO` alebo `NE`/`SE`, bude `u` pravdepodobne šifrovaná podoba písmena `A` (takáto dešifrovacia transformá-

cia sa bude označovať u ↦ A). Túto domnienku potvrdzujú samostatne stojace slová hu a du (NA a SA sú bežné dvojpísmenové slová). Samostatné h v šiestom riadku umožní určiť ako dešifrovanú podobu h znak S, lebo predložka N neexistuje. Teda h ↦ S a následne d ↦ N. Dešifrovanie t a s je možné na základe výskytu dvojpísmenových slov. V slovenčine sú častejšie takéto slová končacie na O (rôzne predložky) na rozdiel od slov končiacich na E. V šifrovom texte sa nachádzajú slová lt (dvakrát), pt, tl a js. Prihliadnuc aj na skutočnosť, že dvojpísmenové slová začínajúce E sa takmer nevyskytujú, bude t ↦ O a s ↦ E. Po uvedených transformáciach má šifrový text čiastočne dešifrovanú podobu:

```
piESzf kEj lxEiE jA jAxESOy pOqOy rifjOy kEj cjrc kaOloc A
xSqcpfzf lO ScriOyNEe piAkOxNE Sclkc Ae qAqO yfESqNOSq oOzA
lOrONAzE clijfAxANA NA xmzESqENOy pfSAkOy SqOzE NEoOzO ANf
jiNrO piAkac OyfEqNcqE SqENm oOzf kEiSqxO xmofEzENE SfiOrA
pOaOxrA pif jAlNEe SqENE oOzA OoqfAaNcqA NAlaEiNmy qyAxOjEzENmy
oiOrAqOy pO joEjNEe piEazfAlrE Aikafxc ScSElfAkEaO S piAkOxNOc
xmSzf NA licam lxOi j rqOiEaO SA xkaAljAzO lO pifefyAkEe SAzm
SqAim pfSAi NEixOjNE OoeASNfz jE pifefyAkfA SAzA SA NEpOcjfxAzA
Ol OlkaOlc xmSEqiOxAqEzA eE lOSq yOjNE jE NEeArA SqOzfkrA kf
SqOz NESqOeA NA pOxOlNOy yfESqE
```

Z dvojpísmenových slov lO a Ol možno usúdiť, že l ↦ D. Následne zo slov SqENE (5. riadok) a DOSq (9. riadok, po predchádzajúcej substitúcii) predpokladáme, že q ↦ T. Túto hypotézu podporuje aj skutočnosť, že q je ôsme najfrekventovanejšie písmeno šifrového textu a T je siedme najfrekventovanejšie písmeno v slovenčine.

```
piESzf kEj DxEiE jA jAxESOy pOTOy rifjOy kEj cjrc kaODoc A
xSTcpfzf DO ScriOyNEe piAkOxNE ScDkc Ae TATO yfESTNOST oOzA
DOrONAzE cDijfAxANA NA xmzESTENOy pfSAkOy STOzE NEoOzO ANf
jiNrO piAkac OyfETNcTE STENm oOzf kEiSTxO xmofEzENE SfiOrA
pOaOxrA pif jADNEe STENE oOzA OoTfAaNcTA NADaEiNmy TyAxOjEzENmy
oiOrATOy pO joEjNEe piEazfADrE Aikafxc ScSEDfAkEaO S piAkOxNOc
xmSzf NA Dicam DxOi j rTOiEaO SA xkaADjAzO DO pifefyAkEe SAzm
STAim pfSAi NEixOjNE OoeASNfz jE pifefyAkfA SAzA SA NEpOcjfxAzA
OD ODkaODc xmSETiOxATEzA eE DOST yOjNE jE NEeArA STOzfkrA kf
STOz NESTOeA NA pOxODNOy yfESTE
```

Napriek tomu, že mnohé písmená ešte nie sú určené, dajú sa teraz znaky určovať z „takmer“ vylúštených slov. Napríklad slová yfESTNOST (2. riadok) a yfESTE (posledný riadok) vedú k trasformáciam y ↦ M a f ↦ I. Zo slova NESTOeA (posledný riadok) dostaneme e ↦ J.

```
piESzI kEj DxEiE jA jAxESOM pOTOM riIjOM kEj cjrc kaODoc A
xSTcpIzI DO ScriOMNEJ piAkOxNE ScDkc AJ TATO MIESTNOST oOzA
DOrONAzE cDijIAxANA NA xmzESTENOM pISAkOM STOzE NEoOzO ANI
jiNrO piAkac OMIETNcTE STENm oOzI kEiSTxO xmoIEzENE SIiOrA
pOaOxrA piI jADNEJ STENE oOzA OoTIAaNcTA NADaEiNmM TMAxOjEzENmM
oiOrATOM pO joEjNEJ piEazIADrE AikaIxc ScSEDIAkEaO S piAkOxNOc
xmSzI NA Dicam DxOi j rTOiEaO SA xkaADjAzO DO piIJIMAkEJ SAzm
STAim pISAi NEixOjNE OoJASNIz jE piIJIMAkIA SAzA SA NEpOcjIxAzA
OD ODkaODc xmSETiOxATEzA JE DOST MOjNE jE NEJArA STOzIkrA kI
STOz NESTOJA NA pOxODNOM MIESTE
```

Rovnako postupujeme ďalej:

- `pOTOM` (1.riadok), `pOxODNOM` (posledný riadok) $\Rightarrow$ `p` $\mapsto$ `P`, `x` $\mapsto$ `V`
- `OMIETNcTE STENm` (4. riadok) $\Rightarrow$ `c` $\mapsto$ `U`, `m` $\mapsto$ `Y`
- `MOjNE jE` (9. riadok) $\Rightarrow$ `j` $\mapsto$ `Z`

```
PiESzI kEZ DVEiE ZA ZAVESOM POTOM riIZOM kEZ UZrU kaODoU A
VSTUPIzI DO SUriOMNEJ PiAkOVNE SUDkU AJ TATO MIESTNOST oOzA
DOrONAzE UDiZIAVANA NA VYzESTENOM PISAkOM STOzE NEoOzO ANI
ZiNrO PiAkaU OMIETNUTE STENY oOzI kEiSTVO VYoIEzENE SIiOrA
POaOVrA PiI ZADNEJ STENE oOzA OoTIAaNUTA NADaEiNYM TMAVOZEzENYM
oiOrATOM PO ZoEZNEJ PiEazIADrE AikaIVU SUSEDIAkEaO S PiAkOVNOU
VYSzI NA DiUaY DVOi Z rTOiEaO SA VkaADZAzO DO PiIJIMAkEJ SAzY
STAiY PISAi NEiVOZNE OoJASNIz ZE PiIJIMAkIA SAzA SA NEPOUZIVAzA
OD ODkaODU VYSETiOVATEzA JE DOST MOZNE ZE NEJArA STOzIkrA kI
STOz NESTOJA NA POVODNOM MIESTE
```

Rovnako postupujeme ďalej a získame zostávajúce substitúcie:

- `PiESzI` (1.riadok), `VSTUPIzI` (2. riadok) $\Rightarrow$ `i` $\mapsto$ `R`, `z` $\mapsto$ `L`
- `kEZ` (1.riadok), `SUDkU` (2.riadok) $\Rightarrow$ `k` $\mapsto$ `C`
- `NEJArA` (9.riadok), `UZrU` (1. riadok) $\Rightarrow$ `r` $\mapsto$ `K`
- `oOzI` (4.riadok), `oOzA` (5. riadok) $\Rightarrow$ `o` $\mapsto$ `B`
- `SUSEDIAkEaO` (6.riadok) $\Rightarrow$ `a` $\mapsto$ `H`

```
PRESLI CEZ DVERE ZA ZAVESOM POTOM KRIZOM CEZ UZKU CHODBU A
VSTUPILI DO SUKROMNEJ PRACOVNE SUDCU AJ TATO MIESTNOST BOLA
DOKONALE UDRZIAVANA NA VYLESTENOM PISACOM STOLE NEBOLO ANI
ZRNKO PRACHU OMIETNUTE STENY BOLI CERSTVO VYBIELENE SIROKA
POHOVKA PRI ZADNEJ STENE BOLA OBTIAHNUTA NADHERNYM TMAVOZELENYM
BROKATOM PO ZBEZNEJ PREHLIADKE ARCHIVU SUSEDIACEHO S PRACOVNOU
VYSLI NA DRUHY DVOR Z KTOREHO SA VCHADZALO DO PRIJIMACEJ SALY
STARY PISAR NERVOZNE OBJASNIL ZE PRIJIMACIA SALA SA NEPOUZIVALA
OD ODCHODU VYSETROVATELA JE DOST MOZNE ZE NEJAKA STOLICKA CI
STOL NESTOJA NA POVODNOM MIESTE
```

Frekvenčnou analýzou a následným dopĺňaním vhodných znakov do takmer dešifrovaných slov sme získali (okrem otvoreného textu) aj použitý šifrovací kľúč:

```
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
u o k l s - - a f e r z y d t p - i h q c x - - m j
```

Zaujímavosťou otvoreného textu je skutočnosť, že okrem písmen, ktoré sú v slovenčine cudzie (`Q`, `W`, `X`), sa v texte nenachádzajú ani dva najmenej frekventované znaky slovenčiny (`F`, `G`).

2.2 Entropia a vzdialenosť jednoznačnosti

Kryptoanalýza jednoduchej substitučnej šifry v časti 2.1 je tým jednoduchšia, čím je šifrový text dlhší. Vtedy je vyšší predpoklad, že frekvenčné charakteristiky textu zodpovedajú charakteristikám znakov v jazyku. Na druhej strane, krátky šifrový text

kryptonalýzu sťažuje. Uvažujme napríklad šifrový text `JDEWR` šifrovaný jednoduchou substitučnou šifrou. Otvoreným textom môže byť slovo `kurca` alebo `kniha` alebo `alebo` a mnohé ďalšie, ak je otvorený text v slovenčine. Ak je otvorený text v angličtine, môže to byť `first` alebo `today` alebo `great`, atď. Základnou podmienkou pre úspešnú kryptoanalýzu je jednoznačnosť dešifrovania.

Definícia 1. Nech X je diskrétna náhodná premenná, ktorá nadobúda hodnoty x_1, $\ldots$, x_n s pravdepodobnosťami $p_1, \ldots, p_n$. Entropia náhodnej premennej X, označíme ju $H(X)$ alebo $H(p_1; \ldots; p_n)$, je daná výrazom

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i.$$

Entropiu môžeme interpretovať ako očakávaný počet bitov potrebných na zapísanie výsledku pokusu X. Ak napríklad X označuje hod mincou, entropia X je $H(1/2; 1/2) = 1$ (bit). Ak X označuje náhodný, rovnako pravdepodobný výber znaku z abecedy 26 znakov, tak entropia X je $H(X) = -26 \cdot (1/26) \cdot \log_2(1/26) = \log_2 26 \approx 4,70$ (bitov).

Uvažujme teraz pravdepodobnosti výskytu znakov prirodzených jazykov. Pre entropiu náhodných premenných zodpovedajúcich výberu jedného znaku slovenčiny a angličtiny podľa frekvenčných charakteristík tabuľky 2.1 dostaneme:

$$H_{\text{slovenčina}}(10{,}88;\ 9{,}39;\ \ldots;\ 0{,}21;\ 0{,}20) \approx 4{,}23$$
$$H_{\text{angličtina}}(12{,}70;\ 9{,}06;\ \ldots;\ 0{,}10;\ 0{,}07) \approx 4{,}15$$

Uvedená entropia modeluje text v prirodzenom jazyku ako postupnosť nezávislých realizácií pokusov náhodnej premennej zodpovedajúcej voľbe jedného znaku. Prirodzene, presnejší model dostaneme, ak budeme brať výskyt n-tíc znakov, pre $n \to \infty$. Pre angličtinu máme:

$$H_{\text{angličtina}}(\text{pravdepodobnosti dvojíc znakov}) \approx 3{,}65$$
$$H_{\text{angličtina}}(\text{pravdepodobnosti trojíc znakov}) \approx 3{,}22$$
$$\ldots$$
$$H_{\text{angličtina}} \approx 1{,}5$$

Hodnoty entropií pre vzrastajúce n sa počítajú ťažko, vzhľadom na veľký (exponenciálny) počet hodnôt, ktoré môže náhodná premenná nadobudnúť. Výslednú entropiu jazyka pre $n \to \infty$ možno určiť štatistickým testovaním. Experimentálny odhad možno získať skúmaním kompresného pomeru pre texty v danom jazyku získané najlepšími kompresnými algoritmami. Podľa vyššie uvedenej hodnoty entropie pre angličtinu môžeme usudzovať, že ASCII text v anglickom jazyku (bez medzier) možno skomprimovať na $1{,}5/8 \approx 19\%$ pôvodnej veľkosti.

Definícia 2. Nech Y označuje náhodnú premennú – otvorený text dĺžky n generovaný zdrojom S nad abecedou mohutnosti q. Potom redundancia D_n textu Y je definovaná:

$$D_n = n \cdot \log_2 q - H(Y).$$

Hodnota $\delta_n = D_n/n$ sa nazýva priemernou redundanciou na jeden znak.

Neformálne, redundancia otvoreného textu vyjadruje, o koľko bitov je otvorený text dlhší ako reťazec potrebný na jeho zápis. Jednoznačnosť dešifrovania šifrového textu je predmetom nasledujúcej definície.

Definícia 3. V prípade útoku len so znalosťou šifrového textu (COA) na systém s priestorom kľúčov K a zdrojom otvoreného textu S nazveme vzdialenosťou jednoznačnosti otvoreného textu hodnotu

$$\min\{n \in \mathbb{N} \mid D_n \geq H(K)\}.$$

Vzdialenosť jednoznačnosti otvoreného textu môžeme interpretovať tak, že ak je šifrový text aspoň takto dlhý, tak mu zodpovedá práve jeden zmysluplný otvorený text (teda dešifrovanie je jednoznačné). Definícia túto dĺžku určuje tak, že akonáhle presiahne počet nadbytočných bitov otvoreného textu počet bitov potrebných na reprezentáciu kľúča, tak tento kľúč vieme jednoznačne určiť.

Uvažujme jednoduchú substitučnú šifru pre angličtinu. Ak sú všetky kľúče (permutácie) rovnako pravdepodobné, máme $H(K) = \log_2(26!) \approx 88{,}38$. Aproximujme redundaciu pre n znakov takto: $D_n = (4{,}7 - 1{,}5) \cdot n = 3{,}2 \cdot n$. Potom pre vzdialenosť jednoznačnosti otvoreného textu určíme z nerovnosti $3{,}2 \cdot n \geq 88{,}38$. Riešením dostaneme $n \geq 28$.

2.3 Vigenereova šifra

Vigenereova šifra je príkladom polyalfabetickej subtitúcie, keď sa rovnaký znak otvoreného textu môže zobraziť na rôzne znaky šifrového textu, v závislosti na pozícii v otvorenom texte. Dané je prirodzené číslo $n \geq 1$ (dĺžka kľúča). Nech abeceda jazyka otvoreného textu A má m prvkov. Každému znaku abecedy priraďme číslo z množiny $\mathbb{Z}_m = \{0, 1, \ldots, m-1\}$ (teda znaky očíslujeme). Množina kľúčov K je množina všetkých n-tíc prvkov zo $\mathbb{Z}_m$: $K = \{k = (k_1, \ldots, k_n) \mid 0 \leq k_1, \ldots, k_n \leq m-1\}$. Množina otvorených textov (P) a množina šifrových textov (C) sú tak isto n-tice prvkov zo $\mathbb{Z}_m$. Šifrovacia a dešifrovacia funkcia sú definované takto:

$$E(p_1 p_2 \ldots p_n, k) = (p_1 + k_1) \bmod m, \ldots, (p_n + k_n) \bmod m,$$
$$D(c_1 c_2 \ldots c_n, k) = (c_1 - k_1) \bmod m, \ldots, (c_n - k_n) \bmod m.$$

Dlhší text rozdelíme na n-tice, ktoré šifrujeme zvlášť.

Ilustrujme Vigenereovu šifru na príklade. Nech $A = \{A, B, \ldots, Z\}$, teda $m = 26$, pričom čísla 0,...,25 priradíme písmenám podľa poradia v abecede. Nech dĺžka kľúča je $n = 5$ a jeho hodnota $(10, 11, 20, 2)$, teda `KLUC`. Šifrovaním otvoreného textu `OTVORENY TEXT` dostaneme:

O	T	V	O	R	E	N	Y		T	E	X	T
14	19	21	14	17	4	13	24		19	4	23	19
10	11	20	2	10	11	20	2		10	11	20	2
24	4	15	16	1	15	7	0		3	15	17	21
y	e	p	q	b	p	h	a		d	p	r	v

Vidno, že vo Vigenereovej šifre sa môže znak otvoreného textu zobraziť na rôzne znaky šifrového textu, napríklad `T` na `e`, `d` alebo `v`. Navyše aj rôzne znaky otvoreného textu sa môžu zobraziť na rovnaké znaky v šifrovom texte, napríklad `V` a `E`

na p. Vigenereova šifra vo všeobecnosti vyrovnáva frekvenčné charakteristiky znakov šifrového textu. Priamočiare uplatnenie rovnakej analýzy ako v prípade jednoduchej substitučnej šifry nie je možné. Kryptoanalýza Vigenereovej šifry postupuje v dvoch krokoch:

1. určenie dĺžky kľúča (hodnoty n);
2. určenie jednotlivých prvkov kľúča $(k_1, \ldots, k_n)$.

Kryptoanalýzu si ilustrujeme na konkrétnom prípade šifrového textu. Otvorený text je v slovenskom jazyku nad abecedou $\{A, B, \ldots, Z\}$. Pre zvýšenie prehľadnosti je šifrový text rozdelený do pätíc.

```
xfawd sgazy xvnwp cxdhq fkwrb xxjaf bfdjq vxoge bxhar ykaoa
yojwz zorou cqwoy cbopc oqnkl izdrb xxlkj yjowf bfxhy jkaph
sppkc oqnkl yalkr bbveq dskny riwrd kshjq brpei krlhd ohkhc
ysvpo mbjwf yppqf xbwge cxxhy jfowp fxyok tbevl ipqfu xxnkp
nfahe npeai dfydf bbzyx kavwz ezeyx wxhad wfand oplad okudh
oyajq tgafx ehkpz ofjuq ulgat nrjex bliwf yqkil nfwha orpes
rxgat cxnkp doeai dfjwe llydk dboks rxrnk dfiat jfjad kowvy
xxogq vkqoj okqre nvfae fbhwl sxywa ymnee cqwpd izdrb xxyda
brpei kxguf sxgkl ulphy kiajf yaaoy kqeyx kwlwj xxopy meoaa
ekzws rsoaj ulpkl soajy omnai dxrwq drlny nbijq fbhiy nlhap
squkr txrdb llgkt yiqne fkkle niwry mhkqt yowvq trrhd ikwkp
brpju lowhq doeai dfwoq kmnqt ulqoj emqfk cmwpe lokri uxiwi
kskzo cxjac ywane jmpub sqhar ypwra vfjec oavet fxqpu cvgpe
bbpre bfwwa epele nhkrk jxjec spwpo mfogq vkuik bqwgp ojwoq
flzud ojwed oekro mekzy chwhu xqkqy cqkqs oppkk fowpy dpwof
kqzkc yowfu dlrab wfzkb owepu vbxkq upwre mereb shazi kshjq
blvph sbopy kskrq vfzkf bfadb lkade nfivk dboqi foaye wkwlb
xbjuc ulgki ysuiy yoayx wfwle xlnec blrje nljaz lbvjq tjaji
szdle meuxc kkwof sxpks xbfyu cqakt xboeu cloar yr
```

Dĺžku kľúča môžeme určiť napríklad Kasiskiho metódou. Myšlienka metódy spočíva v tom, že rovnaké časti otvoreného textu sa zašifrujú na rovnaké časti šifrového textu, ak sú ich pozície vzdialené o celočíselný násobok dĺžky kľúča. Preto v šifrovom texte hľadáme páry rovnakých častí (dĺžky aspoň 3) a zapamätáme si ich posuny. Po nájdení viacerých posunov, označme ich $d_1, d_2, \ldots, d_t$ usúdime, že $n = \mathrm{nsd}(d_1, \ldots, d_t)$. V našom príklade sú niektoré rovnaké štvorice znakov v šifrovom texte zvýraznené (podčiarknuté). Jednotlivé posuny sú: pre štvoricu `rbxx` je posun 55, pre `aidf` je posun 105 a pre `ogqv` je posun 330. Podľa Kasiskiho metódy usúdime, že dĺžka kľúča je $n = \mathrm{nsd}(55, 105, 330) = 5$. Samozrejme, môže sa stať, že v šifrovom texte existuje rovnaká časť s posunom iným ako celočíselný násobok dĺžky kľúča. V našom príklade je takou štvoricou `soaj` s posunom 9. Vo všeobecnosti je potrebné analyzovať faktory najčastejšie sa vyskytujúcich faktorov posunov.

Druhou, jednoduchšie automatizovateľnou a zvyčajne presnejšou metódou je využitie indexu koincidencie. Autorom tejto metódy je Friedman.

Definícia 4. Nech $x = x_1x_2 \dots x_t$ je reťazec t znakov. Index koincidencie reťazca x, označíme ho $I_c(x)$, je pravdepodobnosť, že dva náhodne zvolené znaky z x sú zhodné.

Pripomeňme, že abeceda (otvoreného aj šifrového textu, reprezentovaná aj ako $\mathbb{Z}_m$) má m znakov. Označme počet výskytov jednotlivých znakov v reťazci x postupne f_1, f_2, …, f_m. Potom pre index koincidencie reťazca x platí

$$I_c(x) = \frac{\sum_{i=1}^{m} \binom{f_i}{2}}{\binom{t}{2}} = \frac{\sum_{i=1}^{m} f_i(f_i-1)}{t(t-1)}.$$

Ak reťazec x je reťazcom prirodzeného textu, očakávame, že $I_c \approx \sum_{i=1}^{m} p_i^2$, kde $p_1, \dots, p_m$ sú pravdepodobnosti výskytov jednotlivých znakov v príslušnom jazyku. Možno očakávať, že čím dlhší je reťazec x, tým je index koincidencie bližšie k uvedenej (teoretickej) hodnote. Pre slovenčinu a angličtinu dostaneme takéto očakávané hodnoty indexu koincidencie, využijúc pravdepodobnosti z tabuľky 2.1.

slovenčina: $10{,}88^2 + 3{,}58^2 + \dots + 0{,}21^2 + 0{,}20^2 \approx 0{,}0588$
angličtina: $12{,}70^2 + 9{,}06^2 + \dots + 0{,}10^2 + 0{,}07^2 \approx 0{,}0655$

Ak by boli znaky reťazca volené náhodne s rovnakou pravdepodobnosťou, index koincidencie reťazca nad slovenskou abecedou bude $23 \cdot (1/23)^2 \approx 0{,}0435$. Pre anglickú abecedu $26 \cdot (1/26)^2 \approx 0,0385$.

Dôležitá vlastnosť indexu koincidencie je v tom, že ak vykonáme ľubovoľnú jednoduchú (monoalfabetickú) substitúciu nad abecedou (teda aj posun abecedy ako vo Vigenereovej šifre), hodnota indexu koincidencie sa nezmení. Postupne budeme testovať dĺžku kľúča $n = 1, 2, \dots$. Pre skutočnú dĺžku kľúča n sú znaky i, $n+i$, $2n+i$, …(pre $1 \le i \le n$) šifrované s použitím rovnakého k_i. Teda šifrový text x sa rozpadne na n stôp:

k_1: $x_1\, x_{n+1}\, x_{2n+1} \dots$
k_2: $x_2\, x_{n+2}\, x_{2n+2} \dots$
$\dots$
k_n: $x_n\, x_{2n}\, x_{3n} \dots,$

pričom každá stopa je získaná z otvoreného textu jednoduchou substitúciou. Preto očakávame, že každá stopa má index koincidencie rovný približne indexu koincidencie jazyka. Pokiaľ rozdelíme šifrový text na stopy podľa zlej dĺžky kľúča, budú získané stopy kombináciou dvoch alebo viacerých jednoduchých substitúcií – teda index koincidencie bude bližší k hodnote indexu pre náhodne zvolené reťazce.

V nasledujúcej tabuľke je uvedený priemerný index koincidencie reťazcov (stôp), na ktoré sa rozpadne šifrový text, ak predpokladáme kľúč dĺžky n:

n	I_c	n	I_c	n	I_c
1	0,0438	5	0,0603	9	0,0427
2	0,0440	6	0,0440	10	0,0601
3	0,0438	7	0,0437	11	0,0437
4	0,0443	8	0,0453	12	0,0442

Ľahko vidieť, že dĺžka kľúča je $n = 5$. Poznamenajme, že rovnako „dobré“ hodnoty indexu koincidencie očakávame aj pre násoboky dĺžky kľúča, v našom príklade to ilustruje prípad $n = 10$.

Ďalší krok kryptoanalýzy Vigenereeovej šifry spočíva v určení kľúča $(k_1, \ldots, k_n)$. Najskôr určíme relatívne vzdialenosti k_1 a ostatných častí kľúča: $k_2 - k_1, \ldots, k_n - k_1$.

Definícia 5. Nech $x = x_1 x_2 \ldots x_t$ a $y = y_1 y_2 \ldots y_{t'}$ sú reťazce dĺžky t a t' znakov. Vzájomný index koincidencie reťazcov x a y, označíme ho $MI_c(x, y)$, je pravdepodobnosť, že náhodne zvolený znak z x je zhodný s náhodne zvoleným znakom y.

Označme počet výskytov jednotlivých znakov v reťazci x postupne f_1, f_2, …, f_m a počet výskytov znakov v reťazci y postupne f'_1, f'_2, …, f'_m. Potom vzájomný index koincidencie reťazcov x a y možno vypočítať takto:

$$MI_c(x, y) = \frac{\sum_{i=1}^{m} f_i \cdot f'_i}{t \cdot t'}.$$

Vzájomný index koincidencie reťazcov x a y sa nemení, ak na reťazce aplikujeme rovnakú jednoduchú substitúciou. Očakávame, že reťazce získané z prirodzeného jazyka majú vzájomný index koincidencie približne rovný indexu koincidencie jazyka.

Určenie relatívnej vzdialenosti $k_i - k_1$ (pre $i = 2, \ldots, n$) prebieha tak, že znaky stopy šifrového textu pre k_i postupne posúvame (postupne k nim pripočítavame $\delta = 0, 1, \ldots, m-1$) a skúmame vzájomný index koincidencie so stopou šifrového textu pre k_1. Ak $k_i - k_1 = \delta$, je index približne rovný indexu koincidencie jazyka. V opačnom prípade je index bližšie k indexu pre náhodne distribuované znaky.

Vzájomné indexy koincidencie pre ilustračný širový text uvádza nasledujúca tabuľka (podčiarknuté hodnoty zodpovedajú správnym relatívnym vzdialenostiam):

	$k_2 - k_1$	$k_3 - k_1$	$k_4 - k_1$	$k_5 - k_1$
⋮	⋮	⋮	⋮	⋮
4	0,0431	0,0378	0,0346	0,0339
5	0,0338	0,0352	0,0368	0,0384
6	0,0329	0,0352	0,0403	<u>0,0533</u>
7	0,0384	0,0325	0,0381	0,0440
8	0,0356	0,0450	0,0424	0,0307
9	0,0456	0,0370	0,0367	0,0300
10	0,0348	0,0318	0,0297	0,0394
11	0,0306	0,0380	0,0334	0,0353
12	0,0366	<u>0,0588</u>	<u>0,0590</u>	0,0355
13	<u>0,0614</u>	0,0426	0,0438	0,0415
14	0,0412	0,0321	0,0334	0,0404
15	0,0311	0,0305	0,0322	0,0368
⋮	⋮	⋮	⋮	⋮

Zostávajúcou úlohou kryptoanalýzy je určiť hodnotu k_1. Výber k_1 vieme uskutočniť vyskúšaním všetkých m hodnôt. Vďaka relatívnym posunom ostatných kľúčov tak získame m potenciálnych otvorených textov, z ktorých už ľahko (manuálne alebo

automatizovane) identifikujeme ten správny. Pre náš šifrový text je fragment potenciálnych šifrových textov uvedený v nasledujúcej tabuľke:

k_1	otvorený text
⋮	⋮
I	PKGCPKLGFKPATCBUCJNC ...
J	OJFBOJKFEJOZSBATBIMB ...
K	NIEANIJEDINYRAZSAHLA ...
L	MHDZMHIDCHMXQZYRZGKZ ...
M	LGCYLGHCBGLWPYXQYFJY ...
⋮	⋮

Použitý kľúč je teda `KXWWQ` – posuny druhého až piateho znaku kľúča oproti prvému znaku sú 13, 12, 12 a 6 (podľa vzájomného indexu koincidencie). Začiatok otvoreného textu, s doplením niektorých interpunkčných znamienok:

```
Nie, ani jediny raz sa hlavna vlna neprihnala skor alebo
neskor. Najprv sest asi sestmetrovych vln a potom sa
priblizne tristo metrov od pobrezia tvori hlavna vlna.
Ruti sa uplne kolmo, vztycena. ...
```

3 Blokové šifry

Blokové šifry sú triedou symetrických šifrovacích systémov, ktoré spracúvajú otvorený (aj šifrový) text po blokoch pevnej dĺžky. Blokové šifry je možné využiť nielen na zabezpečenie dôvernosti dát šifrovaním, ale aj na konštrukciu iných kryptografických prvkov: hašovacích funkcií, MAC (Message Authentication Code) a podobne. V tejto časti sa budeme venovať blokovým šifrám použitým na šifrovanie.

Medzi základné požiadavky pri návrhu blokových šifier patria:

- Bezpečnosť – šifrovací systém má byť odolný voči všetkým známym útokom.
- Výkon – potreba šifrovať a dešifrovať dáta spomaľuje ich spracovanie. Preto má byť šifrovací systém na zvolenej implementačnej platforme čo najefektívnejší.

Odolnosť šifrovacieho algoritmu voči kryptoanalýze môžeme častokrát zvyšovať pridávaním dodatočných vrstiev spracovania otvoreného textu. Tým algoritmus spomaľujeme. Na druhej strane menej transformácií otvoreného textu síce urýchli šifrovanie, avšak môže vytvoriť v šifrovacom systéme bezpečnostnú slabinu. Ďalšími požiadavkami na šifrovací systém sú pamäťové nároky (ktoré by mali byť čo najmenšie), obmedzenia vyplývajúce z cieľovej platformy, atď.

Uvedené požiadavky sa netýkajú len blokových šifier, ale sú platné aj pre prúdové šifry a ďalšie kryptografické konštrukcie.

3.1 Princípy konštrukcie

Moderné blokové šifry sú realizované elektronicky, ako hardvérové moduly alebo softvérovo, preto je abeceda otvoreného aj šifrového textu $\{0,1\}$. Označme $V_n = \{0,1\}^n$, pre $n \in \mathbb{N}$, množinu n-bitových vektorov. Bloková šifra (šifrovací systém) je dvojica zobrazení $E : V_n \times K \to V_n$ a $D : V_n \times K \to V_n$ taká, že platí:

$$\forall k \in K \, \forall p \in V_n \; : \; D_k(E_k(p)) = p,$$

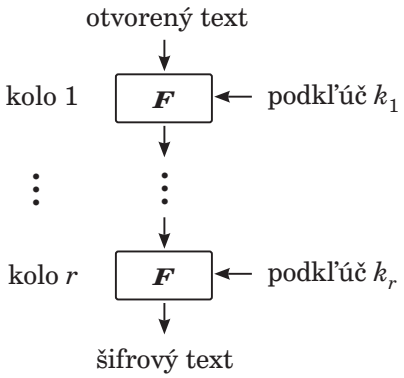
kde K je konečná množina kľúčov. Číslo n sa nazýva dĺžka bloku. Množina K je obvykle rovná V_l, pre nejaké $l \in \mathbb{N}$. Navyše, kľúče sú z K vyberané nezávisle a s rovnakou pravdepodobnosťou. V takomto prípade hovoríme, že (efektívna) dĺžka kľúča je l bitov.

Prirodzenou požiadavkou na množinu kľúčov K je, aby jej mohutnosť bola dostatočne veľká. Malá množina kľúčov umožňuje útok úplným preberaním, keď útočník vyskúša všetky potenciálne kľúče. Čím väčšia je hodnota $|K|$, tým zložitejší je útok. Na druhej strane, priveľký rozsah kľúčov kladie zbytočne zvýšené nároky na ich používanie a správu (generovanie, ukladanie atď.).

Bezpečnosť blokovej šifry je ovplyvnená aj dĺžkou bloku. Príliš krátky blok môže viesť k bezpečnostným problémom. Predstavme si napríklad blokovú šifru s extrémne krátkym blokom, povedzme 4 bity. Vtedy, bez ohľadu na veľkosť $|K|$, je počet všetkých šifrovacích transformácií $(2^4)!$. Tie dokáže útočník prezrieť všetky. Podobne aj ďalšie, pokročilejšie kryptoanalytické techniky využívajú na útok krátku dĺžku bloku. Na druhej strane veľká dĺžka bloku komplikuje šifrovaciu a dešifrovaciu transformáciu a môže byť prekážkou pri implementácii algoritmu na platformách s pamäťovými obmedzeniami.

Pri dĺžke kľúča aj bloku je potrebné voliť vhodný kompromis medzi bezpečnosťou a efektívnosťou.

Najväčšiu skupinu blokových šifier tvoria iterované šifry. Myšlienka iterovaných šifier spočíva v tom, že definujeme základnú transformáciu (kolo), ktorú potom viacnásobne použijeme.



Obrázok 3.1: Iterovaná šifra

V jednotlivých kolách obvykle vstupujú do výpočtu podkľúče šifrovacieho kľúča – v prvom kole podkľúč k_1, v druhom k_2, atď. Podkľúče sú reťazce bitov deterministicky

odvodené z pôvodného kľúča. Proces odvodenia podkľúčov sa nazýva plánovanie kľúča (key-scheduling).

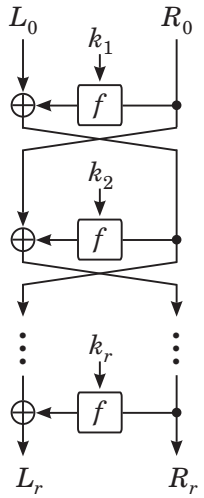
Základnú podmienku – invertovateľnosť šifrovacej transformácie – získame jednoducho vo Feistelovských šifrách. Feistelovské šifry sú triedou iterovaných blokových šifier s rovnakou štruktúrou algoritmu pre šifrovanie aj dešifrovanie. To znamená, že pre dešifrovanie nemusíme implementovať iný algoritmus. Príkladom takejto šifry je algoritmus DES (Data Encryption Standard). Feistelovská šifra rozdelí text na dve polovice, prvú (ľavú) označíme L_0 a druhú (pravú) R_0. V jednotlivých kolách sa počítajú hodnoty L_i, R_i na základe predchádzajúcich:

$$\begin{aligned} L_i &= R_{i-1} \\ R_i &= L_{i-1} \oplus f(k_i, R_{i-1}) \qquad\qquad 1 \leq i < r, \end{aligned}$$

kde f je transformácia ovplyvnená podkľúčom k_i. Výstupom je po r kolách dvojica L_r, R_r, pričom v poslednom kole nedochádza k výmene polovíc:

$$\begin{aligned} L_r &= L_{r-1} \oplus f(k_r, R_{r-1}) \\ R_r &= R_{r-1}. \end{aligned}$$

Schéma Feistelovskej šifry je zobrazená na obrázku 3.2. Pri dešifrovaní môžeme použiť rovnakú schému, stačí otočiť poradie podkľúčov. Kryptografické vlastnosti algoritmu sú určené vlastnosťami funkcie f.



Obrázok 3.2: Feistelovská šifra

Samozrejme, existujú aj iné šifrovacie systémy s rovnakou štruktúrou šifrovania a dešifrovania, ktoré nie sú Feistelovskými šiframi (napr. algoritmus IDEA).

3.2 Viacnásobné šifrovanie

Viacnásobné šifrovanie je motivované potrebou predĺžiť používaný kľúč v šifrovacom algoritme. Najjednoduchšia konštrukcia je použiť rôzne kľúče a šifrovacie, prípadne

dešifrovacie transformácie zoradiť za seba. Tak dostaneme dvojité šifrovanie

$$c = X''_{k_2}(X'_{k_1}(p))$$

alebo trojité šifrovanie

$$c = X'''_{k_3}(X''_{k_2}(X'_{k_1}(p))),$$

kde rôzne X označujú E alebo D, teda použitie šifrovacej alebo dešifrovacej transformácie. Zreťaziť môžeme aj väčší počet transformácií, samozrejme, za cenu zvýšenia výpočtových nárokov. Populárnym módom využitým napríklad aj pri konštrukcii algoritmu Triple DES je trojité šifrovanie v EDE móde:

$$c = E_{k_3}(D_{k_2}(E_{k_1}(p))).$$

Dešifrovanie aplikuje inverzné transformácie s opačným poradím kľúčov:

$$p = D_{k_1}(E_{k_2}(D_{k_3}(c))).$$

Častou voľbou v EDE móde je zvoliť $k_1 = k_3$. Výhodou EDE módu je aj spätná kompatibilita – ak položíme $k_1 = k_2 = k_3$, tak dostaneme pôvodný šifrovací algoritmus.

Dvojité šifrovanie sa nepoužíva, dôvodom je útok popísaný v nasledujúcej stati.

„Meet in the middle“ útok

Meet in the middle („stretnutie uprostred“, v ďalšom texte MIM) je útok na viacnásobné šifrovanie, ktorý umožňuje znížiť časovú zložitosť útoku úplným preberaním na úkor zvýšenia pamäťových nárokov. Ilustrujme útok na dvojitom šifrovaní, kde $c = E_{k_2}(E_{k_1}(p))$. Nech dĺžka kľúča k_1 aj k_2 je l bitov. Máme k dispozícii niekoľko dvojíc otvoreného a šifrového textu $\langle p_i, c_i\rangle_{i=1}^{n}$, šifrovaných rovnakým kľúčom. Štandardný útok úplným preberaním skúša všetky možné dvojice kľúčov k_1, k_2 a testuje správnosť voľby, teda či $c_i = E_{k_2}(E_{k_1}(p_i))$, pre všetky $i = 1, \ldots, n$. Časová zložitosť útoku (počet výberov kľúčov) je v priemernom prípade $O(2^{2l})$ pri pamäťovej zložitosti $O(1)$.

MIM útok dokáže „preliať“ časť časovej zložitosti do pamäťovej. Pre dvojicu $\langle p_1, c_1\rangle$ najskôr vypočítame $D_{k_2}(c_1)$ pre všetky možné kľúče k_2. Získané dvojice $\langle D_{k_2}(c_1), k_2\rangle$ uložíme do hašovacej tabuľky podľa prvej súradnice. Následne skúšame šifrovať otvorený text p_1 všetkými kľúčmi k_1 a testujeme, či hodnota $E_{k_1}(p_1)$ nie je uložená v hašovacej tabuľke. Ak áno, našli sme kľúče k_1 a k_2, pre ktoré $E_{k_1}(E_{k_2}(p_1)) = c_1$. Prirodzene, takýchto dvojíc kľúčov môže byť viac. Nesprávne vylúčime otestovaním na dvojiciach $\langle p_i, c_i\rangle_{i=2}^{n}$. Časová zložitosť MIM útoku je v tomto prípade $O(2^l)$ pri pamäťovej zložitosti $O(2^l)$.

Útok je možné realizovať aj pri menšej pamäťovej kapacite, keď hašovacia tabuľka nemôže pojať všetky dešifrované texty. Nech K je množina všetkých kľúčov a $K' \subsetneq K$ je množina tých kľúčov, pre ktoré sú dvojice $\langle D_{k_2}(c_1), k_2\rangle$ uložené v hašovacej tabuľke. Vtedy v druhej časti útoku najskôr hľadáme v hašovacej tabuľke a ak v nej hodnotu $E_{k_1}(p_1)$ nenájdeme, prechádzame kľúče z $K \setminus K'$ úplným preberaním. Časová zložitosť MIM útoku bude v takomto prípade $O(2^l \cdot |K \setminus K'|)$.

Prezentovaný MIM útok možno ľahko zovšeobecniť na iné schémy viacnásobného šifrovania – napríklad pre EDE a EEE módy trojitého šifrovania s trojicou nezávislých kľúčov.

3.3 Módy činnosti

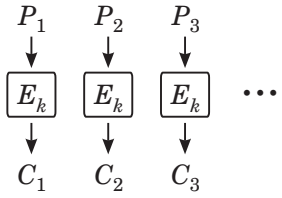
Základná šifrovacia transformácia jedného bloku môže byť v prípade dlhšieho otvoreného textu kombinovaná vo viacerých módoch. Každý z módov má svoje silné a slabé stránky a záleží na situácii a prostredí, ktorý je vhodné použiť. Spomenieme najznámejšie z nich. Pri prezentácií jednotlivých módov budeme bloky otvoreného textu označovať P_i a bloky šifrového textu C_i (pre $i \geq 1$).

ECB (Electronic Codebook)

ECB je základný mód, priamočiare použitie blokovej šifry. Bloky otvoreného textu sú šifrované nezávisle, samozrejme, s použitím rovnakého kľúča. Pre $i \geq 1$ prebieha šifrovanie a dešifrovanie takto:

$$C_i = E_k(P_i) \qquad \text{šifrovanie}$$
$$P_i = D_k(C_i) \qquad \text{dešifrovanie}$$

Šifrovanie v ECB móde je zobrazené na obrázku 3.3.



Obrázok 3.3: ECB mód

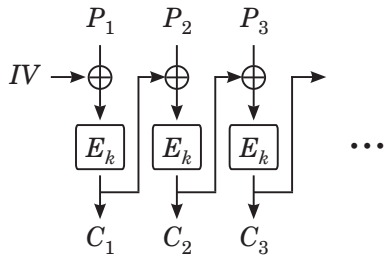
Vlastnosti. Rovnaké bloky otvoreného textu sú zašifrované do rovnakých blokov šifrového textu. Teda opakujúce sa úseky v otvorenom texte dokáže spozorovať ktokoľvek aj zo šifrového textu, pričom nemusí poznať kľúč. Bloky sú spracovávané nezávisle. Ak nie je zabezpečená integrita dát iným spôsobom, útočník môže nepozorovane niektoré bloky vynechať, zmeniť ich poradie a podobne. Uvedené vlastnosti sú dôvodom, prečo sa použitie ECB módu pri šifrovaní dlhších otvorených textov nedoporučuje. Chyba pri prenášaní šifrového textu, ak sa zmení 1 bit šifrového textu, ovplyvní po dešifrovaní len zodpovedajúci jeden blok otvoreného textu. Ak je bloková šifra „kvalitná“, znamená to zmenu každého bitu v tomto bloku s pravdepodobnosťou približne 1/2.

CBC (Cipher Block Chaining)

CBC mód odstraňuje niektoré bezpečnostné problémy ECB módu tak, že previaže šifrovanie bloku otvoreného textu s hodnotou predchádzajúceho bloku šifrového textu. Symbolom $\oplus$ označme sčítanie dvoch reťazcov po bitoch modulo 2 (inými slovami operáciu XOR).

$$C_i = E_k(P_i \oplus C_{i-1}) \quad \forall i \geq 1, \qquad \text{šifrovanie}$$
$$P_i = C_{i-1} \oplus D_k(C_i) \quad \forall i \geq 1, \qquad \text{dešifrovanie}$$

Hodnotu C_0 na začiatku šifrovania nemáme k dispozícii. CBC mód preto používa inicializačný vektor IV (reťazec bitov dĺžky rovnej dĺžke bloku). Položíme $C_0 = IV$. Priebeh šifrovania je zobrazený na obrázku 3.4.



Obrázok 3.4: CBC mód

Vlastnosti. Rovnaké otvorené texty šifrované s použitím rovnakého kľúča vedú k rôznym šifrovým textom, ak sú inicializačné vektory rôzne. Blok šifrového textu C_i závisí na bloku otvoreného textu P_i ako aj na všetkých predchádzajúcich blokoch $P_1, \ldots, P_{i-1}$. To znamená, že zmena poradia blokov šifrového textu ovplyvní dešifrovanie. Korektné dešifrovanie bloku si vyžaduje korektný predchádzajúci blok šifrového textu. Zmena bitu v šifrovom texte ovlyvní dva bloky otvoreného textu. Ak nastala zmena v C_i, ovplyvnený bude P_i (úplne) a P_{i+1} (len na pozícii zmeneného bitu). Na druhej strane zmena bitu v otvorenom texte ovlyvní (pri rovnakom kľúči a IV) zodpovedajúci aj všetky ďalšie bloky šifrového textu.

CBC má „samosynchronizujúcu“ vlastnosť, keď strata celého bloku šifrového textu (povedzme C_i) zapríčiní chybné dešifrovanie nasledujúceho bloku (C_{i+1}), ale ďalšie bloky sú už dešifrované správne. Strata časti textu, ktorý nie je násobkom dĺžky bloku, prirodzene, „pokazí“ dešifrovanie celého nasledujúceho textu (ako ostatne pri všetkých tu prezentovaných módoch).

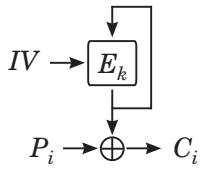
Inicializačný vektor nemusí byť utajený a obvykle sa generuje náhodne a prenáša ako prvý (nultý) blok šifrového textu. Dôležité je však zabezpečiť jeho integritu, keďže zmeny bitov v IV spôsobujú po dešifrovaní zmeny bitov v P_1 na rovnakých miestach.

OFB (Output Feedback)

OFB mód používa blokovú šifru ako synchrónnu prúdovú šifru. Môže teda slúžiť ako návod na vytvorenie prúdovej šifry z blokovej. Šifrovacia transformácia sa využíva len v rámci generátora prúdu blokov, ktoré sa pripočítavajú modulo 2 k blokom otvoreného textu. Vnútorný stav generátora v i. kroku označíme R_i a má dĺžku zhodnú s dĺžkou bloku. Poznamenajme, že v OFB móde nevyužívame invertovateľnosť E_k (existenciu dešifrovacej funkcie).

$$C_i = P_i \oplus R_i, \quad R_i = E_k(R_{i-1}) \quad \forall i \geq 1, \qquad \text{šifrovanie}$$
$$P_i = C_i \oplus R_i, \quad R_i = E_k(R_{i-1}) \quad \forall i \geq 1, \qquad \text{dešifrovanie}$$

Generátor prúdu blokov je inicializovaný na začiatku inicializačným vektorom ($R_0 = IV$). Inicializačný vektor môžeme poslať spolu so šifrovým textom v otvorenej podobe.



Obrázok 3.5: OFB mód

Vlastnosti. Podobne ako pri CBC, šifrovanie rovnakého otvoreného textu s rovnakým kľúčom pri rôznych IV vedie k rôznym šifrovým textom. Prúd generovaných blokov je nezávislý na otvorenom texte, to znamená, že ak šifrujeme ďalší text s rovnakým kľúčom, musíme zmeniť IV. V opačnom prípade útočník sčítaním šifrových textov získa súčet dvojice otvorených textov. Inicializačný vektor nemusí byť tajný (tajný je kľúč). Zmena (invertovanie) bitov v šifrovom texte sa prenáša ako zmena zodpovedajúcich bitov otvoreného textu, čo útočníkovi umožňuje ovplyvniť otvorený text želaným spôsobom bez jeho znalosti. Ak útočník pozná otvorený text, vie vypočítať prúd blokov R_i a skonštruovať šifrový text k otvorenému textu, ktorý si zvolí (s maximálnou dĺžkou nepresahujúcou dĺžku známeho prúdu blokov).

Pri strate bloku šifrového textu alebo akejkoľvek jeho časti je chybou ovplyvnené celé následné dešifrovanie.

Základnou požiadavkou pri prúdových šifrách je zabezpečiť dostatočne veľkú dĺžku periódy generovaného prúdu. Problém pri OFB móde by mohol nastať v prípade, ak by IV spolu s kľúčom viedli ku krátkej perióde generovaného prúdu blokov. Sú dve možnosti riešenia tohto problému. Prvou je nahradenie spätnej väzby vo výpočte R_i pripočítaním jednotky $R_i = R_{i-1} + 1$. Druhou je spoľahnúť sa na pravdepodobnosť, keďže pri náhodnej permutácii 2^n prvkov (dúfame, že transformácia E_k má charakter náhodného bijektívneho zobrazenia) je očakávaná dĺžka cyklu približne 2^{n-1} (kde n označuje dĺžku bloku).

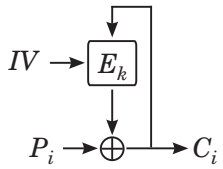
CFB (Cipher Feedback)

CFB mód vytvára z blokovej šifry prúdovú podobne ako OFB mód. CFB však konštruuje samosynchronizujúcu prúdovú šifru so spätnou väzbou zo šifrového textu. Blok otvoreného textu šifruje tak, že k nemu pripočíta šifrovanú podobu predchádzajúceho bloku šifrového textu. Pri CFB nevyužívame invertovateľnosť funkcie E_k.

$$C_i = P_i \oplus E_k(C_{i-1}) \quad \forall i \geq 1, \qquad \text{šifrovanie}$$
$$P_i = C_i \oplus E_k(C_{i-1}) \quad \forall i \geq 1, \qquad \text{dešifrovanie}$$

Keďže na začiatku nemáme k dispozícii C_0, výpočet opäť inicializujeme vektorom $C_0 = IV$.

Vlastnosti. Podobne ako pri CBC a OFB, šifrovanie rovnakého otvoreného textu s rovnakým kľúčom pri rôznych IV vedie k rôznym šifrovým textom. Inicializačný vektor nemusí byť tajný. Pri rovnakom IV a kľúči použitom na šifrovanie dvoch textov útočník získa súčet prvých blokov otvorených textov (ak sčíta prvé bloky



Obrázok 3.6: CFB mód

šifrových textov). Podobne ako pri CBC, závisí C_i na P_i ako aj na všetkých predchádzajúcich blokoch $P_1, \ldots, P_{i-1}$. Zmena poradia blokov šifrového textu ovplyvní dešifrovanie. Korektné dešifrovanie bloku si vyžaduje korektný predchádzajúci blok šifrového textu. Zmena bitu v šifrovom texte ovlyvní dva bloky otvoreného textu. Ak nastala zmena v C_i, ovplyvnený bude P_i (len na pozícii zmeneného bitu) a P_{i+1} (úplne). Na druhej strane zmena bitu v otvorenom texte ovlyvní (pri rovnakom kľúči a IV) zodpovedajúci aj všetky ďalšie bloky šifrového textu.

Podobne ako CBC má aj CFB „samosynchronizujúcu“ vlastnosť, keď strata celého bloku šifrového textu (povedzme C_i) zapríčiní chybné dešifrovanie nasledujúceho bloku (C_{i+1}), ale ďalšie bloky sú už dešifrované správne.

3.4 AES (Rijndael)

Prvým široko používaným blokovým šifrovacím algoritmom bol DES (Data Encryption Standard). DES bol štandardom používaným pre šifrovanie neklasifikovaných dát vo federálnych inštitúciach USA. Krátka dĺžka kľúča (56 bitov) však v súčasnosti dovoľuje útočiť na DES úplným preberaním. Náhradou a pokračovateľom algoritmu DES je AES (Advanced Encryption Standard) [AES01]. Výber nového blokového šifrovacieho algoritmu riadil NIST (National Institute of Standards and Technology). Viackolový proces výberu bol verejný a otvorený pre všetky návrhy algoritmov. Základné požiadavky boli:

- dĺžka bloku 128 bitov,
- variabilná dĺžka kľúča 128, 192 a 256 bitov.

V roku 2000 bol vybraný víťazný algoritmus Rijndael, ktorého autormi sú Vincent Rijmen a Joan Daemen – a stal sa štandardom AES. Dôvodom výberu algoritmu Rijndael bol vyvážený návrh, poskytujúci adekvátnu mieru bezpečnosti, výkonu, jednoduchosti implementácie a flexibility. Rijndeal je rýchly algoritmus v softvérovej aj hardvérovej implementácii, s nízkymi pamäťovými nárokmi.

V tejto časti je stručne popísaná základná konštrukcia algoritmu Rijndael, s cieľom ukázať štruktúru modernej blokovej šifry (nie však detailný popis).

Úvod

Rijndael je iterovaná šifra s dĺžkou bloku 128 bitov. Počet iterácií základnej transformácie závisí na dĺžke kľúča. Zodpovedajúce počty kôl pre dĺžky kľúča 128, 192 a 256 bitov sú 10, 12 a 14. Vnútorne je spracúvaný blok otvoreného alebo šifrového textu reprezentovaný ako dvojrozmerné pole bajtov 4×4. Bajty sú v poli, nazývanom stav

algoritmu, usporiadané takto:

0	4	8	12
1	5	9	13
2	6	10	14
3	7	11	15

Podobne ako iné iterované šifry, aj Rijndael vytvára zo šifrovacieho kľúča niekoľko podkľúčov. Podkľúče sa potom postupne používajú v jednotlivých kolách algoritmu.

Transformácia bloku otvoreného textu

Rijndeal transformuje blok otvoreného textu s využitím štyroch operácií:

- SubBytes – substitúcia bajtov. Každý bajt stavu algoritmu je nahradený novým bajtom podľa definovanej substitúcie $S : \{0,1\}^8 \to \{0,1\}^8$. Funkcia S je bijektívna (teda definuje permutáciu postupnosti 0, 1, ..., 255) a zabezpečuje, okrem iných vlastností, nelinearitu algoritmu.
- ShiftRows – cyklický posun riadkov stavu algoritmu. Každý riadok je posunutý doľava o rôzny počet bajtov (prvý riadok sa neposúva, ďalšie riadky postupne o jeden, dva a tri bajty):

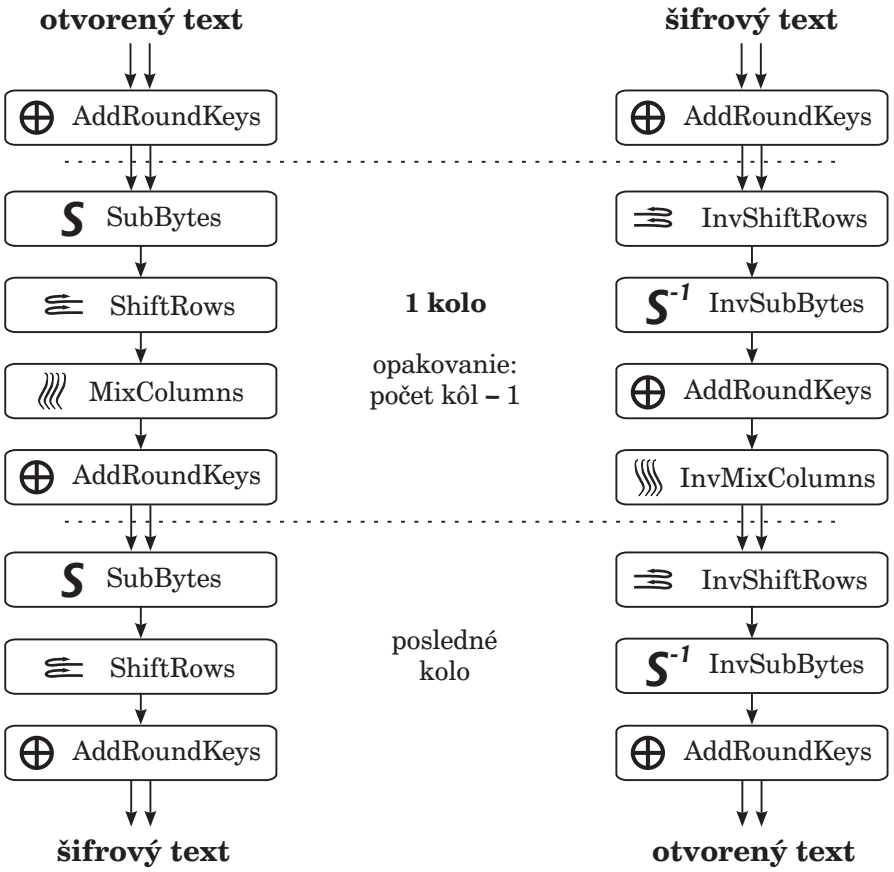
$$\begin{array}{|c|c|c|c|} \hline s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ \hline s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ \hline s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ \hline s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \\ \hline \end{array} \longrightarrow \begin{array}{|c|c|c|c|} \hline s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ \hline s_{1,1} & s_{1,2} & s_{1,3} & s_{1,0} \\ \hline s_{2,2} & s_{2,3} & s_{2,0} & s_{2,1} \\ \hline s_{3,3} & s_{3,0} & s_{3,1} & s_{3,2} \\ \hline \end{array}$$

- MixColumns – transformácia stĺpcov stavu algoritmu. Každý stĺpec (pozostávajúci z bajtov $s_{0,c}$, ..., $s_{3,c}$) sa nahradí novým stĺpcom podľa nasledujúceho predpisu:

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{pmatrix}$$

  V uvedenom maticovom násobení sú prvky oboch matíc interpretované ako prvky konečného poľa $\mathrm{GF}(2^8)$ generovaného ireducibilným polynómom $x^8 + x^4 + x^3 + x + 1$. Sčítanie je realizované ako jednoduchý XOR bajtov.
- AddRoundKey – pripočítanie podkľúča dĺžky 16 bajtov (128 bitov) k stavu algoritmu. Sčítanie je realizované ako XOR zodpovedajúcich bajtov podkľúča a stavu algoritmu.

Každé kolo algoritmu pozostáva z rovnakej postupnosti uvedených operácií, s výnimkou začiatku (kde sa pred prvé kolo vkladá operácia AddRoundKey) a posledného kola (kde sa vynecháva operácia MixColumns). Schematicky je postupnosť operácií znázornená na obrázku 3.7.

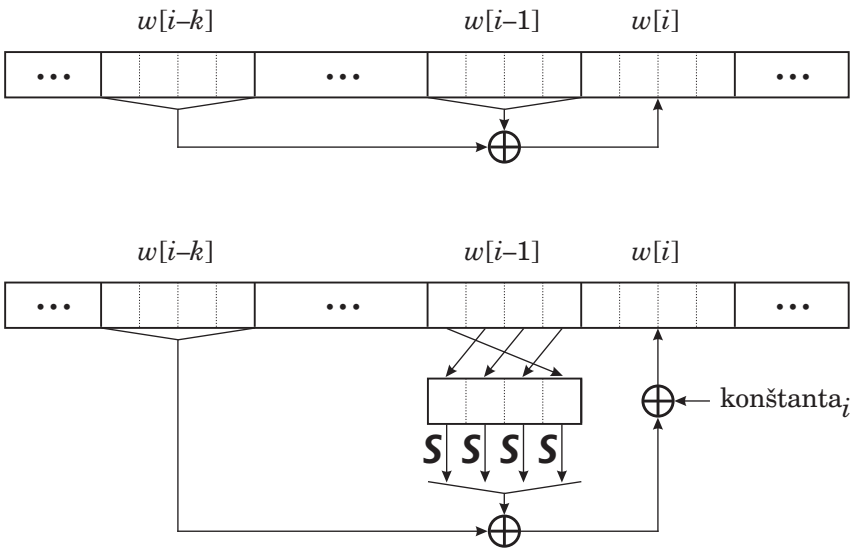


Obrázok 3.7: Operácie AES pri šifrovaní a dešifrovaní

Vytváranie podkľúčov

Postup vytvárania podkľúčov musí zohľadniť variabilnú dĺžku kľúča, rôzny počet kôl šifrovania/dešifrovania a teda aj rôzny počet potrebných podkľúčov. Rijndael označuje výrazom „slovo“ postupnosť 4 bajtov. Slová sú základné jednotky, s ktorými pracuje algoritmus na vytváranie podkľúčov. Algoritmus vytvára dostatočne dlhé pole slov (označme ho w), z ktorého sa zaradom berú podkľúče (teda dokopy musí byť v poli $4R$ slov, kde R označuje počet kôl).

Nech počet slov kľúča je k, teda pre kľúče dĺžky 128, 192 a 256 bitov je hodnota k postupne rovná 4, 6 a 8. Začiatok poľa w sa naplní šifrovacím kľúčom. Ďalšie slová v poli w, povedzme $w[i]$, sú počítané ako XOR slov $w[i-1]$ a $w[i-k]$. V prípade, že aktuálna pozícia slova (teda i) je deliteľná k, pred operáciou XOR sa vykoná transformácia slova $w[i-1]$. Transformácia pozostáva z cyklického posunu bajtov slova o jeden bajt doprava, nasledovanom substitúciou každého bajtu v slove pomocou funkcie S (rovnakej ako pri šifrovaní). K takémuto slovu sa navyše pripočítava definovaná konštanta. Výpočet poľa w schematicky ilustruje obrázok 3.8



Obrázok 3.8: Výpočet poľa slov pre podkľúče AES

Dešifrovanie

Transformácia bloku šifrového textu pri dešifrovaní využíva inverzné operácie k tým, ktoré boli použité pri šifrovaní. Výnimkou je operácia AddRoundKey, keďže pričítaním rovnakého podkľúča ako pri šifrovaní sa podkľúč „odstráni“.

- InvSubBytes – substitúcia bajtov stavu algoritmu, pričom sa použije inverzná funkcia (permutácia) S^{-1}.
- InvShiftRows – cyklický posun riadkov stavu algoritmu doprava (teda naopak ako pri šifrovaní). Prvý riadok sa neposúva, ostatné postupne o jeden, dva a tri bajty.
- InvMixColumns – transformácia stĺpcov stavu algoritmu s použitím matice inverznej k matici použitej pri šifrovaní.

Schematicky je postupnosť operácií pre dešifrovaní znázornená na obrázku 3.7. Samozrejme, poradie podkľúčov pre dešifrovaní je opačné ako pri šifrovaní.

4 Kryptografia s verejnými kľúčmi – úvod

Rozvoj kryptografických metód využívajúcich verejné kľúče (tzv. „public-key cryptography“) nastal po roku 1976, kedy Diffie a Hellman publikovali základnú myšlienku týchto metód [DH76]. Prvými aplikáciami boli konštrukcie asymetrického šifrovania a protokolu na dohodnutie kľúča. Princípy konštrukcií našli postupne uplatnenie aj v ďalších oblastiach. Digitálne podpisy, elektronické peniaze, voľby v sieti a ďalšie riešenia vychádzajú zo spoločných základov kryptografie s verejnými kľúčmi.

4.1 Asymetrické šifrovanie

Symetrické šifrovacie systémy vyžadujú, aby komunikujúce strany poznali kľúč, ktorý umožní úspešné šifrovanie a dešifrovanie. Predstavme si, že máme rozsiahlu počítačovú sieť s n účastníkmi, ktorí majú spolu zabezpečeným spôsobom komunikovať. Keďže potenciálne môže komunikovať ľubovoľná dvojica účastníkov, potrebujeme v sieti $\binom{n}{2}$ rôznych kryptografických kľúčov. Tieto kľúče treba dôveryhodným spôsobom vygenerovať a distribuovať účastníkom (každý účastník má mať $n-1$ kľúčov). Ďalšie súvisiace problémy – bezpečné uloženie kľúčov, ich periodická zmena, pridávanie nových účastníkov naznačujú, že v tomto prípade bude manažment kryptografických kľúčov náročnou úlohou.

Základný princíp

Pokúsme sa vysvetliť najskôr základný princíp asymetrických šifrovacích systémov na príklade. Použijeme telefónny zoznam veľkého mesta. Ten je bežne k dispozícii každému a je utriedený podľa priezviska a mena osôb, ktoré majú v tomto meste telefón. Predstavme si, že iba my máme k dispozícii aj „invertovaný“ zoznam – utriedený podľa telefónnych čísel. Pomocou telefónneho zoznamu nám ktokoľvek môže poslať šifrovanú správu. Šifrovanie prebieha nasledovne: pre každé písmeno otvoreného textu vyberieme náhodne priezvisko začínajúce týmto písmenom a do šifrového textu zapíšeme príslúchajúce telefónne číslo. Dešifrovanie je pri použití invertovaného zoznamu priamočiare, bez neho je to však pomerne náročná úloha.

Formálnejšie: Asymetrický šifrovací systém je dvojica funkcií – verejná a súkromná. Obe tieto funkcie sú skonštruované (zvolené) používateľom. Verejnú funkciu, ako už názov napovedá, používateľ zverejní a je k dispozícii komukoľvek. Súkromnú funkciu si ponechá. Verejná funkcia slúži na šifrovanie a súkromná na dešifrovanie. Teda šifrovať môže každý, dešifrovať len vlastník súkromnej funkcie. Niekedy sa asymetrický šifrovací systém prezentuje ako trieda funkcií parametrizovaných kľúčom (kľúčmi). Potom namiesto o verejnej a súkromnej funkcii hovoríme o verejnom a súkromnom kľúči. V našom príklade bol verejným kľúčom telefónny zoznam a súkromným kľúčom invertovaný telefónny zoznam.

Označme množinu otvorených textov P, množinu šifrových textov C a nech R označuje množinu $\{0,1\}^*$. Nech $E : P \times R \to C$ je verejná funkcia a $D : C \to P$ je súkromná funkcia. Význam množiny R vo funkcii E spočíva v umožnení náhodnej voľby pri šifrovaní, ako sme to videli aj v našom príklade s telefónnym zoznamom. V takom prípade je otvorený text zašifrovaný do jedného z niekoľkých potenciálnych šifrových textov. Niektoré asymetrické systémy náhodnosť pri šifrovaní nepoužívajú (sú deterministické – napríklad RSA), iné ju vyžadujú (napr. ElGamalov systém). Systémy, v ktorých jednému otvorenému textu môže prislúchať viacero šifrových textov, budeme nazývať znáhodnené.

Asymetrický šifrovací systém musí spĺňať nasledujúce vlastnosti, aby mohol byť použiteľný opísaným spôsobom:

1. (**korektnosť**) Dešifrovanie šifrového textu vedie k pôvodnému otvorenému textu:

$$\forall m \in P \; \forall r \in R : \; D(E(m,r)) = m.$$

2. (**realizovateľnosť**) Funkcie E a D sú algoritmicky efektívne realizovateľné. Zároveň je efektívna aj ich konštrukcia používateľom. „Efektívne“ znamená obvykle s deterministickou (prípadne s pravdepodobnostnou) polynomiálnou časovou zložitosťou.
3. (**bezpečnosť**) Zo znalosti E je prakticky nemožné určiť funkciu D^* takú, že D^* je efektívne realizovateľná (pozri 2) a pre nezanedbateľné množstvo $c \in C$: $D^*(c) = D(c)$. Jednoduchšie povedané, má byť veľmi ťažké len zo znalosti E úspešne túto funkciu invertovať. Prirodzene, úplne vylúčiť úspešné invertovanie E nie je možné, vždy je nenulová pravdepodobnosť uhádnutia D. Čo konkrétne znamená „prakticky nemožné“ a „nezanedbateľné množstvo“ závisí od našich bezpečnostných požiadaviek na systém.

V ďalších častiach sa budeme zaoberať niektorými asymetrickými systémami a otázkami súvisiacimi s ich korektnosťou, realizovateľnosťou a bezpečnosťou na konkrétnejšej a praktickejšej úrovni.

Hybridné šifrovanie

V súčasnosti používané asymetrické systémy sú pri šifrovaní (aj dešifrovaní) podstatne pomalšie ako symetrické. V situáciach, keď je rýchlosť šifrovania alebo dešifrovania aplikačne dôležitou požiadavkou (to je obvykle vždy), je použitie výlučne asymetrických systémov nemožné. Výhody asymetrických a symetrických šifrovacích systémov spája tzv. hybridné šifrovanie.

V hybridnom šifrovaní sa na šifrovanie dát použije symetrický systém s náhodne vygenerovaným kľúčom. Asymetrický systém slúži na zašifrovanie tohto kľúča použitím verejnej funkcie adresáta. Adresát po prijatí šifrových textov najskôr dešifruje súkromnou funkciou symetrický kľúč a ten následne použije na dešifrovanie dát.

Označme E_A, D_A verejnú a súkromnú funkciu používateľa A, ktorý je zároveň adresátom správy m. Nech E a D sú šifrovací a dešifrovací algoritmus nejakého symetrického systému. Pri hybridnom šifrovaní správy m pre používateľa A postupujeme takto:

1. Zvolíme náhodne symetrický kľúč k. Používateľovi A pošleme dvojicu:

   $$\langle E_A(k, r), E_k(m)\rangle,$$

   kde r je náhodne zvolené z R.
2. Používateľ A dešifruje symetrický kľúč:

   $$D_A(E_A(k, r)) = k$$

   a následne aj správu: $D_k(E_k(m)) = m$.

Prirodzene, bezpečnosť hybridného šifrovania závisí podstatne od bezpečnosti oboch použitých systémov, tak asymetrického ako aj symetrického. „Rozbitie“ ľubovoľného z nich vedie k rozbitiu celého hybridného systému.

4.2 Kryptografické protokoly

V elektronickom prostredí chceme vytvoriť objekty a procedúry zaužívané v reálnom svete. Príkladom môžu byť podpisy, peniaze, voľby, atď. Ich vlastnosti a mechanizmy využívané v reálnom svete, zaručujúce ich bezpečnosť a teda použiteľnosť, sú zvyčajne v elektronickom prostredí nepoužiteľné. Preto je potrebné vytvoriť elektronické ekvivalenty takýchto objektov a procedúr a ich použiteľnosť zabezpečiť, okrem iného, novými kryptografickými metódami. Riešenia používajúce výlučne symetrické šifrovacie systémy buď neexistujú, alebo sú príliš neefektívne.

Použiteľné konštrukcie riešení naznačených problémov sú založené na asymetrických šifrovacích systémoch a ďalších kryptografických primitívach, ako sú napríklad jednosmerné funkcie (one-way functions), kryptografické hašovacie funkcie alebo schémy na zdieľanie tajomstva.

5 RSA

RSA je jedným z najznámejších a najpoužívanejších asymetrických šifrovacích systémov. Názov je vytvorený zo začiatočných písmen mien jeho autorov – Ronald Rivest, Adi Shamir a Leonard Adleman, ktorí ho zverejnili v roku 1978.

5.1 Inicializácia

Inicializácia je proces vytvorenia vlastnej inštancie RSA – súkromného a verejného kľúča. Popíšme podrobne postup inicializácie.

1. Zvolíme dve (dostatočne veľké) prvočísla p a q ($p \neq q$). Položíme $n = p \cdot q$.
2. Vyberieme prirodzené číslo e také, že $1 < e < \phi(n)$ a $\mathrm{nsd}(e, \phi(n)) = 1$, kde $\phi(n) = (p-1)(q-1)$ je Eulerova funkcia a nsd označuje najväčší spoločný deliteľ svojich argumentov. Teda e je nesúdeliteľné s $\phi(n)$.
3. Vypočítame d také, že $e \cdot d \equiv 1 \pmod{\phi(n)}$.

Aké prvočísla považujeme za veľké, závisí od efektívnosti metód faktorizácie a miere bezpečnosti, ktorú od nášho systému požadujeme. V súčasnosti sa za veľké považujú aspoň 512 bitové prvočísla, aby sme po ich vynásobení dostali aspoň 1024 bitov dlhý modulus n. Bezpečnosti RSA sa budeme podrobnejšie venovať v časti 5.4.

Verejný kľúč je dvojica čísel $\langle e, n \rangle$. Súkromný kľúč tvorí hodnota d. Parameter d sa niekedy nazýva súkromný exponent a parameter e verejný exponent. Prvočísla p, q nie sú pre používanie RSA potrebné, môžeme na ne po inicializácii zabudnúť. Je dôležité, ako uvidíme neskôr, aby sa prvočísla nedostali do rúk potenciálnych útočníkov.

Priestor otvorených aj šifrových textov je množina $\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$.

šifrovanie: $E(m) = m^e \bmod n$
dešifrovanie: $D(c) = c^d \bmod n$

Pri výpočte dešifrovacej transformácie potrebujeme okrem súkromného kľúča d poznať aj hodnotu n. Tá je súčasťou verejného kľúča a teda všeobecne známa.

Príklad. 1. Nech $p = 5$ a $q = 11$. Prirodzene, tieto prvočísla nie sú dostatočne veľké pre praktické použitie, slúžia pre náš ilustračný príklad. Teda $n = 5 \cdot 11 = 55$.

2. Máme $\phi(n) = (5-1)(11-1) = 40$. Zvoľme $e = 3$, nesúdeliteľné s $\phi(n)$.
3. Zodpovedajúce $d = 27$, lebo $d \cdot e \equiv 27 \cdot 3 \equiv 81 \equiv 1 \pmod{40}$.

Priestor správ je množina $\{0, \ldots, 54\}$. Verejný kľúč je $(3, 55)$ a súkromný kľúč 27. Nech je napríklad správa $m = 14$. Potom $E(m) = 14^3 \bmod 55 = 49$ a $D(49) = 49^{27} \bmod 55 = 14$.

Pokiaľ chceme šifrovať dlhý text, rozdelíme ho na bloky a tieto šifrujeme samostatne. Takže ak n je 1024 bitové číslo, tak môžeme text deliť na 1023 bitové bloky a tieto samostatne šifrovať – šifrový text potom pozostáva z blokov dĺžky najviac 1024 bitov. RSA má potom povahu blokovej šifry.

5.2 Korektnosť

V tejto časti ukážeme matematickú korektnosť RSA. To znamená, že po dešifrovaní šifrového textu dostaneme opäť pôvodný otvorený text.

Veta 1 (Korektnosť RSA). *Pre ľubovoľnú inštanciu RSA systému platí:*

$$\forall m \in \mathbb{Z}_n : \; D(E(m)) = m.$$

Dôkaz. Nech e a d sú verejný a súkromný exponent v inštancii RSA systému s $n = p \cdot q$. Potrebujeme ukázať, že $(m^e \bmod n)^d \bmod n = m$, pre všetky $m \in \mathbb{Z}_n$.

Špeciálny prípad je $m = 0$. Vtedy $E(m) = 0$ a $D(0) = 0$, preto tvrdenie platí. Pre $m \in \mathbb{Z}_n \setminus \{0\}$ budeme uvažovať dva prípady: $\mathrm{nsd}(n, m) = 1$ a $\mathrm{nsd}(n, m) \neq 1$. Vieme, že $ed \equiv 1 \pmod{\phi(n)}$. Teda $\exists k \in \mathbb{N} : \; ed = 1 + k\phi(n)$.

1. $\mathrm{nsd}(n, m) = 1$. Počítajme:

$$\begin{aligned} D(E(m)) &= (m^e \bmod n)^d \bmod n \\ &= m^{ed} \bmod n \\ &= m^{1+k\phi(n)} \bmod n \\ &= m \cdot (m^{\phi(n)})^k \bmod n \\ &= m \bmod n = m. \end{aligned}$$

   Predposledná rovnosť vyplýva z Eulerovej vety.

2. $\mathrm{nsd}(n, m) \neq 1$. Potom buď $p \mid m$ alebo $q \mid m$ (nie však obe súčasne, lebo $0 < m < n$). Bez ujmy na všeobecnosti budeme predpokladať, že $m = l \cdot p^s$, kde $s \geq 1$ a $\mathrm{nsd}(l, n) = 1$ $(s, l \in \mathbb{N})$. Potom

$$\begin{aligned} D(E(m)) &= m^{ed} \bmod n \\ &= (lp^s)^{1+k\phi(n)} \bmod n \\ &= l \cdot (p^{1+k\phi(n)})^s \bmod n. \end{aligned} \tag{5.1}$$

Podľa malej Fermatovej (Eulerovej) vety $p^{q-1} \equiv 1 \pmod{q}$. Odtiaľ:

$$\begin{aligned} p^{(q-1)(p-1)} &\equiv 1 \pmod{q} \\ p^{k\phi(n)} &= 1 + aq \qquad \text{pre nejaké } a \in \mathbb{Z} \\ p^{k\phi(n)+1} &= p + apq = p + an \\ p^{k\phi(n)+1} &\equiv p \pmod{n}. \end{aligned}$$

Po dosadení do (5.1) dostaneme:

$$D(E(m)) = lp^s \bmod n = m.$$

□

5.3 Realizovateľnosť

Postupne prejdeme jednotlivými krokmi inicializácie RSA a rozoberieme ich realizovateľnosť.

Generovanie prvočísel

V prvom kroku inicializácie RSA potrebujeme vygenerovať dve veľké prvočísla. Generovanie prebieha nasledovne:

1. Zvolíme náhodné nepárne číslo požadovanej veľkosti.
2. Otestujeme, či je prvočíslom. Ak nie, opakujeme postup.

Prvým problémom pri tomto postupe je počet opakovaní voľby nepárneho čísla, kým narazíme na prvočíslo. Odpoveďou je Prvočíselná veta (Prime Number Theorem):

$$\lim_{n\to\infty} \frac{\pi(n)}{n/\ln n} = 1,$$

kde $\pi(n)$ označuje počet prvočísel nie väčších ako n. Pre veľkosť hodnoty $\pi(n)$ platí napríklad nasledujúci odhad [GKP94, strana 111]:

$$\ln n - \frac{3}{2} < \frac{n}{\pi(n)} < \ln n - \frac{1}{2}, \qquad \text{pre } n \geq 67.$$

Takže pri generovaní 512 bitového prvočísla (≈ 154 ciferné číslo v desiatkovej sústave) je pravdepodobnosť, že náhodne zvolené nepárne 512 bitové číslo je prvočíslom, väčšia ako

$$2 \cdot \left(\frac{2^{513}}{\ln 2^{513} - \frac{1}{2}} - \frac{2^{512}}{\ln 2^{512} - \frac{3}{2}} \right) / 2^{512} \doteq 0.005605.$$

Teda priemerne treba voliť približne 178 krát.

Druhý problém je testovanie prvočíselnosti zvoleného čísla. Prakticky používané algoritmy sú pravdepodobnostné. To znamená, že s určitou nenulovou pravdepodobnosťou, ktorú však vieme ovplyvniť, môžu vypočítať chybný výsledok. Existuje viacero takýchto algoritmov, my si ukážeme Millerov-Rabinov test prvočíselnosti.

Millerov-Rabinov test

vstup:
n – nepárne číslo, pričom $n - 1 = 2^s \cdot t$, kde t je nepárne
k – počet opakovaní testu

for $i \leftarrow 0, \ldots, k-1$
  zvolíme náhodne $b \in_R \mathbb{Z}_n^*$;
  if $(b^t \not\equiv \pm 1 \pmod{n})$
    $l \leftarrow b^t \bmod n$;
    for $j \leftarrow 1, \ldots, s-1$
      $l \leftarrow l^2 \bmod n$;
      if $(l = -1)$ **break** ;
    end for
    if $(l \neq -1)$ **return** „n je zložené číslo“;
end for
return „n je prvočíslo“;

Matematicky sa dá vyjadriť fakt, že pre dané n a zvolené b ($b \in \mathbb{Z}_n^*$) prejde n testom (jedna iterácia) s výsledkom „n je prvočíslo“ takto:

$$b^t \bmod n = \pm 1 \quad \vee \quad \exists j \in \{1, \ldots, s-1\} : \; b^{t \cdot 2^j} \bmod n = -1. \tag{5.2}$$

Pri splnení vlastnosti (5.2) sa hovorí, že n je silným pseudoprvočíslom vzhľadom na bázu b. Millerov-Rabinov test je pravdepodobnostný algoritmus, keďže niekedy môže dať nesprávnu odpoveď. Vlastnosti tohto algoritmu vyplývajú z nasledujúcej vety [Kob87, strana 117].

Veta 2. *Ak n je zložené nepárne číslo, tak vlastnosť* (5.2) *platí pre najviac štvrtinu všetkých $b \in \mathbb{Z}_n^*$.*

Millerov-Rabinov test dáva takéto výsledky:

- ak je n prvočíslo – odpoveď je vždy správna: „n je prvočíslo“;
- ak je n zložené nepárne číslo – pravdepodobnosť omylu, t.j. odpovede „n je prvočíslo“, je najviac $\frac{1}{4^k}$.

Pravdepodobnosť omylu s rastúcim k veľmi rýchlo (exponenciálne) klesá. Môžeme si stanoviť požadovanú nenulovú úroveň omylu a vypočítať hodnotu parametra k tak, aby sme ju dosiahli.

Poznámka. V roku 2002 bol objavený deterministický polynomiálny algoritmus na testovanie prvočíselnosti [AKS02]. Jeho časová zložitosť je však pomerne vysoká. Preto je na testovanie prvočíselnosti vhodnejšie zatiaľ použiť niektorý z pravdepodobnostných algoritmov.

Modulárne umocňovanie

Pri inicializácii RSA ako aj pri samotnom šifrovaní a dešifrovaní sa používa operácia umocňovania v modulárnej aritmetike. To znamená, že potrebujeme efektívny algoritmus, ktorý počíta $v = a^b \bmod n$, pričom čísla a, b, n môžu mať niekoľko sto alebo tisíc bitov.

Nech $b = (b_k \dots b_1 b_0)_2$ je reprezentácia čísla b v binárnej sústave. Najjednoduchší algoritmus na modulárne umocňovanie (prirodzene, deterministický a polynomiálny), vyzerá nasledovne:

Modulárne umocňovanie

Vstup: $a, b, n \in \mathbb{N}$
Výstup: $v = a^b \bmod n$

$v \leftarrow 1$; $p \leftarrow a$;
for $i \leftarrow 0, \dots, k$
  if $(b_i = 1)$ $v \leftarrow p \cdot v \bmod n$;
  $p \leftarrow p^2 \bmod n$;
end for
return v;

Operácie mod a prevod do binárnej sústavy je tiež možné robiť efektívne. Upravený algoritmus s prevodom do binárnej sústavy ukrytým vo vnútri:

Upravené modulárne umocňovanie

Vstup: $a, b, n \in \mathbb{N}$
Výstup: $v = a^b \bmod n$

$v \leftarrow 1$; $p \leftarrow a$;
while $b > 0$
  if (b je nepárne) $v \leftarrow p \cdot v \bmod n$;
  $p \leftarrow p^2 \bmod n$;
  $b \leftarrow b/2$; // celočíselné delenie
end while
return v;

Konštrukcia parametrov e a d

Verejný exponent e môžeme konštruovať niekoľkými spôsobmi. Môžeme e voliť ako náhodné nepárne číslo a testovať, či $\mathrm{nsd}(e, \phi(n)) = 1$, pričom postup opakujeme, kým vhodné e nenájdeme. Zvyčajne sa však verejný exponent vyberá cielene, s ohľadom na efektívnosť šifrovania (t.j. výpočet funkcie E). Možnými kandidátmi sú čísla s krátkou binárnou reprezentáciou alebo s binárnou reprezentáciou v špeciálnom tvare. Napríklad malé prvočísla 3, 7, 11, 13, alebo číslo $65537 = (10000000000000001)_2$.

Súkromný exponent d sa vypočíta z e pomocou rozšíreného Euklidovho algoritmu. Rozšírený Euklidov algoritmus pre $a > b$ počíta $u, v \in \mathbb{Z} :\ ub + va = \mathrm{nsd}(a, b)$. Ak položíme $a = \phi(n)$ a $b = e$, tak dostaneme u, v: $ue + v\phi(n) = \mathrm{nsd}(e, \phi(n))$. To zna-

mená, že $ue \equiv 1 \pmod{\phi(n)}$. Tajný exponent d dostaneme úpravou u – pripočítaním vhodného celočíselného násobku $\phi(n)$, napríklad ak je u záporné.

Rýchly výpočet dešifrovacej funkcie

Snaha o urýchlenie šifrovacej alebo dešifrovacej funkcie je vzhľadom na modulárne operácie s dlhými číslami prirodzená. Urýchlenie šifrovacej funkcie sa (okrem iného) dosahuje voľbou malého verejného exponentu e. Keďže súkromný exponent d sa počíta z e a $\phi(n)$, jeho štruktúru nemôžeme príliš ovplyvniť (v skutočnosti môžeme tým, že najskôr zvolíme d s požadovanou štruktúrou a dopočítame e, čo sa však z bezpečnostných dôvodov neodporúča).

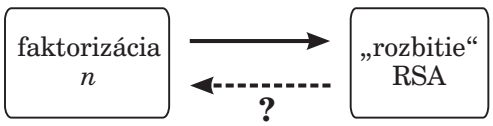
Urýchlenie výpočtu $D(c)$ spočíva zvyčajne vo využití Čínskej zvyškovej vety. Vtedy si ako súkromnú informáciu pamätáme nielen d, ale aj prvočísla p, q. Dešifrovanie šifrového textu $c \in \mathbb{Z}_n$ prebieha tak, že najskôr vypočítame $a_1 = c^d \bmod p$ a $a_2 = c^d \bmod q$. Otvorený text m dostaneme kombináciou hodnôt a_1 a a_2 (pozri dôkaz Čínskej zvyškovej vety):

$$m = (a_1\, q\, (q^{-1} \bmod p) + a_2\, p\, (p^{-1} \bmod q)) \bmod n. \tag{5.3}$$

Na prvý pohľad sa môže zdať, že jedno modulárne umocnenie sme nahradili dvoma ($c^d \bmod p$ a $c^d \bmod q$), takže dešifrovanie sme neurýchlili. Dve realizované umocnenia však majú kratšie moduly, preto prebehnú rýchlejšie ako jedno umocnenie s dlhým modulom n. Následná lineárna kombinácia (5.3) je už len elementárnym výpočtom. Poznamenajme, že hodnoty $q\,(q^{-1} \bmod p)$ a $p\,(p^{-1} \bmod q)$ môžu byť predvypočítané a nie je potrebné ich pri každom výpočte dešifrovacej funkcie znova počítať.

5.4 Bezpečnosť RSA***

Ľahko vidieť, že bezpečnosť RSA závisí na zložitosti problému faktorizácie. Hodnota n je verejne známa, rovnako ako verejný exponent e. Ak by bolo možné n ľahko faktorizovať, t.j. vypočítať prvočísla p a q, tak ktokoľvek vie vypočítať hodnotu súkromného exponentu d rovnakým spôsobom, ako sme to z hodnôt e a $\phi(n)$ urobili pri inicializácii systému. So znalosťou súkromného kľúča už nie je problém dešifrovať ľubovoľný šifrový text.



Obrázok 5.1: Vzťah bezpečnosti RSA a problému faktorizácie

Teda ak vieme faktorizovať n, vieme „rozbiť“ RSA. Opačné tvrdenie je otvorený problém. Zatiaľ nikto nedokázal, že rozbitie RSA je ekvivalentné (musí viesť k) faktorizácii n. Môže existovať spôsob, ako efektívne zisťovať otvorený text zo šifrového textu bez znalosti súkromného kľúča bez toho, aby sme faktorizovali n (presnejšie, faktorizácia n ostane naďalej ťažký problém). Treba však podotknúť, že takýto spôsob útoku sa musí zaobísť bez hodnôt $\phi(n)$ a d, pretože znalosť ľubovoľnej z nich je ekvivalentná faktorizácii n, ako uvidíme v ďalších statiach.

Poznamenajme, že ak by bol dôkaz ekvivalencie (presnejšie dôkaz otvorenej implikácie) rozbitia RSA a faktorizácie konštruktívny, viedlo by to k neodolnosti RSA voči útoku s možnosťou voľby šifrového textu (CCA). Existujú systémy, pre ktoré sa dá dokázať ekvivalencia medzi schopnosťou útočníka efektívne dešifrovať a problémom faktorizácie, napríklad Rabinov systém (pozri kapitolu 7) alebo konštrukcie [LKBS92, Wil80]. Práve „vďaka“ konštruktívnemu dôkazu[1] sú však neodolné voči CCA.

Zaujímavý výsledok dosiahli Boneh a Venkatesan [BV98], keď ukázali, že rozbitie RSA s malým verejným exponentom nemôže byť ekvivalentné faktorizácii, pokiaľ nie je faktorizácia ľahký problém v istej, pomerne obmedzenej triede algoritmov.

Faktorizácia n pri znalosti $\phi(n)$

Ak ktokoľvek pozná $\phi(n)$, tak vie nájsť prvočísla p, q riešením sústavy dvoch rovníc o dvoch neznámych:

$$\begin{aligned} p \cdot q &= n, \\ (p-1)(q-1) &= \phi(n). \end{aligned}$$

Faktorizácia n pri znalosti e a d

Znalosť e môžeme vždy predpokladať, lebo e je súčasťou verejného kľúča rovnako ako n. Predtým, ako popíšeme postup faktorizácie, dokážeme pomocné lemy.

Lema 1. *Nech $n, m \in \mathbb{N}$. Nech existuje celé číslo a také, že $\mathrm{nsd}(a, n) = 1$ a zároveň $a^m \not\equiv 1 \pmod{n}$. Nech $b \in \mathbb{Z}$ je náhodne vybrané číslo také, že $\mathrm{nsd}(b, n) = 1$. Potom pravdepodobnosť toho, že $b^m \equiv 1 \pmod{n}$ je najviac $\frac{1}{2}$.*

Dôkaz. Lemu stačí dokázať pre $a, b \in \mathbb{Z}_n$, lebo počítame modulo n. Nech $\{c_1, \ldots, c_k\} \subseteq \mathbb{Z}_n$ sú všetky prvky také, že $\mathrm{nsd}(c_i, n) = 1$ a $c_i^m \equiv 1 \pmod{n}$. Hľadaná pravdepodobnosť (označme ju Pr) je $\Pr = \frac{k}{\phi(n)}$.

Pre hodnoty $ac_i \bmod n$ $(1 \leq i \leq k)$ platí:

- $ac_i \bmod n$ je nesúdeliteľné s n, lebo a aj c_i sú nesúdeliteľné s n;
- $(ac_i \bmod n)^m \equiv a^m c_i^m \equiv a^m \not\equiv 1 \pmod{n}$;
- $\forall\, j \neq i$: $ac_i \bmod n \neq ac_j \bmod n$, lebo a je nesúdeliteľné s n a hodnoty c_i, c_j sú rôzne.

Takže prenásobením všetkých prvkov množiny $\{c_1, \ldots, c_k\}$ hodnotou a (modulo n) dostávame k navzájom rôznych prvkov, ktoré sú nesúdeliteľné s n a ich m-tá mocnina modulo n nie je 1. Pritom by mohli existovať ešte ďalšie prvky s touto vlastnosťou, teda ich celkový počet je aspoň k. Odtiaľ dostávame $k \leq \frac{\phi(n)}{2}$ a preto pre hľadanú pravdepodobnosť platí $\Pr = \frac{k}{\phi(n)} \leq \frac{1}{2}$. □

Lema 2. *Nech $n \in \mathbb{N}$ je deliteľné prvočíslom p. Nech $m \in \mathbb{N}$ je také, že platí:*

$$\forall x \in \mathbb{Z} : \ \mathrm{nsd}(x, n) = 1 \ \Rightarrow \ x^m \equiv 1 \pmod{n},$$

[1] spôsobom „ak máme možnosť efektívne dešifrovať, tak potom faktorizáciu efektívne vypočítame takýmto algoritmom ...“

pričom $(p-1) \nmid \frac{m}{2}$. *Nech zároveň existuje celé číslo* a *také, že* $\mathrm{nsd}(a,n) = 1$ *a zároveň* $a^{m/2} \not\equiv 1 \pmod{n}$. *Potom pre náhodne zvolené* $b \in \mathbb{Z}$ *také, že* $\mathrm{nsd}(b,n) = 1$, *nadobúda výraz* $b^{m/2} \bmod p$ *hodnotu* 1 *a* -1 *s rovnakou pravdepodobnosťou* $\frac{1}{2}$.

Dôkaz. Lemu stačí dokázať pre $b \in \mathbb{Z}_p$, lebo počítame modulo p. Z vlastnosti uvedenej v predpoklade lemy vyplýva, že m je párne (stačí vziať $x = -1$), teda $m/2$ je celé číslo. Keďže $b^m \equiv 1 \pmod{n}$, tak $b^m \equiv 1 \pmod{p}$ a teda

$$p \mid (b^{m/2})^2 - 1 \;\Rightarrow\; p \mid (b^{m/2}+1)(b^{m/2}-1).$$

To znamená, že 1 a -1 sú jediné prípustné hodnoty pre výraz $b^{m/2} \bmod p$.

Rozdeľme množinu $\mathbb{Z}_p \backslash \{0\}$ na dve množiny $\mathbb{Z}_p^{(+1)}$ a $\mathbb{Z}_p^{(-1)}$ podľa toho, akú hodnotu nadobúda príslušná mocnina prvku:

$$\mathbb{Z}_p^{(+1)} = \{b \in \mathbb{Z}_p \mid \mathrm{nsd}(b,n) = 1 \;\&\; b^{m/2} \bmod p = 1\}$$
$$\mathbb{Z}_p^{(-1)} = \{b \in \mathbb{Z}_p \mid \mathrm{nsd}(b,n) = 1 \;\&\; b^{m/2} \bmod p = -1\}$$

Prvok a z predpokladu lemy patrí do množiny $\mathbb{Z}_p^{(-1)}$. Označme pre množinu $X = \{x_1, \ldots, x_k\}$ zápisom $a \cdot X$ množinu $\{ax_1 \bmod p, \ldots, ax_k \bmod p\}$. Ľahko možno ukázať nasledujúce vzťahy:

$$a \cdot \mathbb{Z}_p^{(-1)} \subseteq \mathbb{Z}_p^{(+1)}$$
$$a \cdot \mathbb{Z}_p^{(+1)} \subseteq \mathbb{Z}_p^{(-1)}$$

Keďže $\mathrm{nsd}(a,n) = 1$, tak $|a \cdot \mathbb{Z}_p^{(-1)}| = |\mathbb{Z}_p^{(-1)}|$ a $|a \cdot \mathbb{Z}_p^{(+1)}| = |\mathbb{Z}_p^{(+1)}|$. S využitím predchádzajúcich inklúzií dostávame:

$$\begin{array}{l} |\mathbb{Z}_p^{(-1)}| = |a \cdot \mathbb{Z}_p^{(-1)}| \le |\mathbb{Z}_p^{(+1)}| \\ |\mathbb{Z}_p^{(+1)}| = |a \cdot \mathbb{Z}_p^{(+1)}| \le |\mathbb{Z}_p^{(-1)}| \end{array} \quad \Longrightarrow \quad |\mathbb{Z}_p^{(-1)}| = |\mathbb{Z}_p^{(+1)}|,$$

odkiaľ bezprostredne vyplýva tvrdenie lemy. □

Postup pri faktorizácii n:

1. Položme $m = e \cdot d - 1$. Potom $m = k\phi(n)$ pre nejaké $k \in \mathbb{N}$. Nasledujúca vlastnosť vyplýva priamočiaro z Eulerovej vety:

$$\forall a \in \mathbb{Z} : \mathrm{nsd}(a,n) = 1 \;\Rightarrow\; a^m \equiv 1 \pmod{n}. \tag{5.4}$$

2. Keďže (5.4) platí, musí byť m párne, stačí vziať napríklad $a = -1$ (t.j. $a = n-1$). Otestujeme, či vlastnosť (5.4) platí aj pre $m' = \frac{m}{2}$. Ak áno, tak položíme $m = m'$ a opäť vykonáme krok 2. Ak nie, pokračujeme 3. krokom.
   Testovanie vlastnosti (5.4) pre m' je pravdepodobnostné. Budeme voliť l náhodných kandidátov a, nesúdeliteľných s n. Ak pre všetkých kadidátov a platí: $a^{m'} \equiv 1 \pmod{n}$, usúdime, že vlastnosť (5.4) platí. Vďaka leme 1 je pravdepodobnosť mylného úsudku $\le 2^{-l}$. Ak aspoň pre jedno a je $a^{m'} \not\equiv 1 \pmod{n}$, tak máme istotu, že (5.4) pre m' neplatí.

3. Takže pre m vlastnosť (5.4) platí a pre $\frac{m}{2}$ nie. Teraz môže nastať jeden z nasledujúcich prípadov:

$$\begin{array}{lll} (p-1) \mid \frac{m}{2} & \& & (q-1) \nmid \frac{m}{2} \\ (p-1) \nmid \frac{m}{2} & \& & (q-1) \mid \frac{m}{2} \\ (p-1) \nmid \frac{m}{2} & \& & (q-1) \nmid \frac{m}{2} \end{array}$$

Prípad $(p-1) \mid \frac{m}{2}$ a $(q-1) \mid \frac{m}{2}$ nastať nemôže, lebo vtedy by pre všetky $a \in \mathbb{Z}$ nesúdeliteľné s n platilo: $a^{m/2} \equiv 1 \pmod{p}$ a $a^{m/2} \equiv 1 \pmod{q}$ (podľa malej Fermatovej vety). Odtiaľ $a^{m/2} \equiv 1 \pmod{n}$, čo je spor tým, že (5.4) neplatí pre $\frac{m}{2}$.
V prvých dvoch prípadoch je situácia symetrická. Bez ujmy na všeobecnosti môžeme teraz predpokladať, že $(p-1) \mid \frac{m}{2}$ a $(q-1) \nmid \frac{m}{2}$. Potom pre náhodne zvolené $b \in \mathbb{Z}$ (pričom $\mathrm{nsd}(b,n) = 1$) platia podľa malej Fermatovej vety a podľa lemy 2 nasledujúce kongruencie s uvedenými pravdepodobnosťami (Pr):

$$\begin{array}{lll} b^{m/2} \equiv 1 \pmod{p} & & \Pr = 1; \\ b^{m/2} \equiv 1 \pmod{q} & & \Pr = 1/2; \\ b^{m/2} \equiv -1 \pmod{q} & & \Pr = 1/2. \end{array}$$

V treťom prípade pre náhodne zvolené $b \in \mathbb{Z}$ (pričom $\mathrm{nsd}(b,n) = 1$) platia podľa lemy 2 nasledujúce kongruencie s uvedenými pravdepodobnosťami:

$$\begin{array}{lll} b^{m/2} \equiv 1 \pmod{p} & & \Pr = 1/2; \\ b^{m/2} \equiv -1 \pmod{p} & & \Pr = 1/2; \\ b^{m/2} \equiv 1 \pmod{q} & & \Pr = 1/2; \\ b^{m/2} \equiv -1 \pmod{q} & & \Pr = 1/2. \end{array}$$

Vo všetkých prípadoch náhodnou voľbou b čoskoro nájdeme také, že $b^{m/2} - 1$ je deliteľné jedným z faktorov n, ale nie druhým (pravdepodobnosť pri náhodej voľbe b je $\frac{1}{2}$). Výpočtom $\mathrm{nsd}(b^{m/2} - 1, n)$ a dostaneme jeden z faktorov n a po predelení aj druhý.

Poznámka. Zdieľanie spoločnej hodnoty n skupinou používateľov nie je bezpečný spôsob použitia RSA. Keďže znalosť jednej dvojice verejného a súkromného kľúča umožňuje efektívnu faktorizáciu n, súkromné kľúče používateľov v takejto skupine nemožno považovať za skutočne tajné.

Všeobecné algoritmy na faktorizáciu

Faktorizácia je významný a intenzívne skúmaný problém. Najjednoduchšou metódou faktorizácie daného $n \in \mathbb{N}$ je postupne skúšať deliť prvočíslami menšími alebo rovnými $\sqrt{n}$ (trial division). Existuje viacero efektívnejších metód.

Asymptoticky najrýchlejším všeobecným faktorizačným algoritmom je v súčasnosti General Number Field Sieve. Bol úspešne použitý na faktorizáciu 174 ciferného (576 bitov dlhého) čísla[2]. Jeho heuristická časová zložitosť je $O(e^{1{,}923 \cdot (\log n)^{1/3} (\log \log n)^{2/3}})$.

[2] výsledok z decembra 2003 (challenge RSA-576)

Špeciálne faktorizačné algoritmy

Pokiaľ majú prvočísla p, q špeciálnu štruktúru, je možné n faktorizovať ľahšie ako použitím všeobecných algoritmov. Z toho vyplýva, že generovanie prvočísel pre RSA musí byť dostatočne kvalitné. Uveďme aspoň dva špeciálne faktorizačné algoritmy spolu s odporučeniami pre voľbu p, q.

- Rozdiel $|p-q|$ by mal byť dostatočne veľký. V opačnom prípade sú p a q blízko. Hodnotu n možno vždy napísať v tvare $n = (\frac{p+q}{2})^2 - (\frac{p-q}{2})^2$. Ak sú p a q blízko, tak $\frac{p+q}{2}$ je len o málo väčšie ako $\lfloor\sqrt{n}\rfloor$. Algoritmus na faktorizáciu n je potom takýto: postupne skúšame $t = \lfloor\sqrt{n}\rfloor, \lfloor\sqrt{n}\rfloor + 1, \ldots$ až nájdeme t také, že $t^2 - n$ je štvorcom nejakého prirodzeného čísla, teda $t^2 - n = s^2$. Potom prvočísla p, q zistíme riešením sústavy lineárnych rovníc:

$$\begin{aligned} p + q &= 2t \\ p - q &= 2s \end{aligned} \qquad \Longrightarrow \qquad \begin{aligned} p &= t + s \\ q &= t - s. \end{aligned}$$

- Čísla $p-1$ aj $q-1$ by mali mať aj veľký prvočíselný faktor. Nech napríklad $p-1$ má len malé prvočíselné faktory. Algoritmus na faktorizáciu n je potom takýto:
  1. Zvolíme nejaký horný odhad najväčšieho prvočíselného deliteľa $p-1$, označme ho B. Vyberieme k tak, že k je deliteľné väčšinou, prípadne všetkými číslami z $\{1, 2 \ldots, B\}$. Napríklad môžeme zvoliť k ako najmenší spoločný násobok čísel 2, 3, ..., B.
  2. Zvolíme náhodne $a \in_R \{2, 3, \ldots, n-2\}$.
  3. Vypočítame $b = \mathrm{nsd}(a^k - 1, n)$.
  4. Ak je b triviálny deliteľ n (teda ak $b = 1$ alebo $b = n$), tak začneme s novou voľbou a alebo k.

  Ak je $p-1$ súčinom len prvočísel menších alebo rovných B, tak pri vhodnej voľbe k bude $(p-1) \mid k$. Napríklad ak $p - 1 = 2^{l_1}3^{l_2}7^{l_3}13^{l_4}$, tak môžeme ako k skúsiť $2^{20}3^{10}7^5 13^{15}$. Pre malé B je „istou“ voľbou k hodnota

$$k = \prod_{i=1}^{\pi(B)} p_i^{\lceil \log_{p_i} n \rceil},$$

  kde $p_1, \ldots, p_{\pi(B)}$ sú všetky prvočísla z $\{1, \ldots, B\}$. Ak $(p-1) \mid k$, potom podľa malej Fermatovej vety $a^k \equiv 1 \pmod{p}$, preto $p \mid a^k - 1$. Pokiaľ zároveň $a^k \not\equiv 1 \pmod{q}$, dostaneme netriviálny faktor n výpočtom $\mathrm{nsd}(a^k - 1, n) = p$.

Malý priestor správ

RSA je asymetrický šifrovací systém, teda ktokoľvek vie šifrovať (prirodzene, ak pozná verejný kľúč), ale len vlastník súkromného kľúča vie dešifrovať. Navyše, v prípade RSA prislúcha danému otvorenému textu práve jeden šifrový text. Táto vlastnosť umožňuje nasledujúci útok na systém.

Protivník zachytí šifrovanú správu. Postupne háda, aký by mohol byť otvorený text. Kandidátov (otvorené texty) postupne šifruje a porovnáva výsledok so zachytenou správou. Ak sú rovnaké vie, že uhádol správny otvorený text. Poznamenajme, že

protivník nepotrebuje poznať a ani sa nepokúša získať súkromný kľúč. Tento spôsob útoku predstavuje vážne nebezpečenstvo, pokiaľ je priestor potenciálnych správ príliš malý.

Rozšírenie priestoru správ a znáhodnenie šifrovacej transformácie sa dosahuje tzv. paddingom (vypchávkou), na úkor dĺžky otvoreného textu spracovateľného v jednej transformácii. Otvorený text sa pred šifrovaním doplní náhodným reťazcom bitov pevnej dĺžky (pozri obrázok 7.1 v kapitole 7). Po dešifrovaní sa tento reťazec zahodí. Dôsledkom je stav, keď jednému otvorenému textu zodpovedá potenciálne veľa šifrových textov a útočník nedokáže efektívne vyskúšať všetky možnosti.

Útok na malý verejný exponent e

Výhoda malého verejného exponentu e spočíva v rýchlom šifrovaní (resp. overovaní digitálnych podpisov) a menších pamäťových nárokoch na uloženie verejného kľúča. Riziko použitia malého verejného exponentu ilustrujeme na nasledujúcich príkladoch.

Posielanie rovnakej správy

Nech používatelia A, B a C používajú rovnaký verejný exponent $e_A = e_B = e_C = 3$. Nech ich verejné moduly sú n_A, n_B a n_C. Predpokladajme, že používateľ U chce poslať všetkým rovnakú správu, označme ju m. Pokiaľ získame všetky tri šifrované podoby m, teda $m^3 \bmod n_A$, $m^3 \bmod n_B$ a $m^3 \bmod n_C$, vieme získať pôvodný otvorený text m. Pri útoku rozlíšime dva prípady:

1. Hodnoty n_A, n_B a n_C nie sú po dvoch nesúdeliteľné. Potom vieme výpočtom nsd faktorizovať aspoň dve z nich[3]. Následne zistíme niektorý súkromný kľúč a dešifrujeme otvorený text m. Poznamenajme, že tento prípad je za normálnych okolností veľmi nepravdepodobný.
2. Hodnoty n_A, n_B a n_C sú po dvoch nesúdeliteľné. Použitím Čínskej zvyškovej vety riešime sústavu kongruencií:

$$\begin{aligned} x &\equiv m^3 \pmod{n_A}, \\ x &\equiv m^3 \pmod{n_B}, \\ x &\equiv m^3 \pmod{n_C}. \end{aligned}$$

Nájdeme riešenie

$$x = a_1 n_B n_C N_1 + a_2 n_A n_C N_2 + a_3 n_A n_B N_3, \tag{5.5}$$

kde

$$\begin{aligned} a_1 &= m^3 \bmod n_A, & N_1 &= (n_B n_C)^{-1} \bmod n_A \\ a_2 &= m^3 \bmod n_B, & N_2 &= (n_A n_C)^{-1} \bmod n_B \\ a_3 &= m^3 \bmod n_C, & N_3 &= (n_A n_B)^{-1} \bmod n_C \end{aligned}$$

Je ľahké overiť, že $x \equiv m^3 \pmod{n_A n_B n_C}$, lebo x aj m^3 sú riešeniami sústavy a podľa Čínskej zvyškovej vety sú všetky riešenia kongruentné modulo $n_A n_B n_C$.

[3]Prípadom, keď sú niektoré n_A, n_B, n_C rovnaké, sa nezaoberáme.

Keďže $m < n_A, n_B, n_C$, tak $m^3 < n_A n_B n_C$. Preto $x \bmod n_A n_B n_C = m^3$. Hodnotu x vypočítame zo vzťahu (5.5), teda vieme zistiť tretiu mocninu otvoreného textu. Výpočtom tretej odmocniny (napr. binárnym vyhľadávaním) následne určíme m.

Závislé správy

Ďalší útok využíva afinnú závislosť správ šifrovaných malým verejným exponentom. Útočník zachytí šifrové texty dvoch správ m_1 a m_2. Označme zodpovedajúce šifrové texty c_1, c_2. Predpokladajme, že útočník pozná afinnú závislosť medzi m_1 a m_2, teda koeficienty a, b, pre ktoré $m_2 = a \cdot m_1 + b$. Nech z je premenná, ktorá označuje neznámu správu m_1. Platí:

$$\begin{aligned} z^e - c_1 &\equiv 0 \pmod{n} \\ (az+b)^e - c_2 &\equiv 0 \pmod{n} \end{aligned}$$

Hodnota m_1 je koreňom oboch polynómov na ľavej strane ekvivalencií, preto sú polynómy deliteľné výrazom $z - m_1$. Útočník vypočíta najväčší spoločný deliteľ oboch polynómov (nad $\mathbb{Z}_n$): $\mathrm{nsd}(z^e - c_1, (az+b)^e - c_2)$. Zvyčajne je výsledkom lineárny polynóm $z - m_1$, z ktorého útočník získa správu m_1 a následne aj m_2, keďže vzťah medzi m_1 a m_2 je známy.

Príklad. Nech $n = 91$ a $e = 5$. Nech $c_1 = 45$, $c_2 = 28$ a medzi správami platí afinná závislosť $m_2 = 30 \cdot m_1 + 11$. Útočník počíta najväčší spoločný deliteľ polynómov:

$$\begin{aligned} \mathrm{nsd}(z^5 - 45, \quad & (30z+11)^5 - 28) = \\ & = \mathrm{nsd}(z^5 + 46, \quad 88z^5 + 40z^4 + 90z^3 + 33z^2 + 47z + 44) \\ & = z + 37 = z - 54 \end{aligned}$$

Teda $m_1 = 54$ a $m_2 = 30 \cdot 54 + 11 = 84$. Korektnosť správ možno skontrolovať ich šifrovaním, po ktorom dostaneme šifrové texty c_1 a c_2.

Útok je ľahko zovšeobecniteľný na ľubovoľnú, útočníkovi známu polynomiálnu závislosť medzi správami m_1 a m_2. Nech $m_2 = p(m_1)$, kde $p(x)$ je polynóm. Útočník bude tentokrát počítať $\mathrm{nsd}(z^e - c_1, p(z)^e - c_2)$. Postup je prakticky realizovateľný vtedy, ak je verejný exponent e malý a polynóm $p(x)$ má malý stupeň. Vtedy sú polynómy pre výpočet najväčšieho spoločného deliteľa krátke a vieme s nimi efektívne počítať. Okrem voľby veľkého verejného exponentu je prevenciou aj náhodný padding (vypchávka) otvoreného textu, popísaný pri probléme s malým priestorom správ, ktorý znemožňuje útočníkovi poznať závislosť otvorených textov.

Sú známe aj ďalšie útoky na malé verejné exponenty [Cop96, CFPR96, Has88]. Vo všeobecnosti sa preto neodporúča voliť v RSA systéme príliš malé hodnoty e (napr. 3, 5, 7 a podobne).

Útok na krátky súkromný exponent d

Výhodami krátkeho súkromného exponentu d je rýchlejšie dešifrovanie (resp. rýchlejšia realizácia podpisovania pri digitálnych podpisoch) a menšie pamäťové nároky

na uschovanie súkromného kľúča. Je známy útok na d, ktorý vypočíta hodnotu d z hodnôt e a n, ak $d < n^{0,292}$ a $e < n^{1,875}$ [BD99]. Poznamenajme, že druhý vzťah je obvykle splnený, keďže vo zvyčajných implementáciach platí $e < \phi(n)$.

Takmer CCA útok

RSA má vlastnosti, ktoré umožňujú takmer CCA útok (útok s možnosťou voľby šifrového textu). Tento útok nevedie k prezradeniu súkromného kľúča, umožňuje však „nepozorovane“ dešifrovať ľubovoľný šifrový text. Majme šifrový text c a zaujíma nás k nemu prislúchajúci otvorený text, teda m také, že $m^e \bmod n = c$. Predpokladajme, že máme možnosť vykonať dešifrovanie ľubovoľného šifrového textu. Prirodzene, mohli by sme sa pokúsiť podstrčiť na dešifrovanie priamo c. Nasledujúci postup nám umožní zamaskovať c tak, aby sa nikto (a najmä nie vlastník súkromného kľúča) nedozvedel, o aký text v skutočnosti máme záujem.

Zvolíme náhodné $x \in_R \mathbb{Z}_n^*$. Teda $x \in \mathbb{Z}_n$ a x je nesúdeliteľné s n. Ak by sme natrafili na x súdeliteľné s n, tak vieme faktorizovať n, zistiť súkromný kľúč a dešifrovať c samostatne. Namiesto c použijeme šifrový text $c' = x^e c \bmod n$. Následne, po obdržaní $m' = D(c') = (c')^d \bmod n$, vypočítame $m = m'x^{-1} \bmod n = (x^e c)^d x^{-1} \bmod n = c^d \bmod n$.

Prirodzene, celý postup funguje len v tom prípade, ak prostredie, v ktorom je RSA použité, umožňuje dešifrovanie na požiadanie. V niektorých situáciách to tak môže byť, napríklad pri použití RSA na autentifikáciu alebo pri používaní slepých podpisov. V jednotlivých prípadoch je potrebné analyzovať, či nie je možné podniknúť popísaný útok a prípadne modifikovať dotknuté (ohrozené) protokoly.

Bezpečnosť posledného bitu otvoreného textu

Zaujímavou otázkou pri ľubovoľnom šifrovacom systéme je, akú informáciu nám dáva šifrový text o otvorenom texte. Presnejšie, čo sme schopní zo šifrového textu efektívne zistiť o otvorenom texte. Napríklad hodnotu najvyššieho alebo najnižšieho bitu. Práve o najnižšiemu (paritnému) bitu otvoreného textu v RSA systéme sa venuje táto časť.

Predstavme si, že máme k dispozícii algoritmus (orákulum), ktorý pre ľubovoľné zadané $E(x)$ zistí paritu x. Predpokladáme, že tento algorimus môže používať hodnoty e a n. Ukážeme platnosť nasledujúceho tvrdenia.

Veta 3. *Ak máme algoritmus určujúci paritu otvoreného textu x z ľubovoľného šifrového textu $E(x)$ v polynomiálnom čase, tak vieme v pravdepodobnostnom polynomiálnom čase dešifrovať ľubovoľný šifrový text, teda počítať $D(c)$ pre všetky $c \in \mathbb{Z}_n$.*

To znamená, že zistiť posledný bit otvoreného textu je v RSA rovnako ťažké, ako „rozbiť“ RSA. Zaveďme označenie zvyškovej triedy modulo n symbolom $[y]_n = y \bmod n$, pre $y \in \mathbb{Z}$. Postup počítania $x = D(c)$ je takýto (vstup je c):

1. Zvolíme náhodne $a, b \in_R \mathbb{Z}_n^*$.
2. Vypočítame $[lx]_n = \text{nsd}([ax]_n, [bx]_n)$. Môžu nastať dva prípady:
   (**a**) $\text{nsd}([ax]_n, [bx]_n) = 1$, teda $[ax]_n$ a $[bx]_n$ sú nesúdeliteľné. My dostaneme výsledok $[lx]_n$ ako hodnotu l (pozri nižšie). Vieme však otestovať, kedy $[lx]_n = 1$. To je práve vtedy, keď $E(lx) = 1$, lebo $E(1) = 1$. Platí $E(lx) = E(l)E(x) \bmod n = (l^e E(x)) \bmod n$ a šifrový text $E(x)$ poznáme. Následne už len vypočítame

hľadané x: $x = l^{-1} \bmod n$ (lebo $[lx]_n = 1$). Výpočet inverzného prvku možno realizovať napríklad pomocou rozšíreného Euklidovho algoritmu.
(**b**) $\mathrm{nsd}([ax]_n, [bx]_n) \neq 1$. To nastalo, keď $E(l)E(x) \bmod n \neq 1$. Postup je potrebné zopakovať s novou voľbou a a b.

Prirodzene, uvedený postup vyvoláva mnohé otázky, na ktoré treba odpovedať. Prejdeme všetkými „bielymi“ miestami nášho algoritmu a objasníme ich.

Ako môžeme počítať s $[ax]_n$ a $[bx]_n$, keď ich nepoznáme?
Ľubovoľné hodnoty v našom algoritme budú mať tvar $[tx]_n$, pričom budeme poznať hodnotu t a teda šifrovú podobu $E(tx) = E(t)E(x) \bmod n$. Samozrejme, s takýmito hodnotami nemôžeme robiť ľubovoľné aritmetické operácie, ale ak poznáme a, b a $E(x)$, vieme vypočítať šifrovanú podobu ľubovoľnej lineárnej kombinácie $[ax]_n$ a $[bx]_n$:

$$\begin{aligned} E(A[ax]_n + B[bx]_n) &= E((Aa + Bb)x \bmod n) \\ &= (Aa + Bb)^e E(x) \bmod n \\ &= E(Aa + Bb)E(x) \bmod n \end{aligned}$$

kde $A, B \in \mathbb{Z}$. Obidva činitele v poslednom výraze sú známe (prvý vieme vypočítať a druhý poznáme), takže vieme vypočítať aj $E(A[ax]_n + B[bx]_n)$.

Ako počítať nsd**?**
Algoritmus, ktorý použijeme na výpočet nsd, používa dve operácie testovania (porovnanie a test parity) a operáciu lineárnej kombinácie argumentov:

1. Ak sú oba argumenty párne, tak ich naraz delíme dvoma dovtedy, kým aspoň jeden z nich nie je nepárny. Príslušná mocnina dvojky bude súčasťou počítaného najväčšieho spoločného deliteľa.
2. Ak je práve jeden z argumentov párny, delíme ho dvoma, pokiaľ sa dá.
3. Oba agrumenty sú nepárne. Porovnáme ich. Ak sú rovnaké, ich najväčším spoločným deliteľom je táto hodnota. Inak od väčšieho argumentu odčítame menší, čím dostaneme opäť jeden párny argument a opakujeme postup.

Uvedený algoritmus pre výpočet nsd pracuje v deterministickom polynomiálnom čase, rovnako, ako klasický Euklidov algoritmus. Poznamenajme, že delenie dvoma je násobenie konštantou ($2^{-1} \bmod n$), teda stále lineárna kombinácia argumentov. Realizácia oboch operácií testovania spočíva v použití algoritmu (orákula) na určovanie parity otvoreného textu a vyplýva z lemy 3 (pozri ďalej). Test rovnosti je triviálnym zložením dvoch operácií porovnania, alebo využitím bijektívnosti RSA.

Koľkokrát treba postup opakovať?
Samozrejme, kým nezvolíme a, b tak, že $[ax]_n$ a $[bx]_n$ sú nesúdeliteľné (požiadavka z druhého kroku nášho postupu) a zároveň $[ax]_n, [bx]_n < \frac{n}{2}$ (požiadavka z lemy 3, aby korektne „fungovali“ operácie testovania). Pravdepodobnosť, že dve náhodne zvolené čísla z $\{1, \ldots, n\}$ sú nesúdeliteľné je $\frac{2\Phi(n)}{n^2}$, kde $\Phi(n) = \sum_{k=1}^{n} \phi(k)$. Platí nasledujúci asymptotický odhad [GKP94, strana 462]:

$$\Phi(n) = \frac{3}{\pi^2} \cdot n^2 + O(n \log n).$$

Teda hľadaná pravdepodobnosť je $6/\pi^2 \doteq 0,6079$, pre $n \to \infty$. Pravdepodobnosť, že dve náhodne zvolené čísla z $\{1,\ldots,n\}$ sú menšie ako $\frac{n}{2}$, je $\frac{1}{4}$. Teda priemerný počet opakovaní nášho postupu je približne $4\pi^2/6 \doteq 6.58$ krát.

Lema 3. *Nech $\mathcal{O}$ je také orákulum (odpovedajúce 0 alebo 1), že $\mathcal{O}E(x) = 0 \Leftrightarrow [x]_n$ je párne. Nech a, b sú zvolené tak, že $[ax]_n, [bx]_n < \frac{n}{2}$. Potom platí:*

$$[ax]_n \geq [bx]_n \iff \mathcal{O}E(2x(a-b)) = 0, \tag{5.6}$$

$$[bx]_n \text{ je párne} \iff \mathcal{O}E(bx) = 0. \tag{5.7}$$

Dôkaz. Tvrdenie (5.7) je triviálne pravdivé. Ukážeme platnosť tvrdenia (5.6).

($\Rightarrow$) Máme $ax = l_1 n + k_1$ a $bx = l_2 n + k_2$, pre nejaké $l_1, l_2 \in \mathbb{N}$, pričom $0 \leq k_2 \leq k_1 < \frac{n}{2}$. Potom $2x(a-b) \bmod n = 2(k_1 - k_2)$, lebo $0 \leq 2(k_1 - k_2) < n$. Keďže $2(k_1 - k_2)$ je párne, $\mathcal{O}E(2x(a-b)) = 0$. Z implementačného hľadiska pripomeňme, že vieme počítať šifrovanú podobu ľubovoľnej lineárnej kombinácie $[ax]_n$ a $[bx]_n$, teda aj hodnotu $E(2x(a-b))$.

($\Leftarrow$) Nech $[ax]_n < [bx]_n$. Potom $0 \leq k_1 < k_2 < \frac{n}{2}$, kde k_1 a k_2 sú ako v prvej časti dôkazu. Odtiaľ dostávame $-n \leq 2(k_1 - k_2) < 0$. Počítajme:

$$\begin{aligned} 2x(a-b) \bmod n &= 2(k_1 - k_2) \bmod n \\ &= n - 2(k_1 - k_2). \end{aligned}$$

Posledný výraz je rozdiel nepárneho a párneho čísla, teda nepárne číslo, a preto $\mathcal{O}E(2x(a-b)) = 1$. □

Postup, ktorý sme uviedli v tejto časti, je možné ďalej vylepšovať. Môžeme uvažovať o orákulách, ktoré sa niekedy mýlia – teda určia hodnotu bitu len s pravdepodobnosťou $\frac{1}{2} + \varepsilon$, pre $\varepsilon > 0$. Prípadne sa môžeme pýtať na bezpečnosť k-teho najmenej významného bitu (nie iba paritného). Pre takúto situáciu platí napríklad nasledujúce tvrdenie [FS97]:

> *Ak máme orákulum, ktoré v čase T dokáže z e a n pre ľubovoľné $E(x)$ určiť k-ty posledný bit x s pravdepodobnosťou $\frac{1}{2}+\varepsilon$ ($\varepsilon > 0$), tak dokážeme v pravdepodobnostnom čase $O((\log n)^2 \varepsilon^{-2}(T + 2^{2k}\varepsilon^{-4}))$ počítať $D(c)$ pre všetky $c \in \mathbb{Z}_n$.*

6 Kvadratické rezíduá

Problematika kvadratických rezíduí (zvyškov) má viaceré aplikácie v kryptografii, preto sa jej v tejto časti budeme v nevyhnutnom rozsahu venovať.

Definícia 6. Číslo $a \in \mathbb{Z}_n^*$ (teda číslo z množiny $\{1,\ldots,n-1\}$, ktoré je nesúdeliteľné s n) sa nazýva kvadratickým rezíduom modulo n (označenie QR_n), ak existuje $b \in \mathbb{Z}_n$ také, že $b^2 \equiv a \pmod{n}$. Ak také b neexistuje, nazývame a kvadratickým nerezíduom modulo n (označenie QNR_n).

Príklad. Nech $n = 11$. Z tabuľky vidieť, že čísla 1, 3, 4, 5 a 9 sú QR_{11}.

x	1	2	3	4	5	6	7	8	9	10
$x^2 \bmod 11$	1	4	9	5	3	3	5	9	4	1

Ak počítame v $\mathbb{Z}_p$, kde p je prvočíslo, je charakterizácia kvadratických rezíduí jednoduchá.

Lema 4. *Nech $p > 2$ je prvočíslo. Potom platí:*

1. *Presne polovica z čísel $\{1, \ldots, p-1\}$ sú QR_p.*
2. *Ak x je riešením rovnice $x^2 \equiv a \pmod p$, pre $a \in \{1, \ldots, p-1\}$, tak aj $-x$ (teda $p - x$) je jej riešením.*

Dôkaz. 1. Nech $x^2 \equiv y^2 \pmod p$, pre nejaké $x, y \in \mathbb{Z}_p^*$. Potom

$$p \mid (x^2 - y^2) \quad \Rightarrow \quad p \mid (x-y)(x+y).$$

Môžu nastať dva prípady:

$$\begin{aligned} p \mid (x-y) \quad &\Rightarrow \quad x = y \\ p \mid (x+y) \quad &\Rightarrow \quad x = p - y. \end{aligned}$$

V prvom prípade sú x a y rovnaké. V druhom sú rôzne (lebo p je nepárne). Teda na kvadratické rezíduum modulo p sa vždy zobrazia (umocnením na druhú) dva prvky zo $\mathbb{Z}_p^*$. Preto je práve polovica čísel z tejto množiny QR_p.

2. Počítajme: $(p-x)^2 \bmod p = (p^2 - 2px + x^2) \bmod p = x^2 \bmod p = a$, čo bolo treba dokázať.

□

Prirodzenou otázkou pri skúmaní kvadratických rezíduí je hľadanie ich druhých odmocnín, teda riešenie rovnice $x^2 \equiv a \pmod n$, kde a je QR_n. V nasledujúcej leme sa zameriame na prípad, keď n je prvočíslo v špeciálnom tvare.

Lema 5. *Nech $p \equiv 3 \pmod 4$ je prvočíslo a nech a je QR_p. Potom riešením rovnice $x^2 \equiv a \pmod p$ je $x = a^{(p+1)/4} \bmod p$.*

Dôkaz. Keďže $p \equiv 3 \pmod 4$, tak $(p+1)/4$ je celé číslo. Počítajme:

$$x^2 \equiv (a^{(p+1)/4})^2 \equiv a^{(p+1)/2} \equiv a \cdot a^{(p-1)/2} \pmod p. \tag{6.1}$$

Vieme, že a je QR_p, preto existuje b také, že $b^2 \equiv a \pmod p$. Odtiaľ dostávame:

$$\begin{aligned} (b^2)^{(p-1)/2} &\equiv a^{(p-1)/2} \pmod p \\ b^{p-1} &\equiv a^{(p-1)/2} \pmod p \\ 1 &\equiv a^{(p-1)/2} \pmod p. \end{aligned}$$

Posledná úprava kongruencie vyplýva z malej Fermatovej vety. Po dosadení do (6.1) dostávame $x^2 \equiv a \pmod p$. □

Poznámka. Riešenie rovnice $x^2 \equiv a \pmod{p}$ pre prvočíslo $p \equiv 1 \pmod 4$ a pre a (QR_p) je zložitejšie, hoci zvládnuteľné v pravdepodobnostnom polynomiálnom čase [Kob87].

Lemu 5 možno použiť aj na testovanie, či nejaké $a \in \mathbb{Z}_p^*$ je QR_p, pre prvočíslo p v tvare $p \equiv 3 \pmod 4$. Postupujeme ako keby a bolo QR_p a nájdeme $x = a^{(p+1)/4} \bmod p$. Následne vykonáme skúšku správnosti, teda overíme, či $x^2 \equiv a \pmod p$. Skúška dopadne úspešne práve vtedy, keď a je naozaj QR_p.

Jednoduchšie testovanie, platné pre ľubovoľné prvočíslo p, poskytuje Eulerovo kritérium.

Lema 6 (Eulerovo kritérium). *Nech $p > 2$ je prvočíslo. Potom*

$$a \text{ je } QR_p \iff a^{\frac{p-1}{2}} \equiv 1 \pmod p.$$

Dôkaz. ($\Rightarrow$) Nech a je QR_p. Potom existuje $b \in Z_p$ také, že $b^2 \equiv a \pmod p$. Počítajme (druhá ekvivalencia vyplýva z malej Fermatovej vety):

$$a^{\frac{p-1}{2}} \equiv b^{p-1} \equiv 1 \pmod p.$$

($\Leftarrow$) Multiplikatívna grupa $(\mathbb{Z}_p^*, \cdot)$ je cyklická. Nech g je jej generátor. Teda pre nejaké $r \in \mathbb{N}$: $g^r = a$. Predpokladáme, že $a^{(p-1)/2} \equiv 1 \pmod p$. Dosadením dostávame (implikácia platí, lebo g je generátor):

$$g^{\frac{r}{2}\cdot(p-1)} \equiv 1 \pmod p \qquad \Longrightarrow \qquad (p-1) \mid \frac{r}{2} \cdot (p-1)$$

Takže $r/2$ je celé číslo. Potom $b = g^{r/2}$ je hľadaný prvok taký, že $b^2 \equiv g^r \equiv a \pmod p$. To znamená, že a je QR_p. □

Nasledujúca veta sa zaoberá kryptograficky najzaujímavejším prípadom – kvadratickými rezíduami modulo n, kde n je súčin dvoch prvočísel. Presnejšie, venuje sa zložitosti hľadania druhých odmocnín QR_n.

Veta 4. *Nech $n = p \cdot q$, kde $p \neq q$ sú nepárne prvočísla. Potom sú nasledujúce tvrdenia ekvivalentné:*

1. *Existuje pravdepodobnostný polynomiálny algoritmus pre faktorizáciu n.*
2. *Existuje pravdepodobnostný polynomiálny algoritmus pre výpočet druhých odmocnín kvadratických rezíduí modulo n.*

Dôkaz. ($1 \Rightarrow 2$) Majme k dispozícii pravdepodobnostný polynomiálny algoritmus pre faktorizáciu n. Použijeme ho a zo zadaného n vypočítame p a q. Chceme riešiť rovnicu $x^2 \equiv a \pmod n$, kde a je QR_n. Ak by a bolo QNR_n, tak jednoducho vypočítame nekorektnú druhú odmocninu (lebo neexistuje), čo vieme následne overiť skúškou správnosti. Najskôr nájdeme druhé odmocniny a modulo p a modulo q. Ak sú $p, q \equiv 3 \pmod 4$, tak môžeme použiť lemu 5, v opačnom prípade treba použiť náročnejší postup (pozri poznámku za lemou 5). Dostaneme hodnoty u, v:

$$u^2 \equiv a \pmod p$$
$$v^2 \equiv a \pmod q$$

Použitím Čínskej zvyškovej vety (prvočísla p a q sú navzájom nesúdeliteľné) riešime sústavu

$$x \equiv u \pmod{p}$$
$$x \equiv v \pmod{q}$$

Riešenie x dostaneme ako lineárnu kombináciu u a v:

$$x = u \cdot q \cdot (q^{-1} \bmod p) + v \cdot p \cdot (p^{-1} \bmod q).$$

Ľahko môžeme overiť, že

$$x^2 \equiv a \pmod{p}$$
$$x^2 \equiv a \pmod{q}$$

Prvočísla p a q sú nesúdeliteľné, preto $x^2 \equiv a \pmod{n}$.

(2 $\Rightarrow$ 1) Majme k dispozícii algoritmus, ktorý rieši rovnice $x^2 \equiv a \pmod{n}$. Ukážeme, ako faktorizovať n.

Zvolíme náhodne $m \in_R \mathbb{Z}_n^*$ a položíme $a = m^2 \bmod n$. Necháme si nájsť riešenie rovnice $x^2 \equiv a \pmod{n}$ algoritmom, ktorý máme k dispozícii. Ten vypočíta riešenie k (jedno zo štyroch možných). Môže nastať jedna z nasledujúcich, rovnako pravdepodobných možností:

A	$k \equiv$	$m \pmod{p}$	$\wedge$	$k \equiv$	$m \pmod{q}$
B	$k \equiv$	$m \pmod{p}$	$\wedge$	$k \equiv -$	$m \pmod{q}$
C	$k \equiv -$	$m \pmod{p}$	$\wedge$	$k \equiv$	$m \pmod{q}$
D	$k \equiv -$	$m \pmod{p}$	$\wedge$	$k \equiv -$	$m \pmod{q}$

Tieto štyri alternatívy vyplývajú z toho, že $m^2 \equiv k^2 \pmod{n}$, teda $pq \mid (m+k)(m-k)$. Ak teraz vypočítame $\mathrm{nsd}(k-m, n) = p$, dostaneme v jednotlivých prípadoch:

A	$\mathrm{nsd}(k-m, n) = n$
B	$\mathrm{nsd}(k-m, n) = p$
C	$\mathrm{nsd}(k-m, n) = q$
D	$\mathrm{nsd}(k-m, n) = 1$

Teda s pravdepodobnosťou 1/2 dostaneme pri výpočte $\mathrm{nsd}(k-m, n)$ netriviálny faktor n. Pravdepodobnosť, že n nefaktorizujeme po l opakovaniach nášho postupu je 2^{-l}. □

Poznámka. Formulácia vety hovorí o pravdepodobnostných polynomiálnych algoritmoch preto, lebo práve tie sú zvyčajne považované za model efektívnych výpočtov (pre naše účely je použitá formulácia dostatočná). Redukcie v dôkaze takýto predpoklad zvlášť nevyužívajú a vo vete by bolo možné uviesť časovú zložitosť závislú na konkrétnej redukcii, počte volaní a zložitosti predpokladaného algoritmu. Pripomeňme, že pravdepodobnostný charakter má len redukcia použitá v druhej časti dôkazu.

7 Rabinov systém

Rabinov systém patrí medzi tie asymetrické šifrovacie systémy, o ktorých bezpečnosti sa dá dokázať, že je ekvivalentná s problémom faktorizácie. Medzi ďalšie takéto systémy patria napríklad schémy [Wil80] alebo [LKBS92]. Všetky však zdieľajú spoločnú slabinu: keďže dôkaz ekvivalencie s problémom faktorizácie je konštruktívny, sú neodolné voči útoku s možnosťou voľby šifrového textu (CCA).

Inicializácia. Používateľ zvolí dve veľké rôzne prvočísla p a q, pričom $p, q \equiv 3 \pmod 4$. Verejný kľúč je číslo $n = p \cdot q$. Súkromný kľúč tvorí faktorizácia n, teda prvočísla p, q.

Množina otvorených textov je $\mathbb{Z}_n$. Šifrový text v Rabinovom systéme je druhou mocninou otvoreného textu: $E(m) = m^2 \bmod n$, pre $m \in \mathbb{Z}_n$. Dešifrovanie je zložitejšie. Šifrovacia funkcia E na množine $\mathbb{Z}_n$ totiž nie je injektívna.

Ak $\mathrm{nsd}(m, n) = 1$, tak $E(m)$ je kvadratické rezíduum modulo n (QR_n, pozri časť 6). Keďže n je súčinom dvoch prvočísel, má každé QR_n, označme ho c, práve štyri druhé odmocniny. Ak $\mathrm{nsd}(m, n) \neq 1$, čo je veľmi nepravdepodobný prípad, tak buď $m = 0$ (vtedy $E(m) = 0$ a vieme jednoznačne dešifrovať) alebo $\mathrm{nsd}(m, n)$ je netriviálny faktor n (vtedy môžeme považovať náš súkromný kľúč za prezradený). V takomto prípade je druhá odmocnina jedna ($m = 0$) alebo sú dve ($m \neq 0$). V ďalšej diskusii predpokladajme situáciu $\mathrm{nsd}(m, n) = 1$.

Štyri odmocniny môžeme vypočítať tak, že najskôr určíme druhé odmocniny c ($c = E(m)$) modulo p a modulo q (ak je číslo QR_n, tak je triviálne aj QR_p a QR_q). V oboch prípadoch sú to dve odmocniny, pozri lemu 5:

$$r_{1,2} = \pm c^{\frac{p+1}{4}} \bmod p,$$
$$r_{3,4} = \pm c^{\frac{q+1}{4}} \bmod q.$$

Druhé odmocniny c modulo n dostaneme ich lineárnou kombináciou podľa Čínskej zvyškovej vety (pre korektnosť tohto postupu pozri prvú časť dôkazu vety 4):

$$M_1 = (ar_1 + br_3) \bmod n,$$
$$M_2 = (ar_1 + br_4) \bmod n,$$
$$M_3 = (ar_2 + br_3) \bmod n,$$
$$M_4 = (ar_2 + br_4) \bmod n,$$

kde $a = q \cdot (q^{-1} \bmod p)$ a $b = p \cdot (p^{-1} \bmod q)$.

Po dešifrovaní dostaneme štyri potenciálne otvorené texty. Pokiaľ je očakávaný otvorený text kontextovo závislý (má nejakú štruktúru), je jednoduché odlíšiť ho od ostatných. V opačnom prípade je potrebné k textu pred šifrovaním pridávať vhodnú hlavičku alebo padding (vypchávku, pozri ďalej).

Bezpečnosť

Ľahko vidieť, že bezpečnosť Rabinovho systému je založená na probléme počítania druhých odmocnín modulo n, kde n je súčin dvoch veľkých prvočísel. Tento problém je ekvivalentný problému faktorizácie n, podľa vety 4.

Nepríjemnou, už vyššie avizovanou vlastnosťou Rabinovho systému je neodolnosť voči útoku s možnosťou voľby šifrového textu (CCA). Predstavme si, že protivník si môže zvoliť šifrový text c, ktorý mu majiteľ súkromného kľúča dešifruje. Teda protivník dostane na požiadanie jednu zo štyroch druhých odmocnín modulo n. CCA prebieha nasledovne: protivník zvolí náhodné $m \in_R \mathbb{Z}_n^*$ a nechá si dešifrovať $c = m^2 \bmod n$. Po zistení k, $k^2 \equiv c \pmod n$, môžu nastať štyri prípady – rovnaké ako pri dôkaze vety 4. Útočník následným výpočtom $\mathrm{nsd}(k - m, n)$ faktorizuje n s pravdepodobnosťou $1/2$. Pri l pokusoch je pravdepodobnosť úspešného útoku protivníka (teda faktorizácie n) $1 - 2^{-l}$. Rabinov systém je preto možné použiť len v takom prostredí (v takých protokoloch), kde nie je možné realizovať CCA.

Závislé správy

Rabinov systém je rovnako ako RSA náchylný na útok pri šifrovaní závislých správ (pozri časť 5.4 venovanú malému verejnému exponentu), keďže pri šifrovaní je exponent iba 2. To znamená, že ak útočník pozná šifrované podoby správ m_1 a m_2 spolu s polynomiálnou závislosťou $m_2 = p(m_1)$, dokáže otvorené texty určiť. Nech c_1, c_2 sú šifrové texty správ m_1, m_2. Potom pre polynóm $z - m_1$ platí:

$$(z - m_1) \mid (z^2 - c_1),$$
$$(z - m_1) \mid (p(z)^2 - c_2).$$

Útočník vypočíta $\mathrm{nsd}(z^2 - c_1, p(z)^2 - c_2)$ a následne určí oba otvorené texty m_1 a m_2.

Tento postup nie je možné použiť na útok vedúci k faktorizácii n. Samozrejme, môžeme zvoliť ľubovoľnú správu m_1 a polynomiálnu závislosť $m_2 = p(m_1)$. Po výpočte nsd však nedostaneme náhodne jedno zo štyroch riešení $x^2 \equiv m_1^2 \pmod n$, čo by nám umožnilo faktorizovať n s pravdepodobnosťou $1/2$ podobne ako pri CCA. Problém je v tom, že pre ostatné otvorené texty k c_1 a c_2 platí predpísaná polynomiálna závislosť len v zanedbateľnom počte prípadov.

Špeciálny prípad nastane vtedy, keď je hodnota m_2 v polynomiálnej závislosti rovnaká pre m_1 aj $-m_1$ (napríklad $m_2 = 2m_1^4 + m_1^2 + 3$). Vtedy argumenty v nsd delí aj polynóm $p + m_1$ a po výpočte dostaneme najväčí spoločný deliteľ v tvare $z^2 - c_1$, z ktorého m_1 nevieme určiť.

Príklad. 1. Nech $n = 77$, $c_1 = 53$, $c_2 = 37$ a $m_2 = 2m_1^4 + m_1^2 + 3$. Potom

$$\begin{aligned}\mathrm{nsd}(z^2 - 53, \quad & (2z^4 + z^2 + 3)^2 - 37) = \\ & = \mathrm{nsd}(z^2 + 24, \quad 4z^8 + 4z^6 + 13z^4 + 6z^2 + 49) \\ & = z^2 + 24 = z^2 - 53,\end{aligned}$$

teda o otvorených textoch sme sa nič nedozvedeli.

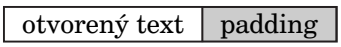
2. Nech $n = 77$, $c_1 = 23$, $c_2 = 58$ a $m_2 = 2m_1^3 + m_1^2 + 3$. Potom

$$\begin{aligned}\mathrm{nsd}(z^2 - 23, \quad & (2z^3 + z^2 + 3)^2 - 58) = \\ & = \mathrm{nsd}(z^2 + 54, \quad 4z^6 + 4z^5 + 12z^3 + z^4 + 6z^2 + 28) \\ & = z + 32 = z - 45,\end{aligned}$$

teda $m_1 = 45$ (alebo $m_1 = -45 = 32$) a $m_2 = 2 \cdot 45^3 + 45^2 + 3 = 17$ (alebo $m_2 = -17 = 60$). Správnosť dešifrovania možno overiť opätovným šifrovaním m_1 a m_2.

Padding (vypchávka)

Padding je doplnenie otvoreného textu pred šifrovaním (šifrovacou transformáciou), najčastejšie reťazcom bitov pevnej dĺžky. Doplnenie môže byť realizované náhodným reťazcom, konštantou, prípadne kontrolným súčtom. Padding znižuje mohutnosť množiny otvorených textov. Na druhej strane, pomáha riešiť niektoré bezpečnostné alebo praktické problémy spojené s použitím konkrétneho asymetrického systému. Napríklad náhodný padding pri RSA pomáha, ak je počet potenciálnych správ malý.



Obrázok 7.1: Padding (vypchávka)

V Rabinovom systéme má zmysel uvažovať o konštantnom paddingu alebo o paddingu s kontrolným súčtom z nasledujúcich dôvodov:

1. Umožní sa jednoznačné dešifrovanie, vďaka schopnosti odlíšiť správny otvorený text. Po dešifrovaní totiž očakávame (a skontrolujeme) definovaný padding.
2. Zamedzí sa realizácii CCA útoku, keď po dešifrovaní skontrolujeme padding. Ak žiadny zo štvorice textov nevyhovuje, bude výsledok dešifrovania prázdny. Poznamenajme, že v takto upravenom Rabinovom systéme už dôkaz ekvivalencie rozbitia systému a problému faktorizácie nie je platný.
3. Vhodne zvolený padding môže znížiť pravdepodobnosť polynomiálne závislých správ.

8 Diskrétny logaritmus

Doteraz sme sa v diskusii o asymetrických šifrovacích systémoch venovali takým, ktorých bezpečnosť bola viac či menej závislá na probléme faktorizácie. Ďalším problémom, na ktorého zložitosti sú postavené mnohé kryptografické konštrukcie, je problém diskrétneho logaritmu.

Definícia 7. Nech $(G, \cdot)$ je konečná grupa a nech $b, y \in G$. Potom diskrétny logaritmus y pri základe b je ľubovoľné $x \in \mathbb{N}$ také, že $b^x = y$. Problémom diskrétneho logaritmu nazývame úlohu nájdenia diskrétneho logaritmu pre zadané b a y.

Prirodzene, diskrétny logaritmus nemusí existovať pre všetky dvojice b a y. Poznamenajme, že ak vezmeme ľubovoľnú konečnú grupu $(G, \cdot)$ a prvok $b \in G$, tak množina $\{1, b, b^2, b^3, \dots\}$ vytvorí cyklickú podgrupu grupy $(G, \cdot)$. V kryptografických aplikáciach sa najčastejšie počíta v cyklických grupách pri základe, ktorý je generátorom grupy. Vtedy je existencia diskrétneho logaritmu zaručená a problém diskrétneho logaritmu možno formulovať konkrétnejšie:

Nech $(G, \cdot)$ je konečná cyklická grupa rádu $n \in \mathbb{N}$ a nech $g \in G$ je jej generátor. Pre dané $y \in G$ je potrebné vypočítať $x \in \mathbb{Z}_n$ také, že $g^x = y$.

Voľba generátora nemá na zložitosť výpočtu diskrétneho logaritmu vplyv. Nech g a h sú generátory grupy $(G, \cdot)$ rádu n. Predpokladajme, že vieme počítať diskrétny logaritmus pri základe h. Potom diskrétny logaritmus, napríklad prvku $c \in G$, pri základe g vypočítame takto:

1. Vypočítame $a \in \mathbb{Z}_n$: $h^a = g$. Keďže g je generátor, musí byť a nesúdeliteľné s n. Preto existuje inverzný prvok k a modulo n. Označme tento prvok a^{-1}.
2. Vypočítame $b \in \mathbb{Z}_n$: $h^b = c$.
3. Hľadaný diskrétny logaritmus je ba^{-1}:

$$g^{ba^{-1}} = (h^a)^{ba^{-1}} = h^b = c.$$

Nech $b \in \mathbb{Z}_n^*$ (pre nejaké $n \in \mathbb{N}$) a $G = \{1, b, b^2, \dots\}$, kde operácia „$\cdot$“ označuje násobenie modulo n. V takom prípade budeme hovoriť o diskrétnom logaritme modulo n. Obvykle sa volí grupa $(\mathbb{Z}_p^*, \cdot)$, kde p je prvočíslo.

Príklad. 1. Nech $p = 11$ a $g = 7$. Poznamenajme, že 7 je generátor $(\mathbb{Z}_{11}^*, \cdot)$.

i	1	2	3	4	5	6	7	8	9	10
g^i	7	5	2	3	10	4	6	9	8	1

Teda napríklad diskrétny logaritmus 4 pri základe 7 je 6.

2. Nech $p = 11$ a nech $g = 5$.

i	1	2	3	4	5	6	7	8	9	10
g^i	5	3	4	9	1	5	3	4	9	1

Potom diskrétny logaritmus 9 pri základe 5 je 4 (alebo aj 9), ale diskrétny logaritmus 2 pri základe 5 v $(\mathbb{Z}_{11}^*, \cdot)$ neexistuje.

V súčasnosti nie je známy efektívny všeobecný algoritmus na riešenie problému diskrétneho logaritmu. Zložitosť známych algoritmov a ich aplikovateľnosť je ovplyvnená grupou, v ktorej je problém potrebné riešiť. V prípade prvočíselných grúp $(\mathbb{Z}_p^*, \cdot)$ je momentálne najlepším algoritmom Number Field Sieve s časovou zložitosťou $O(e^{1{,}923 \cdot (\log p)^{1/3} (\log \log p)^{2/3}})$. V iných grupách, napríklad nad eliptickými krivkami, sa postupy počítania diskrétnych logarimov nad modulárnou aritmetikou nedajú použiť, a preto je v nich počítanie diskrétnych logaritmov náročnejšie. Treba podotknúť, že vzťah zložitosti problémov faktorizácie a diskrétneho logaritmu nie je známy.

Pre prvočíselné grupy $(\mathbb{Z}_p^*, \cdot)$ existujú efektívne algoritmy na výpočet diskrétneho logaritmu, ak má p špeciálnu štruktúru. Napríklad Pohligov-Hellmanov algoritmus (pozri časť 8.1) umožňuje využiť prípad, keď má $p - 1$ len malé prvočíselné faktory. Preto sa snažíme zvoliť prvočíslo p tak, aby $p - 1$ malo aj veľký prvočíselný faktor. Obľúbenou voľbou je hľadať p v tvare $p = 2q + 1$, kde q je tiež prvočíslo. Tento tvar má navyše tú výhodu, že pri znalosti q vieme ľahko testovať, či je nejaký prvok $g \in \mathbb{Z}_p^*$ generátorom $(\mathbb{Z}_p^*, \cdot)$. Platí:

$$g \text{ je generátor} \iff g^2 \neq 1 \ \wedge \ g^q \neq 1.$$

Táto vlastnosť vyplýva z Lagrangeovej vety, že rád konečnej grupy je deliteľný rádom ľubovoľnej jej podgrupy. Vo všeobecnosti efektívne testovanie, či je prvok zo $\mathbb{Z}_p^*$ generátorom, vyžaduje znalosť faktorizácie $p-1$. Tvar $p = 2q+1$ má ešte jednu výhodu. Keďže počet generátorov grupy $(\mathbb{Z}_p^*, \cdot)$ je presne $\phi(p-1)$, v našom prípade je to $\phi(p-1) = \phi(2q) = q-1$. Teda pravdepodobnosť, že náhodne zvolený prvok bude generátorom je $(q-1)/(2q)$, t.j. pre veľké q približne 50%.

Poznámka. Generovanie prvočísla p v tvare $2q+1$ realizujeme tak, že najskôr volíme náhodne prvočíslo q a následne testujeme prvočíselnosť čísla $2q+1$. Postup opakujeme, kým nenájdeme vhodné q (a teda aj p).

8.1 Pohligov-Hellmanov algoritmus

Nech $(G, \cdot)$ je konečná cyklická grupa rádu $n \in \mathbb{N}$ a nech $g \in G$ je jej generátor. Pohligov-Hellmanov algoritmus (z roku 1978) počíta diskrétny logaritmus v tejto grupe pri základe g. Algoritmus je efektívny vtedy, keď n má iba malé prvočíselné faktory.

Vstupom algoritmu je prvok $c \in G$, výstupom hodnota $x \in \mathbb{Z}_n$, pričom $g^x = c$. Nech $p_1, \ldots, p_k$ sú všetky navzájom rôzne prvočíselné delitele n. Potom hodnotu n môžeme vyjadriť takto:

$$n = p_1^{n_1} \cdot p_2^{n_2} \cdot \ldots \cdot p_k^{n_k},$$

kde $n_1, \ldots, n_k > 0$. Postup pri výpočte diskrétneho logaritmu má dva kroky:

1. Najskôr vypočítame také $x^{(1)}, \ldots, x^{(k)}$, že platí

$$\begin{aligned} x^{(1)} &\equiv x \bmod p_1^{n_1} \\ x^{(2)} &\equiv x \bmod p_2^{n_2} \\ &\vdots \\ x^{(k)} &\equiv x \bmod p_k^{n_k} \end{aligned}$$

2. Pomocou Čínskej zvyškovej vety poskladáme x z hodnôt $x^{(1)}, \ldots, x^{(k)}$. Prvočísla $p_1, \ldots, p_k$ sú navzájom rôzne a teda nesúdeliteľné, preto je možné použiť Čínsku zvyškovú vetu.

Zostáva ukázať, ako výpočítať hodnoty $x^{(i)}$, pre ľubovoľné $i = 1, \ldots, k$. Vyjadrime $x^{(i)}$ v sústave so základom p_i:

$$x^{(i)} = d_0 + d_1 p_i + d_2 p_i^2 + \cdots + d_{n_i-1} p_i^{n_i-1},$$

kde $0 \leq d_j < p_i$, pre všetky $j = 0, \ldots, n_i - 1$. Vypočítajme:

$$c^{n/p_i} = g^{x \cdot n/p_i}.$$

Rád prvku g je n, lebo g je generátor grupy $(G, \cdot)$. To znamená, že hodnotu výrazu v exponente stačí počítať modulo n. Pri úprave výrazu využijeme fakt, že hľadáme

$x^{(i)} \equiv x \pmod{p_i^{n_i}}$ a teda $x = x^{(i)} + l_i p_i^{n_i}$, pre nejaké celé číslo l_i:

$$\begin{aligned} x \cdot \frac{n}{p_i} &\equiv (x^{(i)} + l_i p_i^{n_i}) \cdot \frac{n}{p_i} \equiv x^{(i)} \cdot \frac{n}{p_i} + n \cdot l_i p_i^{n_i - 1} \\ &\equiv \left(d_0 + d_1 p_i + \ldots + d_{n_i - 1} p_i^{n_i - 1}\right) \cdot \frac{n}{p_i} \\ &\equiv d_0 \cdot \frac{n}{p_i} \pmod{n}. \end{aligned}$$

Dostávame:

$$c^{n/p_i} = g^{d_0 \cdot n/p_i} = \beta_i^{d_0},$$

kde sme označili $\beta_i = g^{n/p_i}$. Hodnotu d_0 možno potom určiť prehľadaním postupnosti $\beta_i^0, \beta_i^1, \ldots, \beta_i^{p_i - 1}$. Keďže g je generátor, sú všetky prvky v postupnosti navzájom rôzne.

Predpokladajme teraz, že poznáme hodnoty $d_0, \ldots, d_{r-1}$, pre $r \geq 1$. Ďalšiu cifru $x^{(i)}$ v sústave so základom p_i, teda d_r, určíme podobným postupom ako d_0. Položíme

$$c_r = c \cdot g^{-(d_0 + d_1 p_i + \ldots + d_{r-1} p_i^{r-1})}.$$

Následne vypočítame $c_r^{n/p_i^{r+1}}$. Pre zjednodušenie zápisu v odvodení opäť uvádzame len exponent generátora g (modulo n):

$$\begin{aligned} \left(x - \left(d_0 + \ldots + d_{r-1} p_i^{r-1}\right)\right) \cdot \frac{n}{p_i^{r+1}} &\equiv \\ &\equiv \left(x^{(i)} + l_i p_i^{n_i} - \left(d_0 + \ldots + d_{r-1} p_i^{r-1}\right)\right) \cdot \frac{n}{p_i^{r+1}} \\ &\equiv \left(d_r + d_{r+1} p_i + \ldots + d_{n_i - 1} p_i^{n_i - 1 - r}\right) \cdot \frac{n}{p_i} \\ &\equiv d_r \cdot \frac{n}{p_i} \pmod{n}. \end{aligned}$$

Dostávame:

$$c_r^{n/p_i^{r+1}} = g^{d_r \cdot n/p_i} = \beta_i^{d_r}$$

a prehľadaním postupnosti $\beta_i^0, \beta_i^1, \ldots, \beta_i^{p_i - 1}$ určíme d_r.

Popísaným spôsobom získame všetky hodnoty $x^{(1)}, \ldots, x^{(k)}$ a následným výpočtom podľa Čínskej zvyškovej vety aj hľadaný diskrétny logaritmus, teda x.

Z postupu vidieť, že potrebujeme tabuľky hodnôt

$$\begin{aligned} &\langle 1, \beta_1, \beta_1^2, \ldots, \beta_1^{p_1 - 1} \rangle \\ &\langle 1, \beta_2, \beta_2^2, \ldots, \beta_2^{p_2 - 1} \rangle \\ &\quad \ldots \\ &\langle 1, \beta_k, \beta_k^2, \ldots, \beta_k^{p_k - 1} \rangle, \end{aligned}$$

kde $\beta_i = g^{n/p_i}$, pre $i = 1, \ldots, k$. To znamená, že ak má rád grupy veľký prvočíselný faktor, bude vytvorenie tabuľky (alebo jej priebežné počítanie) pre tento faktor zdĺhavé. V takomto prípade je Pohligov-Hellmanov algoritmus neefektívny.

Pohligov-Hellmanov algoritmus v $(\mathbb{Z}_p^*, \cdot)$
Rád grupy $(\mathbb{Z}_p^*, \cdot)$ je $n = p - 1$. Pre prvočíslo $p > 2$ sa dá rád grupy vyjadriť v tvare $n = 2^s \cdot t$, kde $s \geq 1$ a t je nepárne. Nech g je generátor grupy a nech $c \in \mathbb{Z}_p^*$. Opäť chceme zistiť diskrétny logaritmus c pri základe g, teda x také, že $g^x = c$.

Vyjadrime hľadané x v dvojkovej sústave:

$$x = d_0 + 2d_1 + \ldots + 2^{s-1}d_{s-1} + \ldots + 2^l d_l,$$

kde $l = \lfloor \lg(p-1) \rfloor + 1$. Počítajme:

$$c^{(p-1)/2} = g^{x(p-1)/2} = g^{d_0(p-1)/2} = \begin{cases} 1 & \text{ak } d_0 = 0 \\ -1 & \text{ak } d_0 = 1 \end{cases}$$

Hodnotu bitu d_1 určíme podobne:

$$\begin{aligned} \left(c \cdot g^{-d_0}\right)^{(p-1)/2^2} &= g^{(x-d_0)(p-1)/2^2} \\ &= g^{d_1(p-1)/2} = \begin{cases} 1 & \text{ak } d_1 = 0 \\ -1 & \text{ak } d_1 = 1 \end{cases} \end{aligned}$$

Takto postupujeme až po bit d_{s-1}:

$$\begin{aligned} \left(c \cdot g^{-d_0 - 2d_1 - \ldots - 2^{s-2}d_{s-2}}\right)^{(p-1)/2^s} &= \\ &= g^{(2^{s-1}d_{s-1} + \ldots + 2^l d_l)(p-1)/2^s} \\ &= g^{(d_{s-1} + \ldots + 2^{l-s+1}d_l)(p-1)/2} \\ &= g^{d_{s-1}(p-1)/2} \cdot g^{(p-1)(d_s + \ldots + 2^{l-s}d_l)} \\ &= g^{d_{s-1}(p-1)/2} = \begin{cases} 1 & \text{ak } d_{s-1} = 0 \\ -1 & \text{ak } d_{s-1} = 1 \end{cases} \end{aligned}$$

Ďalej sa už nedá postupovať, pretože $2^{s+1} \nmid (p-1)$. Určili sme posledných s bitov diskrétneho logaritmu. Keďže $s \geq 1$, vieme vždy vypočítať aspoň paritný bit x. Poznamenajme, že voľba prvočísla p v tvare $2q+1$, kde q je tiež prvočíslo, obmedzuje tento výpočet práve na paritný bit (v takomto prípade je $s = 1$).

9 ElGamalov systém

Autorom tohto systému z roku 1984 je Taher ElGamal. Uveďme najskôr jeho popis.

Inicializácia. Zvolíme veľké prvočíslo p a prvok $g \in \mathbb{Z}_p^*$ (môže ale nemusí to byť generátor). Hodnoty p a g môžu byť spoločné pre viacerých používateľov. Ďalej zvolíme náhodne $x \in_R \{2, 3, \ldots, p-2\}$ a vypočítame $y = g^x \pmod p$. Verejný kľúč je potom trojica (y, p, g) a súkromný kľúč hodnota x.

Spoločné hodnoty p a g znižujú pamäťové nároky na uloženie verejných kľúčov používateľov. Na druhej strane prinášajú isté bezpečnostné riziká, ak tieto hodnoty útočník predpripraví špeciálnym spôsobom [Gor92].

Šifrovanie. Priestor otvorených textov je množina $\mathbb{Z}_p$, pri dlhších textoch môžeme tieto rozdeliť na bloky požadovanej dĺžky. Nech $m \in \mathbb{Z}_p$ je otvorený text (správa), ktorý chceme šifrovať.

1. Zvolíme náhodné $k \in_R \{1, 2, \ldots, p-1\}$.
2. Šifrový text je dvojica $\langle r, s \rangle$, kde $r = g^k \bmod p$ a $s = y^k \cdot m \bmod p$. Pripomeňme, že hodnota y je súčasťou verejného kľúča.

Dešifrovanie. Používateľ, poznajúc súkromný kľúč x, môže dešifrovať správu takto:

$$m = (r^x)^{-1} \cdot s \bmod p.$$

Lema 7 (Koreknosť ElGamalovho systému). *Nech p, g, y, x sú parametre inštancie ElGamalovho systému. Nech $m \in \mathbb{Z}_p$ je správa a $\langle r, s \rangle$ jej šifrová podoba. Potom $m = (r^x)^{-1} \cdot s \bmod p$.*

Dôkaz. Počítajme:

$$\begin{aligned}(r^x)^{-1} \cdot s \bmod p &= (g^{kx})^{-1} \cdot y^k \cdot m \bmod p \\ &= g^{-kx} \cdot g^{kx} \cdot m \bmod p = m \bmod p = m.\end{aligned}$$

Keďže $g \neq 0$, príslušné inverzné prvky existujú. □

Všimnime si, že šifrový text je dvakrát dlhší ako otvorený text. Na druhej strane, jednému otvorenému textu zodpovedá potenciálne $O(p)$ šifrových textov. ElGamalov systém je príkladom znáhodneného šifrovania.

Príklad. Nech $p = 11$ a $g = 7$. Nech používateľ zvolí súkromný kľúč $x = 4$. Potom $y = 7^4 \bmod 11 = 3$. Predpokladajme, že správa je $m = 9$.

- Šifrovanie: zvolíme napríklad $k = 2$. Potom šifrový text je dvojica

  $$\langle 7^2 \bmod 11, 3^2 \cdot 9 \bmod 11 \rangle = \langle 5, 4 \rangle.$$

- Dešifrovanie: vypočítame $(5^4)^{-1} \cdot 4 \bmod 11 = 5 \cdot 4 \bmod 11 = 9$.

Pristavme sa teraz pri realizovateľnosti ElGamalovho systému. Realizačné problémy voľby prvočísla, modulárneho umocňovania ako aj počítania inverzného prvku sme už diskutovali pri RSA systéme. V tomto prípade, vďaka prvočíselnosti p, môžeme inverzný prvok počítať (namiesto rozšíreného Euklidovho algoritmu) aj použitím malej Fermatovej vety: $w^{p-2} \equiv w^{-1} \pmod p$, $\forall w \in \mathbb{Z}_p^*$. Takýto výpočet je však časovo náročnejší. O voľbe g, pokiaľ má byť generátorom, sme čiastočne hovorili pri probléme diskrétneho logaritmu v kapitole 8.

Zvýšenie efektívnosti šifrovania možno dosiahnuť aj predvýpočtom hodnôt r a $y^k \bmod p$, ktoré nezávisia na správe. Potom šifrovanie pozostáva len z jedného modulárneho násobenia a je veľmi rýchle. Nevýhodou je potreba držať predvypočítanú hodnotu $y^k \bmod p$ v tajnosti – v prípade prezradenia by protivník vedel dešifrovať správu m jednoduchým predelením (resp. prenásobením hodnoty s výrazom $(y^k)^{-1} \bmod p$).

9.1 Bezpečnosť

Bezpečnosť ElGamalovho systému je založená na zložitosti problému diskrétneho logaritmu. Ak by bol protivník schopný vypočítať x, tak by vedel dešifrovať ľubovoľnú správu. Ak by bol schopný vypočítať k zo vzťahu $r = g^k \bmod p$, tak vie výpočtom $s \cdot (y^k)^{-1} \bmod p$ zistiť správu m. Takže, ak vieme efektívne počítať diskrétny logaritmus, tak vieme „rozbiť“ ElGamalov systém. Na druhej strane, zatiaľ sa nepodarilo dokázať, že problém rozbitia tohto systému je ekvivalentný problému diskrétneho logaritmu. Môže existovať spôsob, ako zo známych hodnôt $r = g^k \bmod p$ a $y = g^x \bmod p$ vypočítať $g^{kx} \bmod p$ (a následne dešifrovať) bez toho, aby sme zistili hodnotu k alebo x, teda bez výpočtu diskrétneho logaritmu. Zatiaľ taký spôsob nepoznáme. V špeciálnych prípadoch ekvivalenciu ukázal Maurer [Mau94].

Takže v prípade ElGamalovho systému a problému diskrétneho logaritmu sme v podobnej situácii ako v prípade systému RSA a problému faktorizácie. V kryptológii je známy Diffieho-Hellmanov (skrátene DH) problém, ktorého pôvod vychádza z Diffieho-Hellmanovho protokolu (bližšie o ňom budeme hovoriť neskôr). Dôležité je, že bezpečnosť ElGamalovho systému je ekvivalentná práve tomuto problému. Uveďme najskôr zjednodušenú formuláciu DH problému:

> Počítajme v grupe $(\mathbb{Z}_p^*, \cdot)$, kde p je prvočíslo. Nech $g \in \mathbb{Z}_p^*$. Pre dané hodnoty g^a, g^b je potrebné vypočítať g^{ab}.

Lema 8. *Nech p, g sú parametre pre inštanciu ElGamalovho systému. Potom sú nasledujúce tvrdenia ekvivalentné:*

1. *Existuje pravdepodobnostný polynomiálny algoritmus, ktorý dešifruje ľubovoľný šifrový text bez znalosti súkromného kľúča.*
2. *Existuje pravdepodobnostný polynomiálny algoritmus pre riešenie ľubovoľného DH problému pre grupu $(\mathbb{Z}_p^*, \cdot)$ a prvok g.*

Dôkaz. (1 $\Rightarrow$ 2) Nech vieme dešifrovať v ElGamalovom systéme. Potrebujeme riešiť DH problém, teda pre zadané g^a a g^b je potrebné vypočítať g^{ab}. Skonštruujeme šifrový text $r = g^a$, $s = 1$ a položme $y = (g^b)^{-1}$. To znamená, že súkromný kľúč x zodpovedajúci tomuto y má hodnotu $-b$. Použijeme algoritmus na dešifrovanie šifrového textu $\langle r, s\rangle$ so znalosťou verejného kľúča (y, p, g). Dostaneme otvorený text:

$$m = (r^x)^{-1} \cdot s \bmod p = ((g^a)^{-b})^{-1} \cdot 1 \bmod p = g^{ab} \bmod p,$$

ktorý je zároveň riešením DH problému.

(2 $\Rightarrow$ 1) Máme k dispozícii šifrový text $\langle r, s\rangle$ a verejný kľúč (y, p, g). Použijeme algoritmus pre DH problém so vstupom r, y. Keďže $r = g^k$ a $y = g^x$, algoritmus vypočíta hodnotu $g^{kx} = y^k$, z ktorej ľahko určíme otvorený text:

$$s \cdot (y^k)^{-1} \bmod p = y^k \cdot m \cdot y^{-k} \bmod p = m.$$

□

Poznámka. Pojem „pravdepodobnostný polynomiálny algoritmus“ v predchádzajúcej leme používame len preto, lebo predstavuje triedu výpočtov považovaných za

efektívne. Redukcie používané pri dôkaze nemajú pravdepodobnostnú povahu, preto by bolo možné nahradiť tento pojem výrazom „deterministický polynomiálny algoritmus“, prípadne uviesť časovú zložitosť presne, v závislosti na časovej zložitosti predpokladaných algoritmov.

Pri implementácii ElGamalovho systému je potrebné venovať pozornosť tomu, aby sa hodnota k, ani hodnota y^k nedostala k protivníkovi. Najlepšie je na ne hneď po použití zabudnúť. Vzhľadom na zviazanosť systému s problémom diskrétneho logaritmu je vhodné, keď sú parametre p a g volené tak, ako o tom hovoríme v časti 8, t.j. $p = 2q + 1$, kde q je prvočíslo a g je generátor grupy $(\mathbb{Z}_p^*, \cdot)$.

Poznámka. ElGamalov systém je možné sformulovať v ľubovoľnej konečnej grupe. Z hľadiska bezpečnosti je dôležité, aby v takejto grupe bol problém diskrétneho logaritmu (presnejšie Diffieho-Hellmanov problém) ťažký.

10 Hašovacie funkcie***

Kryptografické hašovacie funkcie (ďalej len hašovacie funkcie) zobrazujú elektronický dokument alebo správu na „odtlačok“, obvykle podstatne kratší ako pôvodný dokument. Hašovacia funkcia h je vo všeobecnosti zobrazenie $h : X \to Y$, kde Y je konečná množina a X môže ale nemusí byť konečná množina. Hodnotu $x \in X$ nazývame dokumentom, správou alebo vzorom, hodnotu $h(x)$ najčastejšie odtlačkom. Odtlačok $h(x)$ slúži ako reprezentant pôvodného dokumentu x a môže byť namiesto x použitý.

Hašovacie funkcie sa používajú v schémach digitálnych podpisov (kapitola 11), pri kontrole integrity, v kryptografických protokoloch a ďalších konštrukciách. Bezpečné použitie hašovacej funkcie v spomenutých aplikáciach vyžaduje, aby funkcia mala určité vlastnosti. Definujme teda základné vlastnosti hašovacích funkcií.

Definícia 8 (jednosmernosť). Hašovacia funkcia $h : X \to Y$ je jednosmerná (one-way), ak pre dané $y \in Y$ nie je možné efektívne nájsť $x \in X$ také, že $h(x) = y$.

Jednosmernosť sa zvykne označovať aj ako odolnosť vzoru (preimage resistance) a znamená, že zo samotného odtlačku dokumentu nevieme (efektívne) rekonštruovať pôvodný dokument. Pokiaľ pracujeme s pevne zvolenou hašovacou funkciou, pojem „efektívne nájsť“ nie je možné sformalizovať. Z hľadiska teórie zložitosti vieme pre pevne zvolenú funkciu s konečným oborom hodnôt hľadať vzory k odtlačkom v konštantnom čase $O(1)$.

Podobná situácia je aj pri symetrických šifrovacích algoritmoch, ktoré sú tiež pevne zvolené. Bezpečnosť konkrétnej konštrukcie je posudzovaná ako počet krokov najefektívnejšieho útoku v porovnaní s dostupnou výpočtovou kapacitou.

Definícia 9 (kolízie – slabá odolnosť). Hašovacia funkcia $h : X \to Y$ je slabo odolná voči kolíziam, ak pre dané $x \in X$ nie je možné efektívne nájsť $x' \in X \setminus \{x\}$ také, že $h(x) = h(x')$.

Iné označenie pre slabú odolnosť voči kolíziám je odolnosť druhého vzoru (second preimage resistance). Znamená, že k danému dokumentu nevieme nájsť iný dokument s rovnakým odtlačkom. Silnejšia forma odolnosti voči kolíziam je taká, keď nevieme nájsť dva rôzne dokumenty s rovnakým odtlačkom (collision resistance).

Definícia 10 (kolízie – silná odolnosť). Hašovacia funkcia $h : X \to Y$ je silne odolná voči kolíziam (alebo len „odolná voči kolíziam“), ak nie je možné efektívne nájsť dvojicu $x, x' \in X$ takú, že $x \neq x'$ a $h(x) = h(x')$.

Príklad. Nech $h : X \to X$ je funkcia identity, teda $\forall x \in X : h(x) = x$. Potom h nie je jednosmerná – vzor k odtlačku vieme nájsť ľahko. Na druhej strane je funkcia h slabo aj silne odolná voči kolíziam, keďže h nemá žiadne kolízie.

Príklad. Nech $h : \mathbb{Z}_n \to \mathbb{Z}_n$ je definovaná takto: $h(x) = x^2 \bmod n$, kde $n = p \cdot q$ je súčin dvoch prvočísel, pričom faktorizáciu nepoznáme. Potom h je jednosmerná, lebo výpočet vzoru je výpočtovo ekvivalentný faktorizácii n (pozri vetu 4 v kapitole 6). Vo všeobecnosti môžeme faktorizáciu považovať za úlohu, ktorú nevieme efektívne riešiť. Funkcia h nie je odolná voči kolíziam (slabo ani silne). Kolízie vieme nájsť ľahko, sú nimi ľubovoľné dvojice x a $-x$, keďže $h(x) = h(-x)$.

Ľahko vidieť, že $|Y|$ by nemala byť príliš malá. V opačnom prípade môžeme nájsť kolízie tak, že postupne berieme prvky v X a pre každú hodnotu $y \in Y$ si pamätáme príslušný vzor. Najneskôr po $|Y| + 1$ pokusoch natrafíme na kolíziu (samozrejme, ak $|X| > |Y|$). Ak hašovacia funkcia distribuuje vzory do množiny Y rovnomerne, teda pravdepodobnosť $\Pr_{x \in X}[h(x) = y]$ je pre rôzne y zhruba rovnaká, tak možno rovnakým postupom „útočiť“ aj na jednosmernosť – hľadať k danému $y \in Y$ prvok $x \in h^{-1}(y)$.

Na druhej strane, príliš veľká množina Y má tiež nevýhody. Odtlačky sú dlhé a pre veľmi krátke dokumenty môže byť odtlačok dlhší ako pôvodný dokument. Keďže odtlačky nahrádzajú samotné dokumenty v schémach digitálnych podpisov, dlhé odtlačky môžu komplikovať (z hľadiska časovej zložitosti) transformácie pre vytvorenie, resp. overenie podpisu.

Pokiaľ hašovacia funkcia nie je jednosmerná, môžeme hľadať kolízie funkcie $h : X \to Y$ takýmto postupom:

1. zvolíme náhodné $x \in_R X$ a vypočítame $y = h(x)$
2. vypočítame x' také, že $h(x') = y$ (predpokladáme, že h nie je jednosmerná)
3. ak $x \neq x'$, tak máme kolíziu

Analyzujme pravdepodobnosť úspechu algoritmu.

Lema 9. *Nech X a Y sú konečné množiny. Potom pravdepodobnosť, že vyššie uvedený postup nájde kolíziu vo funkcii $h : X \to Y$ je aspoň $1 - |Y|/|X|$.*

Dôkaz. Označme hľadanú pravdepodobnosť Pr. Nech $[x]$ označuje triedu ekvivalencie, obsahujúcu všetky tie $x' \in X$, ktoré majú rovnaký odtlačok ako x, teda $[x] = \{x' \in X \mid h(x) = h(x')\}$. Nech C označuje množinu všetkých tried ekvivalencie pre funkciu h. Ľahko vidieť, že $|C| \leq |Y|$. Počítajme Pr:

$$\begin{aligned}\Pr &= \frac{1}{|X|} \sum_{x \in X} \frac{|[x]| - 1}{|[x]|} = \frac{1}{|X|} \sum_{c \in C} \sum_{x \in c} \frac{|c| - 1}{|c|} \\ &= \frac{1}{|X|} \sum_{c \in C} (|c| - 1) = \frac{1}{|X|} \sum_{c \in C} |c| - \frac{|C|}{|X|} \geq 1 - \frac{|Y|}{|X|}.\end{aligned}$$

□

Prirodzene, ak algoritmus skončí neúspechom (teda $x = x'$), môžeme ho opakovať. Pravdepodobnosť, že nenájdeme kolíziu po k opakovaniach, je najviac $(|Y|/|X|)^k$.

Poznámka. Uvedený postup hľadania kolízií predpokladá „totálnu“ neplatnosť jednosmernosti, teda schopnosť nájsť vzor pre každé $y \in h(X)$. Pokiaľ je hašovacia funkcia invertovateľná len na časti množiny Y, môže sa stať, že práve na tejto časti je hašovacia funkcia injektívna a algoritmus žiadne kolízie nenájde. Na druhej strane, v prakticky používaných hašovacích funkciách môžeme odôvodnene predpokladať, vďaka viac-menej rovnomernej distribúcii obrazov, že z odolnosti voči kolíziám naozaj vyplýva vlastnosť jednosmernosti.

Kontrola integrity pomocou hašovacích funkcií

Dôležitosť definovaných vlastností hašovacích funkcií ilustrujeme na probléme kontroly integrity. Kontrola integrity znamená overenie, či neboli dáta (súbor, správa, dokument) zmenené. Zachovanie integrity je dôležité pri konfiguračných súboroch operačného systému, pri distribúcii softvéru a iných situáciach.

Na kontrolu integrity súborov v operačnom systéme možno použiť nasledujúci postup. Vypočítame odtlačky súborov, ktorých integritu chceme kontrolovať. Odtlačky odložíme tak, aby nemohli byť zmenené (napríklad na externé médium). Kontrola sa neskôr uskutoční opätovným výpočtom odtlačkov a porovnaním výsledkov s archivovanými odtlačkami. Prirodzene, rovnaký efekt dosiahneme archivovaním a porovnávaním celých súborov. Výhoda použitia hašovacích funkcií spočíva v tom, že odtlačky sú v porovnaní s pôvodnými súbormi krátke a jednoduchšie sa s nimi manipuluje. Aby bol takýto spôsob kontroly bezpečný, použitá hašovacia musí byť:

- jednosmerná – v opačnom prípade útočník, ak sa dostane k nejakému odtlačku, dokáže vypočítať obsah takého súboru, ktorým môže nahradiť pôvodný súbor, pričom kontrola túto zámenu neodhalí (predpokladáme, že vypočítaný súbor je iný ako pôvodný).
- slabo odolná voči kolíziam – vtedy útočník nevie vypočítať k pôvodnému súboru iný súbor s rovnakým odtlačkom, ktorým by mohol pôvodný súbor nahradiť.
- odolná voči kolíziam – v opačnom prípade môže útočník skonštruovať dva rôzne „kolízne“ súbory. Ak sa mu podarí presvedčiť používateľa o užitočnosti niektorého z nich, môže neskôr vymeniť súbor za druhý, pričom kontrola integrity ddopadne úspešne.

Podobne sa hašovacie funkcie používajú pri distribúci softvérových balíkov. Z jedného miesta sa distribuuje odtlačok balíka (napr. z hlavnej web stránky) a z iného miesta (obvykle z jedného z mnohých zrkadlových servrov, aby nebol hlavný web preťažený) samostný obsah balíka. Používateľ po stiahnutí balíka porovná odtlačok s odtlačkom z vypočítaným zo získaného balíka. Overenie slúži na kontrolu správnosti prenosu a za predpokladu, že útočník nevie vnútiť falošný „hlavný“ odtlačok, aj kontrolu autentickosti balíka.

10.1 Narodeninový útok

Narodeninový útok (birthday attack) je všeobecný útok na hašovacie funkcie, ktorého cieľom je nájsť kolízie. Postup útoku na hašovaciu funkciu $h : X \to Y$ je takýto:

1. Zvolíme rôzne vzory $x_1, x_2, \ldots, x_k \in X$.
2. Vypočítame ich odtlačky $h(x_1), \ldots, h(x_k)$.
3. Odtlačky utriedime a zistíme, či sú medzi nimi kolízie.

Posúdenie efektívnosti útoku vyžaduje analýzu zložitosti a pravdepodobnosti, že uvedeným postupom nájdeme kolíziu. Predpokladajme, že výpočet odtlačku hašovacej funkcie je realizovaný s časovou zložitosťou $O(1)$. Vtedy je zložitosť prvých dvoch krokov útoku $O(k)$. Množina Y je konečná (v prakticky používaných hašovacích funkciách najviac niekoľko sto bitov), preto môžeme triediť s lineárnou časovou zložitosťou (napríklad algoritmom radixsort). Hľadanie kolízií v utriedenej postupnosti odlačkov spočíva v jednoduchom prejdení postupnosti. Celková časová zložitosť útoku je teda $O(k)$.

Dôležitým parametrom útoku je počet vzorov, s ktorými budeme pracovať. Čím je hodnota k väčšia, tým pravdepodobnejšie je nájdenie kolízie, avšak zvyšuje sa časová (ale aj pamäťová) zložitosť. Presnejšia analýza vyžaduje predpoklad o „distribúcii“ odtlačkov hašovacej funkcie. Predpokladajme z hľadiska úspešnosti útoku najnepriaznivejšiu situáciu, keď sú odtlačky distribuované rovnomerne:

$$\forall\, y \in Y : \Pr[h_1(x) = y] = \frac{1}{|Y|}$$

Potom hodnoty $h(x_1), \ldots, h(x_k)$ sú náhodne, rovnako pravdepodobne distribuované prvky v Y a môžeme použiť nasledujúcu vetu.

Veta 5. *Nech $y_1, \ldots, y_k \in_R Y$ sú náhodne, rovnako pravdepodobne a nezávisle vybrané prvky, pričom $|Y| = n$ a $k < n$. Označme* $\Pr_{\text{col}}$ *pravdepodobnosť, že aspoň dva sú rovnaké. Potom platí:*

$$\Pr_{\text{col}} \geq 1 - e^{\frac{-k(k-1)}{2n}}.$$

Dôkaz. Pravdepodobnosť, že náhodne zvolené prvky $y_1, \ldots, y_k$ sú všetky rôzne je

$$\frac{n(n-1)(n-2)\cdot\ldots\cdot(n-k+1)}{n^k} = \Big(1-\frac{1}{n}\Big)\Big(1-\frac{2}{n}\Big)\cdot\ldots\cdot\Big(1-\frac{k-1}{n}\Big).$$

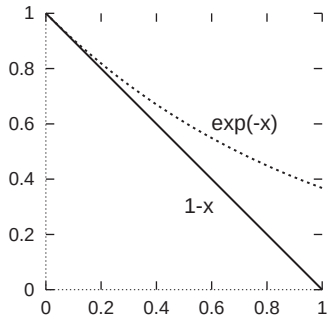
Potom

$$\Pr_{\text{col}} = 1 - \prod_{i=1}^{k-1}\Big(1-\frac{i}{n}\Big).$$

Vďaka Taylorovmu rozvoju funkcie $e^{-x} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \ldots$ dostávame $1 - x \leq e^{-x}$ pre ľubovoľné $x \in \langle 0, 1\rangle$. Túto nerovnosť možno vidieť aj z priebehu funkcií $1 - x$ a e^{-x}, pozri obrázok 10.1.
Dosadením do výrazu pre pravdepodobnosť $\Pr_{\text{col}}$ dostaneme tvrdenie vety:

$$\Pr_{\text{col}} \geq 1 - e^{-\frac{1}{n}} e^{-\frac{2}{n}} \cdot\ldots\cdot e^{-\frac{k-1}{n}} = 1 - e^{-\frac{k(k-1)}{2n}}.$$

□



Obrázok 10.1: Funkcie $1 - x$ a e^{-x} na intervale $\langle 0, 1\rangle$

Pokiaľ chceme určiť k také, že pravdepodobnosť nájdenia kolízie narodeninovým útokom je väčšia ako zvolená konštanta $\varepsilon > 0$, musíme riešiť nerovnosť $\mathrm{Pr}_{\mathrm{col}} \geq \varepsilon$:

$$\begin{aligned} 1 - e^{-\frac{k(k-1)}{2n}} &\geq \varepsilon \\ 2n \cdot \ln(1-\varepsilon) &\geq -k(k-1) \\ k^2 - k - 2n \cdot \ln\frac{1}{1-\varepsilon} &\geq 0 \end{aligned}$$

Riešením príslušnej kvadratickej nerovnosti dostaneme dolný odhad počtu zvolených vzorov:

$$k \geq \frac{1}{2} + \sqrt{\frac{1}{4} + 2n \cdot \ln\frac{1}{1-\varepsilon}} \approx \sqrt{n} \cdot \sqrt{2 \cdot \ln\frac{1}{1-\varepsilon}}.$$

Teda ak ε je konštanta, tak $k = \Omega(\sqrt{n})$. Zložitosť narodeninového útoku, keď pravdepodobnosť nájdenia kolízie je významná (povedzme $1/2$ alebo $2/3$), je preto $O(\sqrt{n})$.

Poznámka. Narodeninový útok bol pomenovaný podľa tzv. „narodeninového paradoxu“ (birthday paradox). Koľko ľudí potrebujeme mať v skupine, aby pravdepodobnosť toho, že niekto bude mať narodeniny v zvolený deň v roku, bola aspoň $1/2$? Ľahko možno vidieť, že potrebujeme aspoň $\lceil 365/2 \rceil = 183$ ľudí. Pozmeňme otázku. Koľko ľudí je potrebných v skupine, aby pravdepodobnosť toho, že niektorí dvaja majú narodeniny v rovnaký deň, bola aspoň $1/2$? Odpoveď je (možno pre niekoho paradoxná), že stačí 23 ľudí, lebo pravdepodobnosť je v takom prípade $1 - (1 - 1/365)(1 - 2/365) \cdot \ldots \cdot (1 - 22/365) \approx 0{,}507$. Veľký rozdiel v počtoch ľudí v odpovediach na uvedené otázky sa zvykne označovať ako narodeninový paradox. Poznamenajme, že v druhom prípade sme v podstate hľadali kolízie v dátumoch narodenia, keď hašovacia funkcia je zobrazenie z množiny všetkých ľudí do množiny 365 dní v roku.

Narodeninový útok je dôvod, prečo sa v praxi požaduje dvojnásobná dĺžka výstupu hašovacej funkcie v porovnaní s dĺžkou kľúča symetrických šifrovacích algoritmov. Zložitosť všeobecného útoku na šifrovací algoritmus, teda útoku úplným preberaním, je $O(2^k)$, ak množina kľúčov je $\{0,1\}^k$. Na vynútenie rovnakej zložitosti všeobecného útoku na hašovaciu funkciu, teda narodeninového útoku, je potrebná množina hodnôt hašovacej funkcie $O(2^{2k})$.

10.2 Konštrukcie hašovacích funkcií

Hašovacie funkcie možno konštruovať rozličným spôsobom – použiť výpočtovo ťažké problémy, blokové šifry, prípadne navrhnúť algoritmus špeciálne pre hašovaciu funkciu.

Konštrukcie založené na výpočtovo ťažkých problémoch umožňujú dokázať, že hašovacia funkcia má požadované vlastnosti (jednosmernosť, odolnosť voči kolíziám), predpokladajúc zložitosť nejakého problému, napríklad diskrétneho logaritmu [**?**], faktorizácie [**?**] a podobne. Na druhej strane, takéto hašovacie funkcie sú (v porovnaní s inými konštrukciami) veľmi pomalé a v praxi sa nepoužívajú.

Konštrukcie z blokových šifier

Pri konštrukcii z blokových šifier sa vstupná správa rozdelí na bloky dĺžky zodpovedajúcej dĺžke bloku použitej šifry, resp. dĺžke kľúča (v závislosti na spôsobe, akým sa počíta odtlačok). Nech m je správa a označme jej rozdelenie na bloky $m_1 m_2 \dots m_t$. Nech E označuje šifrovaciu transformáciu. Existujú viaceré bezpečné konštrukcie, ak použitá šifra má požadované vlastnosti. Uvedieme dve z nich.

Matyas, Meyer a Oseas navrhli nasledujúcu schému:

$$X_i = E_{X_{i-1}}(m_i) \oplus m_i, \text{ pre } i = 1, \dots, t,$$

kde výsledný odtlačok je hodnota X_t, pričom X_0 je pevne zvolený inicializačný vektor. Iná schéma, ktorej autormi sú Davies a Meyer, počíta odtlačok takto:

$$X_i = E_{m_i}(X_{i-1}) \oplus X_{i-1}, \text{ pre } i = 1, \dots, t.$$

Na záver pripomeňme, že dĺžka bloku musí byť dostatočne dlhá, aby sa zamedzilo praktickej realizácii narodeninového útoku.

Iterované hašovacie funkcie

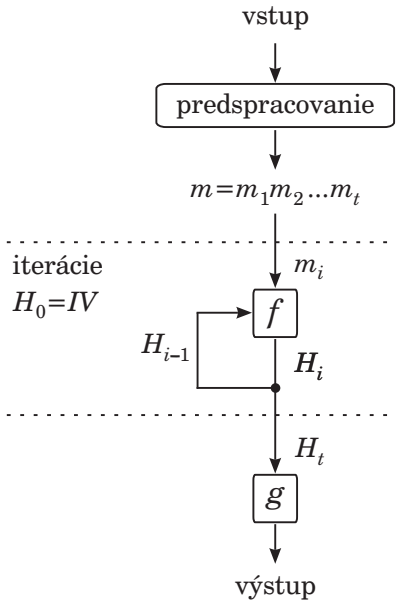
Iterované hašovacie funkcie spracúvajú vstupné údaje v blokoch pevnej dĺžky. Keďže dĺžka vstupu nemusí byť násobkom bloku, úvodná transformácia (predspracovanie) zarovná vstup (prípadne ho ešte upraví, napr. doplní zápis dĺžky vstupu). Následne rozdelí vstup na požadované bloky – označme ich m_1, m_2, …, m_t.

Bloky vstupu sa postupne spracúvajú v „kompresnej“ funkcii f a počíta sa priebežná hodnota odtlačku:

$$\begin{aligned} H_0 &= IV, \\ H_i &= f(m_i, H_{i-1}), \qquad i = 1, \dots, t, \end{aligned}$$

kde IV je pre danú hašovaciu funkciu konštantný inicializačný vektor. Výstupom hašovacej funkcie (odtlačkom) je hodnota H_t, v niektorých konštrukciách ešte spracovaná výstupnou funkciou g. Ilustratívne znázornenie iterovanej hašovacej funkcie je na obrázku 10.2.

Konštrukcia iterovaných hašovacích funkcií je najčastejšou konštrukciou prakticky používaných hašovacích funkcií. Dôvodom je aj skutočnosť, že ak použitá kompresná funkcia má vhodné vlastnosti, možno dobré vlastnosti dokázať aj pre výslednú



Obrázok 10.2: Iterovaná hašovacia funkcia

hašovaciu funkciu (samozrejme, pri vhodnej konštrukcii). Príkladom je Merkleho-Damgårdova konštrukcia, ktorá rozširuje funkciu $f : \{0,1\}^{n+r+1} \to \{0,1\}^n$ (pre $r \geq 1$) odolnú voči kolíziám na funkciu $h : \{0,1\}^* \to \{0,1\}^n$, takisto odolnú voči kolíziám:

1. Nech x je vstup dĺžky l bitov. Rozdeľme x na bloky $x_1, x_2, \ldots, x_t$ dĺžky r bitov, pričom v prípade potreby doplníme posledný blok nulami.
2. Doplníme ďalší blok x_{t+1}, obsahujúci binárny zápis l (ak $l \geq 2^r$, bude doplnených blokov viac).
3. Postupne počítame $h(x) = H_{t+1}$:

$$H_1 = f(0^{n+1} \,||\, x_1)$$
$$H_i = f(H_{i-1} \,||\, 1 \,||\, x_i) \qquad i = 2, \ldots, t+1.$$

Lema 10. *Nech funkcia $f : \{0,1\}^{n+r+1} \to \{0,1\}^n$ (pre $r \geq 1$) je odolná voči kolíziám. Potom aj funkcia $h : \{0,1\}^* \to \{0,1\}^n$ určená vyššie uvedeným postupom je odolná voči kolíziám.*

Dôkaz. Sporom. Nech vieme efektívne nájsť $x \neq x'$ také, že $h(x) = h(x')$. Nech dĺžka x je l a dĺžka x' je l'. Označme počet blokov vstupov (už po doplnení reprezentácie dĺžok) $t+1$ a $t'+1$. Rozlíšme dva prípady: $t = t'$ a $t \neq t'$.

1. ($t = t'$) Keďže $h(x) = h(x')$, tak $f(H_t \,||\, 1 \,||\, x_{t+1}) = f(H'_t \,||\, 1 \,||\, x'_{t+1})$. Buď sú argumenty funkcie f rôzne (a teda máme spor s odolnosťou f voči kolíziám), alebo sú rovnaké. Potom $x_{t+1} = x'_{t+1}$ a $H_t = H'_t$, teda $f(H_{t-1} \,||\, 1 \,||\, x_t) = f(H'_{t-1} \,||\, 1 \,||\, x'_t)$. Opäť, pri rôznosti argumentov dostávame spor, pri rovnosti pokračujeme ďalej. Postupne dostaneme $f(0^{n+1} \,||\, x_1) = f(0^{n+1} \,||\, x'_1)$. Keďže $x \neq x'$ (doposiaľ boli argumenty f rovnaké, t.j. $x_2 = x'_2, \ldots, x_{t+1} = x'_{t+1}$), tak $x_1 \neq x'_1$ a našli sme kolíziu v f. To je spor s predpokladom lemy.

2. ($t \neq t'$) Bez ujmy na všeobecnosti $t < t'$. Keďže $h(x) = h(x')$, tak $f(H_t \,||\, 1 \,||\, x_{t+1}) = f(H'_{t'} \,||\, 1 \,||\, x'_{t'+1})$. Použijúc rovnaké argumenty ako v predchádzajúcom prípade buď dostaneme spor (pre rôznosť argumentov funkcie f), alebo „rozbalíme“ konštrukciu až na začiatok. Vtedy máme rovnosť $f(0^{n+1} \,||\, x_1) = f(H'_{t'-t} \,||\, 1 \,||\, x'_{t'-t+1})$. Argumenty f sú rôzne (líšia sa aspoň v $n+1$. bite), čo je spor s predpokladom lemy (našli sme kolíziu v f).

□

10.3 Secure Hash Algorithms (SHA)

Hašovacie funkcie SHA-256, SHA-384 a SHA-512 sú súčasťou štandardu SHS (Secure Hash Standard) [SHS02]. Súčasťou štandardu je aj staršia hašovacia funkcia SHA-1. Číslo za pomlčkou pri nových hašovacích funkciách znamená dĺžku odtlačku, ktorý daná funkcia počíta. Poznamenajme, že trojica dĺžok je dvojnásobkom štandardizovaných dĺžok šifrovacieho kľúča v algoritme AES. Všetky hašovacie funkcie definované v SHS sú príkladom iterovaných hašovacích funkcií.

Cieľom tejto časti je predstaviť štruktúru modernej hašovacej funkcie. V ďalšom popíšeme hlavné časti konštrukcie algoritmu SHA-256.

SHA-256 je definovaná pre správy dĺžky menšej ako 2^{64} bitov, ktoré spracúva po blokoch dĺžky 512 bitov. Všetky výpočty v algoritme sú realizované na 32-bitových slovách. Úvodnou transformáciou vstupných údajov je ich zarovnanie tak, aby ich dĺžka bola násobkom 512 bitov:
Za správu sa pridá bit 1 a posledných 64 bitov je vyhradených na zápis dĺžky správy v bitoch. Medzi bit 1 a zápis dĺžky sa vloží potrebný počet núl. Mimochodom, rovnaké zarovnanie používa aj funkcia SHA-1.

Medzivýsledok hašovania po spracovaní i blokov vstupu označíme $H^{(i)}$. Tento medzivýsledok má 256 bitov a je rozdelený na 8 slov $H_0^{(i)}, \ldots, H_7^{(i)}$. Hodnota $H^{(0)}$ je definovaná ako konštantný inicializačný vektor.

Z každého bloku vstupu, rozdeleného na 16 slov (označme ich $m_0, \ldots, m_{15}$), sa najskôr vypočíta postupnosť 64 slov $W_0, \ldots, W_{63}$:

$$W_t = \begin{cases} m_t & \text{pre } 0 \leq t \leq 15, \\ G_1(W_{t-2}) + W_{t-7} + G_0(W_{t-15}) + W_{t-16} & \text{pre } 16 \leq t \leq 63. \end{cases}$$

Sčítanie je interpretované modulo 2^{32} a funkcie G_0, G_1 sú definované takto (RR^k je cyklická rotácia vpravo o k bitov a SR posun vpravo o k bitov):

$$G_0(x) = RR^7(x) \oplus RR^{18}(x) \oplus SR^3(x)$$
$$G_1(x) = RR^{17}(x) \oplus RR^{19}(x) \oplus SR^{10}(x)$$

Následne sa slová W_t postupne spracujú v cykle. V cykle vystupujú dočasné premenné (slová) a, b, c, d, e, f, g, h a najskôr sa vykoná nasledujúce priradenie:

$$a \leftarrow H_0^{(i-1)}; \quad b \leftarrow H_1^{(i-1)}; \quad c \leftarrow H_2^{(i-1)}; \quad d \leftarrow H_3^{(i-1)};$$
$$e \leftarrow H_4^{(i-1)}; \quad f \leftarrow H_5^{(i-1)}; \quad g \leftarrow H_6^{(i-1)}; \quad h \leftarrow H_7^{(i-1)}$$

V cykle pre $t = 0, \ldots, 63$ sa dočasné premenné modifikujú takto:

$$a \leftarrow T_1 + T_2; \qquad b \leftarrow a; \qquad c \leftarrow b; \qquad d \leftarrow c;$$
$$e \leftarrow d + T_1; \qquad f \leftarrow e; \qquad g \leftarrow f; \qquad h \leftarrow g$$

kde hodnoty T_1 a T_2 sú určené vzťahmi (funkcie F_0, F_1, Ch a Maj sú logické funkcie vyhodnocované po bitoch a K_t sú v algoritme pevne definované konštanty):

$$\begin{aligned} T_1 &= h + F_1(e) + Ch(e, f, g) + K_t + W_t \\ T_2 &= F_0(a) + Maj(a, b, c) \\ F_0(x) &= RR^6(x) \oplus RR^{11}(x) \oplus RR^{25}(x) \\ F_1(x) &= RR^2(x) \oplus RR^{13}(x) \oplus RR^{22}(x) \\ Ch(x, y, z) &= (x \wedge y) \oplus (\neg\, x \wedge z) \\ Maj(x, y, z) &= (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) \end{aligned}$$

Záverečný výpočet $H^{(i)}$ je daný týmito priradeniami:

$$a \leftarrow a + H_0^{(i-1)}; \quad b \leftarrow b + H_1^{(i-1)}; \quad c \leftarrow c + H_2^{(i-1)}; \quad d \leftarrow d + H_3^{(i-1)};$$
$$e \leftarrow e + H_4^{(i-1)}; \quad f \leftarrow f + H_5^{(i-1)}; \quad g \leftarrow g + H_6^{(i-1)}; \quad h \leftarrow h + H_7^{(i-1)}$$

Odtlačok správy je daný hodnotou $H^{(i)}$ po spracovaní posledného bloku zarovnanej správy.

10.4 MAC

Existujú situácie, keď pri komunikácii nie je potrebné zabezpečiť dôvernosť prenášaných údajov, ale ich integrita a autentickosť je dôležitá. Teda šifrovanie údajov nie je potrebné a každá z komunikujúcich strán po prijatí údajov (správy) potrebuje vedieť, že údaje neboli počas prenosu zmenené a ich autorom je naozaj účastník, s ktorým komunikuje. Práve pre takýto účel existuje konštrukcia autentizačných kódov správ (message authentication code – MAC). Hrubo povedané, ide o hašovacie funkcie s kľúčom. Odtlačok správ závisí nielen na správe samotnej, ale aj na kľúči, ktorý poznajú len komunikujúce strany. V komunikácii sa okrem správy posiela aj zodpovedajúci MAC. Vytvoriť korektný MAC a overiť jeho korektnosť možno len so znalosťou použitého kľúča.

Najčastejšie konštrukcie MAC sú z blokových symetrických šifier a z hašovacích funkcií. MAC založené na symetrických šifrách sú obvykle pomalšie.

Poznámka. Elementárne konštrukcie z hašovacích funkcií, keď kľúč zreťazíme so správou a vypočítame odtlačok takto upravenej správy majú bezpečnostné slabiny. Ak je použitá hašovacia funkcia iteratívna a kľúč pripájame pred správu (označme ju x), tak z výsledného MAC je možné dopočítať MAC bez znalosti kľúča pre ľubovoľnú správu, ktorá začína x, keďže následné iterácie už na kľúči nezávisia.

Ak je použitá hašovacia funkcia iteratívna a kľúč pripájame za správu, tak po získaní kolízie v hašovacej funkcii je aj MAC pre kolíznu dvojicu správ rovnaký.

CBC-MAC

CBC-MAC je konštrukcia MAC z blokových šifier. Myšlienka spočíva v použití CBC módu blokovej šifry (pozri časť 3.3), pričom posledný blok šifrového textu bude výstupom MAC. Pre zvýšenie bezpečnosti schémy sa posledný blok ešte dodatočne spracuje.

Nech $m = m_1 m_2 \dots m_t$ je správa rozdelená na bloky dĺžky zodpovedajúcej dĺžke bloku použitého šifrovacieho algoritmu. Šifrovaciu transformáciu označíme E a dešifrovaciu transformáciu D. Výpočet MAC (označíme M) prebieha takto:

1. $H_0 = IV$, kde IV je inicializačný vektor
2. $H_i = E_k(H_{i-1} \oplus m_i)$, pre $i = 1, \dots, t$
3. $M = E_k(D_{k'}(H_t))$

Konštrukcia využíva dva symetrické kľúče k a k'. Posledný blok správy je spracovaný trojitým šifrovaním v móde EDE. Požadujeme, aby $k \neq k'$, v praxi možno kľúč k' odvodiť z k, napríklad $k' = E_k(k)$, $k' = \overline{k}$ (negácia k) a podobne. Ak sú kľúče rovnaké, tak $M = H_t$ (akoby sa tretí krok neuskutočnil) a schéma má nasledujúcu bezpečnostnú slabinu.

Nech $k = k'$. Nech útočník pozná dvojice (m, M) a (m', M') pre vstupy m a m' dĺžky jedného bloku. Teda $M = E_k(m)$ a $M' = E_k(m')$. Predpokladajme, že útočník dokáže získať MAC pre takúto zvolenú správu dĺžky dvoch blokov: $m \,||\, x$, pre nejaké x (operácia $||$ označuje zreťazenie). MAC tejto správy má hodnotu $E_k(M \oplus x)$. Rovnakú hodnotu má aj MAC dvojblokovej správy $m' \,||\, M \oplus M' \oplus x$:

$$E_k(E_k(m') \oplus M \oplus M' \oplus x) = E_k(M' \oplus M \oplus M' \oplus x) = E_k(M \oplus x).$$

HMAC

HMAC je najznámejšou a najpoužívanejšou metódou konštrukcie MAC z hašovacej funkcie. Výhodou HMAC je, že ak iterovaná hašovacia funkcia použitá v HMAC spĺňa isté predpoklady, výsledná konštrukcia je dokázateľne bezpečná [BCK96].

Označme hašovaciu funkciu H, kľúč k a nech x sú údaje (správa), pre ktoré sa MAC počíta. Výpočet HMAC je daný nasledujúcim vzťahom:

$$\mathrm{HMAC}_k(x) = H(k \oplus \mathrm{opad} \,||\, H(k \oplus \mathrm{ipad} \,||\, x)),$$

kde operácia $\oplus$ označuje XOR a operácia $||$ zreťazenie. Hodnoty opad a ipad sú rôzne reťazce dostatočnej dĺžky. Napriek dvojnásobnému použitiu hašovacej funkcie má výpočet HMAC v podstate rovnakú zložitosť ako výpočet odtlačku údajov, keďže vonkajšie použitie H počíta odtlačok len krátkeho reťazca.

Poznámka. Konštrukcia HMAC je internetový štandard RFC 2104 a v praxi sa používa napríklad v protokoloch IPSec a SSL/TLS. V tomto štandarde sú pre hašovacie funkcie MD5 a SHA-1 zvolené konštanty opad a ipad ako 64 krát opakovaný bajt 0x5C (pre opad) a 0x36 (pre ipad).

Použitie MAC na zabezpečenie dôvernosti

Zaujímavý spôsob použitia MAC na zabezpečenie dôvernosti údajov navrhol Ronald Rivest v roku 1998 [Riv98]. Šifrovanie, teda trasnformácia údajov do podoby nezrozumiteľnej bez znalosti vhodného kľúča, nemusí byť jediným prostriedkom dosiahnutia dôvernosti uložených alebo prenášaných údajov. Údaje možno „ukryť“ v množstve ďalších, dodaných údajov tak, že pôvodné síce budú prítomné v otvorenom tvare, avšak odlíšiť ich od ostatných bude pre útočníka ťažké.

Ilustrujme ideu na príklade. Účastníci A a B komunikujú a zdieľajú tajný kľúč pre použitie v MAC. Označme tento kľúč k. Správu posielanú v kroku i označíme m_i a MAC reťazca x označíme $H_k(x)$. Pôvodný prenos správy rozdelený na päť paketov by moho vyzerať takto:

1. $A \to B$: $(1, \text{„Ahoj Peter,“}, H_k(1 \,||\, m_1))$
2. $A \to B$: $(2, \text{„stretneme sa“}, H_k(2 \,||\, m_2))$
3. $A \to B$: $(3, \text{„zajtra o 17:30“}, H_k(3 \,||\, m_3))$
4. $A \to B$: $(4, \text{„na Hlavnom nám.“}, H_k(4 \,||\, m_4))$
5. $A \to B$: $(5, \text{„Ivan“}, H_k(5 \,||\, m_5))$

Poznamenajme, že pakety sú očíslované a súčasťou každého paketu je MAC vypočítaný z čísla paketu a príslušného textu. Doplnením falošných paketov, v ktorých sú rovnaké čísla paketov, ale namiesto MAC zvolíme náhodný reťazec bitov požadovanej dĺžky, bude komunikácia takáto:

1. $A \to B$: $(1, \text{„Ahoj Peter,“}, H_k(1 \,||\, m_1))$
1'. $A \to B$: $(1, \text{„Ahoj Lenka,“}, \dots)$
2'. $A \to B$: $(2, \text{„koncert začne“}, \dots)$
2. $A \to B$: $(2, \text{„stretneme sa“}, H_k(2 \,||\, m_2))$
3. $A \to B$: $(3, \text{„zajtra o 17:30“}, H_k(3 \,||\, m_3))$
3'. $A \to B$: $(3, \text{„dnes podvečer“}, \dots)$
4'. $A \to B$: $(4, \text{„na železničnej stanici“}, \dots)$
4. $A \to B$: $(4, \text{„na Hlavnom nám.“}, H_k(4 \,||\, m_4))$
5. $A \to B$: $(5, \text{„Ivan“}, H_k(5 \,||\, m_5))$
5'. $A \to B$: $(5, \text{„Monika“}, \dots)$

Účastník B dokáže ľahko zistiť, vďaka kľúču k a hodnotám MAC, ktoré pakety sú správne. Na druhej strane útočník nepozná k a nevie odlíšiť správne pakety od nesprávnych. V príklade môže byť z prenesených paketov vyskladaných 32 viac-menej zmysluplných správ. Útočník nevie rozlíšiť, ktorá z nich je tá správna.

Skracovaním dĺžky textov v paketoch sa zjednodušuje úloha dogenerovať falošné pakety tak, aby rôzne kombinácie paketov (skutočných aj falošných) boli zmysluplné

a útočník nedokázal odlíšiť správnu správu. Limitne môžeme obmedziť obsah textu v pakete na jeden bit. Pri prenose správy $10011\ldots$ budú prenášané tieto pakety:

$A \to B$	$A \to B$
$(1, 0, \ldots)$	$(4, 0, \ldots)$
$(1, 1, H_k(1 \,\|\| 1))$	$(4, 1, H_k(4 \,\|\| 1))$
$(2, 0, H_k(2 \,\|\| 0))$	$(5, 0, \ldots)$
$(2, 1, \ldots)$	$(5, 1, H_k(5 \,\|\| 1))$
$(3, 0, H_k(3 \,\|\| 0))$	$\ldots$
$(3, 1, \ldots)$	

V takomto prípade útočník nedokáže identifikovať, bez znalosti k, prenášanú správu spomedzi potenciálnych 2^n správ (teda ľubovoľná správa dĺžky n bitov je prípustná, kde n označuje dĺžku správy). Z hľadiska bezpečnosti je dôležité, aby pri použití rovnakého kľúča k mali prenášané pakety rôzne čísla. V opačnom prípade, pri zopakovaní číslovania paketov pre prenos dvoch správ, útočník vie zistiť hodnotu bitu v pakete (ak sa príslušné MAC zhodujú) alebo ich komplementaritu (ak sa MAC nezhodujú).

Samozrejme, takýto prenos údajov má nedostatok vo veľkom náraste komunikačnej zložitosti. Praktická použiteľnosť schémy je preto malá. Na druhej strane poskytuje zaujímavý pohľad na jeden z bezpečnostných atribútov údajov – dôvernosť.

11 Digitálne podpisy

Digitálne podpisy sú kryptografické konštrukcie, ktorých cieľom je zabezpečiť autentickosť a integritu (prípadne aj nepopretie autorstva) správ a dokumentov v elektronickom prostredí. Digitálny ekvivalent klasických „papierových“ podpisov musí brať na zreteľ aj špecifiká prostredia, napríklad jednoduché kopírovanie údajov. Preto digitálny podpis závisí nielen na identite podpisujúceho, ale aj na podpisovanom dokumente. V opačnom prípade by bolo možné údaj reprezentujúci digitálny podpis pripojiť k ľubovoľnému dokumentu. Základné vlastnosti digitálnych podpisov, podmieňujúce ich použiteľnosť, sú:

- používateľ vie efektívne vytvoriť digitálny podpis k danému dokumentu;
- ktokoľvek vie efektívne overiť korektnosť digitálneho podpisu;
- nikto nemôže efektívne vytvoriť digitálny podpis iného používateľa k dokumentu, ktorý tento používateľ nepodpísal.

Z tretej vlastnosti vyplýva, že *nie je* možné:

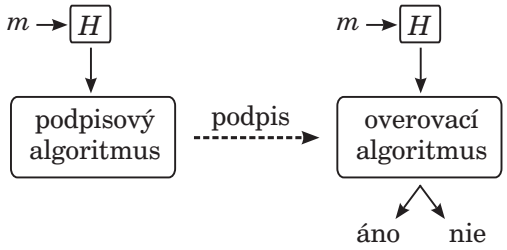
- zobrať digitálny podpis z jedného dokumentu, pripojiť ho k inému dokumentu a vytvoriť pritom korektne podpísaný dokument;
- zmeniť podpísaný dokument so zachovaním korektnosti digitálneho podpisu;
- poprieť vlastný digitálny podpis.

Uvedené bezpečnostné vlastnosti majú neformálny charakter. Pre ich naplnenie je potrebné prihliadať okrem kryptografickej stránky riešenia aj na implementáciu, otázky súvisiace s manažmentom kľúčov, časom podpísania a podobne. V ďalšom sa sústredíme práve na kryptografickú stránku realizácie digitálnych podpisov. Táto pozostáva z tzv. schémy digitálnych podpisov, obsahujúcej podpisový algoritmus a

overovací algoritmus. Podpisový algoritmus je predpis určujúci spôsob, ako vytvoriť digitálny podpis. Obvykle zahŕňa aj spôsob formátu a zápisu dát, týmto otázkam sa v tejto kapitole nebudeme venovať. Overovací algoritmus je postup overenia korektnosti digitálneho podpisu.

Schémy digitálnych podpisov sú v praxi takmer výlučne realizované použitím systémov s verejnými kľúčmi s pomocou kryptografických hašovacích funkcií. V skutočnosti sa namiesto dokumentu podpisuje jeho odtlačok. Použitie hašovacích funkcií je motivované praktickými aj bezpečnostnými dôvodmi. Podpisovanie dokumentov by pomalšími asymetrickými systémami trvalo neúnosne dlho. Výpočet odtlačku dokumentu a jeho podpísanie je podstatne rýchlejšie. Navyše, vlastnosti kryptografických hašovacích funkcií – jednosmernosť a odolnosť voči kolíziam – pomáhajú chrániť schémy pred niektorými útokmi.

Všeobecný náčrt schémy digitálnych podpisov je na obrázku 11.1, kde m označuje podpisovaný dokument a H hašovaciu funkciu.



Obrázok 11.1: Schéma digitálneho podpisu

Pre bezpečnosť tohto postupu je nevyhnutné, aby bola hašovacia funkcia H jednosmerná a odolná voči kolíziam. Jednosmernosť H zabezpečuje napríklad odolnosť niektorých schém pred falšovaním náhodnej správy, ako uvidme v častiach 11.1 alebo 11.2. Ak H nie je odolná voči kolíziam, môžeme skonštruovať dva dokumenty $m_1 \neq m_2$ také, že $H(m_1) = H(m_2)$. Ak jeden z nich používateľ podpíše, tak výsledný digitálny podpis je zároveň korektným podpisom druhého dokumentu. Samozrejme, kryptografická sila použitého asymetrického systému je tiež nutnou podmienkou bezpečnosti schémy digitálneho podpisu.

11.1 RSA schéma

Realizácia digitálnych podpisov je zhodná s realizáciou asymetrického šifrovacieho systému RSA. V tomto prípade slúži šifrovacia transformácia ako overovací algoritmus a dešifrovacia transformácia ako podpisový algoritmus. Používateľ inicializuje svoju inštanciu RSA (časť 5.1). Budeme používať zaužívané označenie – súkromný kľúč (exponent) označíme d a verejný kľúč tvorí verejný exponent e a hodnota n. Nech m je podpisovaný (prípadne overovaný) dokument. Nech H je hašovacia funkcia, ktorej výstup je v $\mathbb{Z}_n$. Digitálny podpis dokumentu m označíme s.

Podpisovanie: $s = H(m)^d \bmod n$.

Overovanie: podpis s dokumentu m je korektný práve vtedy, keď $s^e \bmod n = H(m)$.

Korektnosť tejto podpisovej schémy vyplýva z toho, že v RSA systéme sú šifrovacia a dešifrovacia transformácia navzájom inverzné funkcie:

$$s^e \bmod n = H(m)^{ed} \bmod n = H(m).$$

V schéme predpokladáme, že $H(m) \in \mathbb{Z}_n$. V prípade väčšej dĺžky $H(m)$ by sme mohli postupovať ako pri asymetrickom šifrovaní/dešifrovaní a podpisovanie a overovanie rozdeliť do blokov. Na druhej strane je uvedený predpoklad plne odôvodnený, keďže v praktických podmienkach je dĺžka (v bitoch) modulu n niekoľkokrát väčšia ako dĺžka výstupu hašovacej funkcie H. To môže so sebou priniesť iný problém: ak je výstupom H napr. 160 bitov dlhý odtlačok dokumentu, tak pri 1024 bitovom module n je priestor potenciálnych textov pre použitý RSA systém výrazne redukovaný. Aby sa predišlo možným bezpečnostným problémom, pred aplikáciou súkromnej transformácie môžeme $H(m)$ predĺžiť na dĺžku zodpovedajúcu dĺžke n. Predĺženie (padding) môže byť napríklad náhodné. Prirodzene, pri overovaní digitálneho podpisu tieto nadbytočné bity ignorujeme.

Výpočtová efektívnosť RSA schémy je ovplyvnená voľbou súkromného a verejného kľúča, použitím Čínskej zvyškovej vety pri umocňovaní so súkromným exponentom d a podobne. Tieto otázky už boli rozoberané pri popise asymetrického šifrovacieho systému RSA. Poznamenajme len, že pri voľbe krátkeho verejného exponentu je možné očakávať podstatne rýchlejšie overovanie digitálnych podpisov ako ich vytváranie.

Bezpečnosť RSA schémy pre digitálne podpisy závisí na zložitosti problému faktorizácie a diskusia o bezpečnostných aspektoch asymetrického šifrovacieho systému RSA sa týka aj tejto schémy. K bezpečnosti schémy prispieva podstatnou mierou aj hašovacia funkcia, od ktorej sa očakáva jednosmernosť a odolnosť voči kolíziám.

Falšovanie náhodnej správy

Predpokladajme, že pri realizácii RSA schémy nepoužijeme hašovaciu funkciu. Priamočiaro použijeme dešifrovaciu (na podpisovanie) a šifrovaciu (na overovanie) transformáciu RSA systému na dokument m. Pri dlhších dokumentoch tieto rozdelíme na bloky, ktoré podpisujeme a overujeme zvlášť. Samozrejme, nedostatkom tohto riešenia je neefektívnosť – podpisovanie rozsiahlych dokumentov trvá dlho. Druhou, kryptograficky závažnejšou nevýhodou je, že z dvoch podpísaných dokumentov vie ktokoľvek bez vedomia používateľa vytvoriť tretí dokument, ktorý bude korektne podpísaný. To je možné vďaka multiplikatívnej štruktúre RSA:

$$m_1^d \cdot m_2^d \bmod n = (m_1 m_2)^d \bmod n.$$

Nech $\langle m_1, m_1^d \bmod n\rangle$ a $\langle m_2, m_2^d \bmod n\rangle$ sú dvojice dokumentov spolu s ich digitálnymi podpismi. Potom môžeme vytvoriť nový dokument s korektným podpisom, pričom nie je potrebné poznať súkromný kľúč používateľa:

$$\langle m_1 m_2 \bmod n, (m_1 m_2)^d \bmod n\rangle.$$

Pre jednoduchosť predpokladáme $m_1, m_2 \in \mathbb{Z}_n$. Dokonca by stačil len jeden podpísaný dokument: $m_1 = m_2$. Prirodzene, s vysokou pravdepodobnosťou bude vytvorený dokument nezmyselný, ale možnosť vytvárania falošných dokumentov s korektnými digitálnymi podpismi znižuje dôveryhodnosť schémy. Poznamenajme, že

použitie hašovacej funkcie problém odstraňuje – H obvykle nemá multiplikatívnu vlastnosť $H(m_1 m_2) = H(m_1)H(m_2)$.

Iný spôsob vytvorenia falošného dokumentu a zodpovedajúceho podpisu spočíva v tom, že útočník zvolí náhodný podpis $s \in \mathbb{Z}_n$ a vypočíta $m = s^e \bmod n$. Ak v RSA schéme nepoužívame hašovaciu funkciu, je hodnota s korektným podpisom dokumentu m. Tento útok sa zvykne nazývať „falšovanie náhodnej správy“ (random message forgery). Pri použití hašovacej funkcie je útok nerealizovateľný, vďaka jednosmernosti H.

11.2 ElGamalova schéma

ElGamalov asymetrický šifrovací systém bol popísaný v kapitole 9. Keďže šifrovacia a dešifrovacia transformácia nie sú navzájom inverzné, nie je možné použiť tento systém pre digitálne podpisy tak priamočiaro ako v prípade RSA. Napriek tomu bude inicializácia zhodná s inicializáciou šifrovacieho systému.

Inicializácia. Zvolíme veľké prvočíslo p a prvok $g \in \mathbb{Z}_p^*$, pričom g môže ale nemusí byť generátor grupy $(\mathbb{Z}_p^*, \cdot)$. Hodnoty p a g môžu byť spoločné pre viacerých používateľov. Ďalej zvolíme náhodne $x \in_R \{2, 3, \ldots, p-2\}$ a vypočítame $y = g^x \bmod p$. Verejný kľúč, trojica (y, p, g), slúži na overovanie podpisu. Súkromný kľúč je hodnota x a využíva sa pri podpisovaní.

Podpisovanie. Nech m je dokument, ktorý chceme podpísať. Nech H je hašovacia funkcia, ktorej výstup je v $\mathbb{Z}_p^*$. Postupujeme takto:

1. Zvolíme náhodne $k \in_R \{1, \ldots, p-2\}$ také, že $\mathrm{nsd}(k, p-1) = 1$.
2. Vypočítame $r = g^k \bmod p$.
3. Vypočítame s také, že je splnená nasledujúca rovnosť:

$$H(m) = (xr + ks) \bmod (p-1). \tag{11.1}$$

4. Digitálny podpis správy m je dvojica $\langle r, s \rangle$.

Hodnotu s je možné pri podpisovaní vypočítať takto:

$$s = (H(m) - xr)k^{-1} \bmod (p-1).$$

Rovnica objasňuje potrebu zvoliť k v prvom kroku tak, aby bolo nesúdeliteľné s $p-1$. Vtedy existuje k hodnote k inverzný prvok modulo $p-1$.

Overovanie. Ktokoľvek, kto pozná verejný kľúč (y, p, g) a správu m, môže overiť správnosť digitálneho podpisu $\langle r, s \rangle$. Podpis sa považuje za korektný práve vtedy, keď

$$y^r \cdot r^s \equiv g^{H(m)} \pmod{p} \quad \wedge \quad 1 \leq r < p.$$

Korektnosť vyplýva z nasledujúceho vzťahu (posledná kongruencia je dôsledkom malej Fermatovej vety):

$$y^r \cdot r^s \equiv g^{xr} \cdot g^{ks} \equiv g^{rx+ks} \equiv g^{H(m)} \pmod{p}.$$

Výpočtová náročnosť ElGamalovej schémy je ovplyvnená najmä modulárnymi umocneniami. Podpisovanie si vyžaduje jedno a overovanie dokonca tri umocnenia. Schéma umožňuje hodnotu r predvypočítať (je potrebné si popri predvypočítanej hodnote pamätať aj hodnotu k, prípadne k^{-1}). Potom je výpočet digitálneho podpisu rýchly – len dve modulárne násobenia. Takáto úprava si vyžaduje držať v tajnosti hodnotu k, resp. k^{-1}. Ak sa útočník dozvie hodnotu k použitú pri podpisovaní, dokáže zo vzťahu (11.1) vypočítať súkromný kľúč používateľa (teda „rozbiť“ celú podpisovú schému):

$$x = (H(m) - ks) \cdot r^{-1} \bmod (p-1).$$

Bezpečnosť ElGamalovej schémy digitálneho podpisu sa opiera o zložitosť problému diskrétneho logaritmu a o bezpečnosť použitej hašovacej funkcie H. Poznamenajme, že podobne ako v prípade ElGamalovho asymetrického šifrovacieho systému, nie je známy dôkaz, že „rozbitie“ tejto schémy je ekvivalentné výpočtu diskrétneho logaritmu.

Potreba kontroly $1 \leq r < p$

Ukážeme, že testovanie $1 \leq r < p$ je pri overovaní digitálneho podpisu potrebné. Predpokladajme, že by sa táto nerovnosť pri overovaní nekontrolovala. Nech $\langle r, s\rangle$ je korektný podpis dokumentu m. Teda $y^r \cdot r^s \equiv g^{H(m)} \pmod p$. Vytvoríme digitálny podpis dokumentu m' ($m' \neq m$), bez znalosti súkromného kľúča. Najskôr vypočítame

$$u = H(m') \cdot H(m)^{-1} \bmod (p-1).$$

Predpokladáme, že príslušný inverzný prvok modulo $p-1$ existuje. Potom platí:

$$g^{H(m')} \equiv g^{H(m)\cdot u} \equiv y^{ru} \cdot r^{su} \pmod p.$$

Vypočítame dvojicu $\langle r', s'\rangle$ – digitálny podpis dokumentu m'. Položíme $s' = su \bmod (p-1)$. Potrebujeme, aby hľadané r' spĺňalo nasledujúce kongruencie:

$$\begin{aligned} r' &\equiv ru \pmod{p-1};\\ r' &\equiv r \pmod p. \end{aligned}$$

Keďže $\mathrm{nsd}(p-1, p) = 1$, môžeme na výpočet r' použiť Čínsku zvyškovú vetu. Dvojica $\langle r', s'\rangle$ je potom korektným digitálnym podpisom m', ak netestujeme rozsah hodnoty r'. Hodnota r' bude s vysokou pravdepodobnosťou väčšia ako p, inak by totiž muselo platiť $r' = r$, odkiaľ $u = 1$ a teda $H(m) \equiv H(m') \pmod{p-1}$. Vzhľadom na obvyklú veľkosť p a dĺžku výstupu H to znamená nájdenie kolízie pre hašovaciu funkciu H.

Poznámka. Pri výpočte r' a s' môžeme postupovať aj inak. Napríklad položíme $s' = s$ a r' vypočítame pomocou Čínskej zvyškovej vety zo sústavy

$$\begin{aligned} r' &\equiv ru \pmod{p-1};\\ r' &\equiv r^u \pmod p. \end{aligned}$$

Hodnota r' bude opäť s vysokou pravdepodobnosťou väčšia ako p. V opačnom prípade by z prvej kongruencie platilo $r' = r$, a teda nájdenie kolízie pre hašovaciu funkciu H.

Bleichenbacher [Ble96] našiel iný spôsob falšovania podpisov v ElGamalovej schéme, ak je parameter g nevhodne zvolený. Útok je úspešný, ak $g \mid (p-1)$ a g má len malé prvočíselné faktory. Preto sa treba voľbe takýchto g vyhnúť. Extrémnym príkladom zlej voľby tohto parametra je $g = 2$. Pripomeňme, že pri samotnom probléme diskrétneho logaritmu voľba generátora neovplyňuje zložitosť problému.

Viacnásobné použitie k

Použitie rovnakej hodnoty k pre podpis dvoch rôznych dokumentov je vážny bezpečnostný problém a vedie k prezradeniu súkromného kľúča. To znamená, že voľbe (pseudo)náhodného generátora a jeho inicializácii je potrebné venovať pri implementácii schémy náležitú pozornosť.

Nech je k použité pri podpisovaní dokumentov m_1 a m_2 ($m_1 \neq m_2$). Nech $\langle r, s_1 \rangle$ je podpis dokumentu m_1 a nech $\langle r, s_2 \rangle$ je podpis m_2, kde $r = g^k \bmod p$. Teda platí:

$$\begin{aligned} y^r \cdot r^{s_1} &\equiv g^{H(m_1)} \pmod{p}, \\ y^r \cdot r^{s_2} &\equiv g^{H(m_2)} \pmod{p}, \end{aligned}$$

odkiaľ po predelení dostávame

$$g^{H(m_1)-H(m_2)} \equiv g^{k(s_1-s_2)} \pmod{p}.$$

Pre zjednodušenie diskusie predpokladajme, že g je generátor $(\mathbb{Z}_p^*, \cdot)$. Potom musia byť exponenty v predchádzajúcom vzťahu navzájom kongruentné modulo $p-1$:

$$H(m_1) - H(m_2) \equiv k(s_1 - s_2) \pmod{p-1}. \tag{11.2}$$

Nech $d = \mathrm{nsd}(s_1 - s_2, p-1)$. Zo vzťahu (11.2) vyplýva $d \mid (H(m_1) - H(m_2))$. Položme

$$\begin{aligned} m' &= \frac{H(m_1) - H(m_2)}{d}, \\ s' &= \frac{s_1 - s_2}{d}, \\ p' &= \frac{p-1}{d}. \end{aligned}$$

Zo vzťahu (11.2) dostaneme $m' \equiv ks' \pmod{p'}$. Keďže $\mathrm{nsd}(s', p') = 1$, tak máme

$$k \equiv m' \cdot (s')^{-1} \pmod{p'}.$$

Teda vieme vypočítať hodnotu $k \bmod p'$. Skutočná hodnota $k \in \mathbb{Z}_p^*$ je preto vyjadriteľná v tvare

$$k = \left((m' \cdot (s')^{-1} \bmod p') + i \cdot p' \right) \bmod (p-1),$$

pre $0 \leq i \leq d-1$. Po otestovaní d kandidátov zistíme hodnotu k (správnosť k vieme otestovať pomocou vzťahu $g^k \bmod p = r$). Následne vypočítame hodnotu súkromného kľúča.

Spoločné hodnoty p a g

Bezpečnostným rizikom, spoločným pre schémy založené na probléme diskrétneho logaritmu, je použitie spoločných parametrov p a g (vo všeobecnosti grupy a generátora) pre skupinu používateľov. To sa týka schém pre digitálny podpis ako aj asymetrických šifrovacích systémov. Ak tieto hodnoty generuje nejaká nadriadená autorita podľa ľubovôle, je možné zvoliť ich tak, aby obsahovali „zadné dvierka“. To umožní tvorcovi parametrov zistiť súkromné kľúče používateľov, alebo falšovať ich digitálne podpisy [Ble96, Gor92]. Preto je dôležité zamedziť možnému „uvareniu“ špeciálnych parametrov napríklad použitím procedúry na generovanie jednotlivých parametrov schémy, ktorá umožní neskôr dokázať, že bola naozaj použitá (a teda parametre sú dôveryhodné). Príklad takej procedúry je možné nájsť v štandarde Digital Signature Standard [DSS01].

Jednoduchý spôsob, ako zamedzeniť použitiu zlých parametrov je, keď si každý používateľ vygeneruje celý systém (schému) sám. Predlžujú sa tým však verejné kľúče.

Falšovanie náhodnej správy

Použitie kryptografickej hašovacej funkcie zabraňuje možnosti skonštruovať korektne podpísanú náhodnú správu. Podobná situácia je aj v RSA schéme, pozri časť 11.1. Predpokladajme, že v ElGamalovej schéme nepoužijeme hašovaciu funkciu a podpisujeme priamo m.

Zvoľme najskôr náhodne $i, j \in_R \mathbb{Z}^*_{p-1}$. Potom postupne vypočítame:

$$\begin{aligned} r &= g^i \cdot y^j \bmod p, \\ s &= -r \cdot j^{-1} \bmod (p-1), \\ m &= -r \cdot i \cdot j^{-1} \bmod (p-1). \end{aligned}$$

Dvojica $\langle r, s \rangle$ je korektným digitálnym podpisom dokumentu m (ľahko vidieť, že $1 \leq r < p$):

$$\begin{aligned} y^r \cdot r^s &\equiv y^r \cdot (g^i \cdot y^j)^{-r \cdot j^{-1}} \equiv y^r \cdot g^{-ri \cdot j^{-1}} \cdot y^{-r} \\ &\equiv g^{-ri \cdot j^{-1}} \equiv g^m \pmod p. \end{aligned}$$

Použitie hašovacej funkcie (vďaka jej jednosmernosti) takémuto útoku zabraňuje.

11.3 Digital Signature Algorithm (DSA)

DSA algoritmus (schéma) je súčasťou štandardu DSS – Digital Signature Standard [DSS01]. Štandard okrem DSA zahŕňa aj schémy RSA a ECDSA (realizácia DSA na eliptických krivkách). Okrem toho DSS popisuje spôsob generovania parametrov pre jednotlivé schémy. V ďalšom sa budeme venovať výlučne DSA schéme.

DSA je schéma ElGamalovho typu, ktorej bezpečnosť je založená na probléme diskrétneho logaritmu. Rovnako ako pri podobných schémach, nie je ekvivalencia s týmto problémom dokázaná. Štandard predpisuje použitie DSA s hašovacou funkciou SHA-1, ktorá vytvára 160 bitov dlhý odtlačok. Symbol H v popise schémy bude preto označovať funkciu SHA-1.

Inicializácia. Zvolíme parametre p, q, g, ktoré môžu byť spoločné pre skupinu používateľov:

p – 1024 bitov dlhé, náhodne zvolené prvočíslo;
q – 160 bitov dlhé prvočíslo také, že $q \mid (p-1)$;
g – vypočítame $g = h^{(p-1)/q}$, kde $h \in_R \{2, 3, \ldots, p-2\}$ náhodne zvolené číslo také, že $h^{(p-1)/q} > 1$.

Prirodzene, pri generovaní prvočísel p, q postupujeme tak, že najskôr nájdeme q a potom hľadáme p v požadovanom tvare. Spôsob voľby g zaručuje, že g má v grupe $(\mathbb{Z}_p^*, \cdot)$ rád q. Označme rád prvku g ako rád(g). Podľa malej Fermatovej vety:

$$g^q = h^{\frac{p-1}{q} \cdot q} = 1,$$

preto rád$(g) \leq q$. Zároveň, ak $g^k = 1$, tak aj $g^{2k} = 1$, $g^{3k} = 1$, atď. To znamená, že rád$(g) \mid q$. Keďže $g > 1$ a q je prvočíslo, musí platiť rád$(g) = q$. Teda g generuje grupu prvočíselného rádu q.

Používateľ si zvolí súkromný kľúč $x \in_R \mathbb{Z}_q^*$ a vypočíta hodnotu $y = g^x \bmod p$. Verejný kľúč je štvorica (y, p, q, g).

Podpisovanie. Pri podpisovaní dokumentu m postupujeme takto:

1. Zvolíme náhodne $k \in_R \{1, \ldots q-1\}$.
2. Vypočítame $r = (g^k \bmod p) \bmod q$.
3. Vypočítame

$$s = k^{-1}(H(m) + xr) \bmod q. \tag{11.3}$$

Ak pri podpisovaní dostaneme $r = 0$ alebo $s = 0$, čo je však nepravdepodobné, vygenerujeme nové k. Dvojica $\langle r, s \rangle$ je digitálnym podpisom dokumentu m.

Overovanie. Predpokladáme, že máme k dispozícii dokument m, jeho digitálny podpis $\langle r, s \rangle$ ako aj verejný kľúč používateľa, ktorý dokument podpísal. Najskôr overíme, že $s, r \in \mathbb{Z}_q^*$. Potom vypočítame parametre:

$$u_1 = H(m) \cdot s^{-1} \bmod q,$$
$$u_2 = r \cdot s^{-1} \bmod q.$$

Digitálny podpis je korektný, keď platí rovnosť

$$(g^{u_1} \cdot y^{u_2} \bmod p) \bmod q = r. \tag{11.4}$$

Ukážme korektnosť DSA schémy. Nech $\langle r, s \rangle$ je digitálny podpis dokumentu m. Pri overovaní rovnosti (11.4) dostaneme:

$$\begin{aligned} g^{u_1} y^{u_2} \bmod p \bmod q &= g^{H(m)s^{-1}} \cdot g^{xrs^{-1}} \bmod p \bmod q \\ &= g^{s^{-1}(H(m)+xr)} \bmod p \bmod q \\ &= g^k \bmod p \bmod q \\ &= r \end{aligned}$$

Všetky operácie v exponentoch prvku g sú počítané modulo q, lebo g generuje grupu rádu q. V DSA sú obe zložky podpisu $\langle r, s\rangle$ počítané nakoniec modulo q, preto je podpis v DSA schéme kratší ako podpis v ElGamalovej alebo RSA schéme, pri zachovaní rovnakej miery bezpečnosti.

Hodnotu súkromného kľúča x vieme vypočítať zo vzťahu (11.3), ak poznáme dokument m, digitálny podpis $\langle r, s\rangle$ a parameter k. Preto je potrebné parameter k po použití vymazať, v prípade predvýpočtu r dbať na zabezpečenie dôvernosti v rovnakej miere ako pri súkromnom kľúči x.

Možnosť počítať diskrétny logaritmus pri základe g vedie k výpočtu x z verejného kľúča y a teda k rozbitiu schémy.

V DSA vieme, podobne ako v RSA schéme a ElGamalovej schéme, falšovať náhodnú správu, ak nepoužijeme hašovaciu funkciu H, alebo ak H nie je jednosmerná.

11.4 Slepé podpisy

Pri dosahovaní rôznych bezpečnostných cieľov môžeme požadovať konštrukcie digitálnych podpisov s dodatočnými vlastnosťami. Existujú špeciálne schémy, v mnohých prípadoch v podobe protokolov, pre jednorazové podpisy (môžu byť použité na podpísanie len jednej správy), pre podpisy vyžadujúce pri overení aktívnu účasť podpisujúceho, pre podpisy v zastúpení („proxy“ podpisy), atď. V tejto časti popíšeme jednu realizáciu tzv. slepých podpisov.

Schéma slepých podpisov je protokol, kde podpisujúci nevie, čo podpisuje, ale druhý účastník získa korektný podpis zvoleného dokumentu. Uplatnenie takejto schémy je v protokoloch pre digitálne peniaze alebo pri realizácii elektronického notára.

Slepé podpisy sa dajú „postaviť“ na RSA schéme. Nech S je podpisujúca strana, nech e, n je verejný kľúč a d súkromný kľúč S. Označme symbolom A účastníka, ktorý chce získať podpis dokumentu m. Zároveň zápisom $X \to Y$: z označíme poslanie hodnoty z subjektom X subjektu Y. Postup podpisovania je takýto:

1. $A \to S$: $r = H(m) \cdot x^e \bmod n$, kde $x \in_R \mathbb{Z}_n^*$
2. $S \to A$: $s = r^d \bmod n$, teda S podpíše prijatú správu a podpis pošle A
3. A zo získaného podpisu vypočíta digitálny podpis m:

$$\begin{aligned} s \cdot x^{-1} \bmod n &= (r^d \bmod n) \cdot x^{-1} \bmod n \\ &= H(m)^d \cdot x^{ed} \cdot x^{-1} \bmod n \\ &= H(m)^d \bmod n. \end{aligned}$$

Keďže A zvolí x v prvom kroku náhodne, S nevie zistiť, akú hodnotu chce v skutočnosti A nechať podpísať.

12 Schémy na zdieľanie tajomstva

Schémy na zdieľanie tajomstva (secret sharing schemes) riešia nasledujúcu úlohu. Skupina n účastníkov schémy má zdieľať tajnú informáciu s, pričom rekonštruovať sú ju schopní len za určitých podmienok (napríklad keď kooperuje aspoň polovica účastníkov). To znamená, že každý účastník má svoju súkromnú informáciu, nazvime ju podiel, inak by pri rekonštrukcii nebol potrebný.

Schémy na zdieľanie tajomstva sú použiteľné v situáciach, keď je potrebné rozdeliť šifrovací alebo dešifrovací kľúč medzi viacerých účastníkov. Tieto schémy slúžia aj ako základ pre konštrukciu takých podpisových schém, v ktorých možno dokument podpísať len súčinnosťou definovanej časti účastníkov.

Podmienky rekonštrukcie tajnej zdieľanej informácie sa nazývajú prístupová štruktúra. Nech $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ je množina všetkých účastníkov schémy. Potom prístupovou štruktúrou nazveme ľubovoľnú podmnožinu potenčnej množiny $2^{\mathcal{P}}$, teda množiny obsahujúcej všetky podmnožiny $\mathcal{P}$. Schopnosť rekonštruovať tajnú zdieľanú informáciu s má mať iba taká množina účastníkov, ktorá patrí do prístupovej štruktúry.

Obvyklá požiadavka na prístupovú štruktúru je monotónnosť. To znamená, že ak má nejaká skupina účastníkov schopnosť rekonštruovať tajnú zdieľanú informáciu s, tak aj ľubovoľná nadmnožina tejto skupiny má schopnosť rekonštruovať s. Formálnejšie:

Definícia 11. Nech $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ a nech $\Gamma \subseteq 2^{\mathcal{P}}$ je prístupová štruktúra. Potom Γ je monotónna, ak $\forall A, B \subseteq \mathcal{P}$ platí:

$$A \in \Gamma \ \wedge \ A \subseteq B \quad \Longrightarrow \quad B \in \Gamma.$$

Najčastejšia prístupová štruktúra je tzv. (t, n)-prahová prístupová štruktúra, keď ľubovoľná množina účastníkov mohutnosti aspoň t $(1 \leq t \leq n)$ môže rekonštruovať s:

$$\Gamma = \{A \mid A \subseteq \mathcal{P}, \ |A| \geq t\}.$$

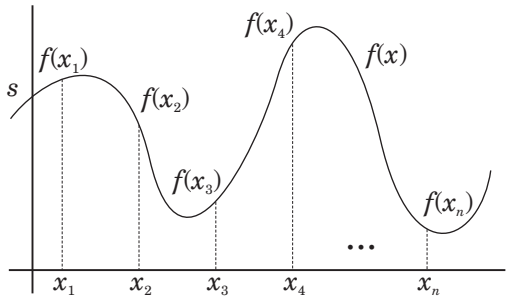
Schému s takouto prístupovou štruktúrou budeme nazývať (t, n) schéma.

12.1 Shamirova schéma

Shamirova schéma na zdieľanie tajomstva bola jednou z prvých schém s (t, n)-prahovou prístupovou štruktúrou. Hlavná myšlienka konštrukcie spočíva v tom, že polynóm stupňa $t - 1$ je jednoznačne určený t rôznymi bodmi.

Inicializácia. Cieľom inicializácie je vytvoriť podiely účastníkov. Predpokladáme, že túto fázu vykonáva dôveryhodná autorita. Máme daný počet účastníkov n a prahovú hodnotu t (prípadne aj tajnú informáciu s). Pri inicializácii schémy vytvoríme jednotlivé podiely účastníkov.

1. Zvolíme prvočíslo $p \geq n + 1$ a ak nemáme, tak aj tajnú zdieľanú informáciu $s \in \mathbb{Z}_p$.
2. Zvolíme n rôznych prvkov zo $\mathbb{Z}_p^*$, označme ich $x_1, x_2, \ldots, x_n$. Teda $x_i \neq x_j$, pre všetky $i \neq j$.
3. Zvolíme náhodný polynóm $f(x)$ stupňa najviac $t - 1$ nad $\mathbb{Z}_p$, pričom má platiť $f(0) = s$. Teda $f(x) = a_{t-1}x^{t-1} + \ldots + a_1 x + a_0$, kde $a_{t-1}, \ldots, a_1 \in_R \mathbb{Z}_p$ a $a_0 = s$.
4. Účastník P_i, pre $i = 1, \ldots, n$, dostane dvojicu $\langle x_i, f(x_i)\rangle$. Súkromná informácia účastníka P_i, teda podiel, je hodnota $f(x_i)$. Hodnoty $x_1, \ldots x_n$ sú obvykle verejne známe (niekedy dokonca pre všetky i položíme $x_i = i$), ale ak si to aplikácia vyžaduje, môžu byť súčasťou podielu účastníka.



Obrázok 12.1: Shamirova schéma

Po vygenerovaní schémy a rozdaní podielov sa polynóm vymaže, keďže obsahuje informáciu s.

Rekonštrukcia s. Ukážme, ako môže ľubovoľná množina účastníkov mohutnosti aspoň t rekonštruovať s. Stačí ukázať, že ľubovoľných práve t účastníkov vie vypočítať s. Ak by ich bolo viac ako t, tí navyše nemusia participovať. Bez ujmy na všeobecnosti predpokladajme, že na rekonštrukcii sa zúčastňujú účastníci $P_1, \ldots, P_t$. Spoločne majú k dispozícii nasledujúce vzťahy:

$$\begin{aligned} f(x_1) &= a_{t-1}x_1^{t-1} + \ldots + a_1x_1 + a_0 \\ f(x_2) &= a_{t-1}x_2^{t-1} + \ldots + a_1x_2 + a_0 \\ &\vdots \\ f(x_t) &= a_{t-1}x_t^{t-1} + \ldots + a_1x_t + a_0 \end{aligned}$$

Táto sústava t lineárnych rovníc o t neznámych má práve jedno riešenie, pretože determinant jej hlavnej matice nie je 0 (je to Vandermondov determinant):

$$\det \begin{pmatrix} x_1^{t-1} & \ldots & x_1 & 1 \\ x_2^{t-1} & \ldots & x_2 & 1 \\ \vdots & \ddots & \vdots & \vdots \\ x_t^{t-1} & \ldots & x_t & 1 \end{pmatrix} = \prod_{1 \le i < j \le t} (x_j - x_i) \neq 0.$$

Posledná nerovnosť vyplýva z toho, že $x_1, \ldots, x_n$ sú navzájom rôzne. Keďže $s = a_0$, po vyriešení sústavy zistíme tajnú zdieľanú informáciu.

Iný spôsob výpočtu s spočíva v použití Lagrangeovho interpolačného polynómu:

$$f(x) = \sum_{i=1}^{t} \left(f(x_i) \cdot \prod_{\substack{j=1 \\ j \neq i}}^{t} \frac{x - x_j}{x_i - x_j} \right),$$

kde stačí len dosadiť a vypočítať $f(0) = s$.

Príklad. Skonštruujme $(3, 5)$ schému pre tajnú zdieľanú informáciu $s = 7$. Nech $p = 11$. Zvoľme $x_1 = 1$, $x_2 = 2$, $x_3 = 3$, $x_4 = 4$, $x_5 = 5$. Zvoľme (náhodne) polynóm

nad $\mathbb{Z}_{11}$ stupňa najviac 2 a s absolútnym členom rovným s: $f(x) = 2x^2 + 8x + 7$. Podiely účastníkov sú potom (spolu s hodnotami x_i):

P_1	P_2	P_3	P_4	P_5
$\langle 1, 6\rangle$	$\langle 2, 9\rangle$	$\langle 3, 5\rangle$	$\langle 4, 5\rangle$	$\langle 5, 9\rangle$

Ak chcú P_2, P_3, P_4 rekonštruovať s, použitím Lagrangeovho interpolačného polynómu dostanú:

$$\begin{aligned} f(0) &= 9 \cdot \frac{(0-3)(0-4)}{(2-3)(2-4)} + 5 \cdot \frac{(0-2)(0-4)}{(3-2)(3-4)} + 5 \cdot \frac{(0-2)(0-3)}{(4-2)(4-3)} \\ &= 9 \cdot \frac{1}{2} + 5 \cdot \frac{8}{-1} + 5 \cdot \frac{6}{2} = 9 \cdot 6 + 5 \cdot 3 + 5 \cdot 3 = 7. \end{aligned}$$

Bezpečnosť schémy

Predpokladajme, že $t-1$ účastníkov chce vypočítať s. Ak by ich bolo menej, určite nemôžu zistiť o hodnote s viac. Bez ujmy na všeobecnosti môžeme prepokladať, že sú to $P_1, \ldots, P_{t-1}$. Títo účastníci testujú hypotézu, či je tajnou zdieľanou informáciou hodnota $s' \in \mathbb{Z}_p$. Skombinujúc svoje podiely dostanú sústavu $t-1$ lineárnych rovníc o $t-1$ neznámych:

$$\begin{aligned} f(x_1) - s' &= a_{t-1}x_1^{t-1} + \ldots + a_1 x_1 \\ f(x_2) - s' &= a_{t-1}x_2^{t-1} + \ldots + a_1 x_2 \\ &\vdots \\ f(x_{t-1}) - s' &= a_{t-1}x_{t-1}^{t-1} + \ldots + a_1 x_{t-1} \end{aligned}$$

Determinant matice tejto sústavy je nenulový:

$$\prod_{j=1}^{t-1} x_i^{-1} \cdot \prod_{1 \le i < j < t} (x_j - x_i) \neq 0,$$

teda sústava má práve jedno riešenie. Pre každé s' existuje f' také, že $f'(x_i) = f(x_i)$, pre $i = 1, \ldots t-1$. Preto $P_1, \ldots, P_{t-1}$ nemajú ako odlíšiť s' od s. To znamená, že ľubovoľná skupina $t-1$ účastníkov v Shamirovej (t, n) schéme nevie vypočítať o tajnej zdieľanej informácii s nič a sú v rovnakej situácii ako ktokoľvek, kto nemá ani jeden súkromný podiel.

Ľahko vidieť, že ak sa pri rekonštrukcii zverejnia podiely účastníkov alebo tajná zdieľaná informácia, je schéma na zdieľanie tajomstva jednorazová.

Nečestný účastník

Nech sa niektorý účastník (napríklad P_i) rozhodne sabotovať rekonštrukciu, teda namiesto svojho podielu $f(x_i)$ poskytne pri rekonštrukcii hodnotu $y \neq f(x_i)$. Predpokladajme, že výpočtu s sa zúčastňujú $P_1, \ldots, P_t$ a nečestný účastník je P_t. Táto

skupina vypočíta ako zdieľanú informáciu s', pričom platí:

$$\begin{aligned} s' &= s - f(x_t) \cdot \prod_{j=1}^{t-1} \frac{-x_j}{x_t - x_j} + y \cdot \prod_{j=1}^{t-1} \frac{-x_j}{x_t - x_j} \\ &= s + (y - f(x_t)) \cdot \prod_{j=1}^{t-1} \frac{-x_j}{x_t - x_j}. \end{aligned}$$

Ak sú hodnoty $x_1, \ldots, x_n$ verejne známe, alebo sa ich účastníci dozvedia pri rekonštrukcii, tak P_t vie po získaní s' určiť hodnotu s.

Detekciu podvádzania môžeme riešiť napríklad obmedzením povolených hodnôt pre tajnú zdieľanú informáciu s. Nech S je horná hranica hodnoty s, teda $s \in \mathbb{Z}_S$. Zvolíme prvočíslo p tak aby platilo:

$$p > \max\{S\varepsilon^{-1}, n\},$$

kde $\varepsilon \in (0, 1)$ je ľubovoľné (zvolené) číslo označujúce pravdepodobnosť neodhaleného podvodu. Čím nižšia pravdepodobnosť, tým viac prvkov obsahuje pole, v ktorom schému vytvárame. Predpokladajme, že $t-1$ účastníkov (nech sú to $P_1, \ldots, P_{t-1}$) sa rozhodne oklamať účastníka (nech je to P_t), ktorý s nimi rekonštruuje s. Teda $P_1, \ldots, P_{t-1}$ si zvolia hodnoty $y_1, \ldots, y_{t-1}$ namiesto svojich skutočných podielov. Nevedia však hodnotu podielu P_t: $f(x_t)$. Cieľom podvádzajúcich účastníkov je zvoliť $y_1, \ldots, y_{t-1}$ tak, aby sa pri rekonštrukcii spolu s $f(x_t)$ dospelo k zdieľanej hodnote $s' \in \mathbb{Z}_S$.

Ľahko vidieť, že ľubovoľnou voľbou $y_1, \ldots, y_{t-1}$ a s' je jednoznačne určený polynóm stupňa najviac $t-1$ (označme ho f'). Aby bol výsledok rekonštrukcie s', musí pre podiel účastníka P_t platiť $f(x_t) = f'(x_t)$. To znamená, že pre ľubovoľnú voľbu $y_1, \ldots, y_{t-1}$ je pravdepodobnosť rekonštrukcie nejakého $s' \in S$ rovná S/p. Teda pravdepodobnosť úspešného oklamania P_t je $S/p < \varepsilon$. Odolnosť schémy pre menší počet (ako $t-1$) podvádzajúcich účastníkov je priamym dôsledkom tejto vlastnosti – s rovnakou pravdepodobnosťou úspechu.

Iný spôsob detekcie využíva digitálne podpisy. Dôveryhodná autorita, ktorá generuje schému, digitálne podpíše jednotlivé podiely účastníkov. Pri rekonštrukcii je potom jednoduché overiť pravosť podielov. Takéto riešenie vyžaduje, na rozdiel od predchádzajúceho, zavedenie ďalšej kryptografickej konštrukcie. Navyše, bezpečnosť predchádzajúceho riešenia bola nepodmienená, teda nezávisela na žiadnych nedokázaných predpokladoch, aké by si vyžadovalo použitie digitálnych podpisov (o zložitosti faktorizácie, výpočte diskrétneho logaritmu a podobne). Na druhej strane, použitie digitálnych podpisov umožňuje identifikovať podvádzajúceho účastníka.

12.2 Asmuthova-Bloomova schéma

Asmuthova-Bloomova schéma je ďalším príkladom (t, n) schémy. Schéma využíva vlastnosti sústavy lineárnych kongruencií a na rekonštrukciu tajnej zdieľanej informácie Čínsku zvyškovú vetu.

Inicializácia. Vytvorenie podielov účastníkov (t, n) schémy musí vykonať dôveryhodná autorita.

1. Zvolíme navzájom rôzne a nesúdeliteľné prirodzené čísla $m_1 < m_2 < \ldots < m_n$ (väčšie ako 1) také, že:

$$\underbrace{m_1 m_2 \ldots m_t}_{\text{najmenších } t \text{ hodnôt}} > \underbrace{m_n m_{n-1} \ldots m_{n-t+2}}_{\text{najväčších } t-1 \text{ hodnôt}}.$$

   Tieto hodnoty môžu byť verejne známe. Označme súčin t najmenších modulov M, teda $M = m_1 m_2 \ldots m_t$.
2. Tajnú zdieľanú informáciu s zvolíme z množiny $\mathbb{Z}_M$.
3. Podiel účastníka P_i, pre $i = 1, \ldots, n$, je hodnota $k_i = s \bmod m_i$.

Rekonštrukcia s. Predpokladajme, že skupina t účastníkov $P_{i_1}, \ldots, P_{i_t}$ chce rekonštruovať s. Použijúc svoje podiely dostanú sústavu lineárnych kongruencií pre neznámu hodnotu s':

$$\begin{aligned} s' &\equiv k_{i_1} \pmod{m_{i_1}} \\ s' &\equiv k_{i_2} \pmod{m_{i_2}} \\ &\vdots \\ s' &\equiv k_{i_t} \pmod{m_{i_t}} \end{aligned}$$

Keďže všetky m_i sú navzájom nesúdeliteľné, účastníci získajú riešenie pomocou Čínskej zvyškovej vety:

$$s' = \sum_{j=1}^{t} k_{i_j} M_{i_j} N_{i_j} \bmod M',$$

kde $M' = m_{i_1} m_{i_2} \ldots m_{i_t}$, $M_{i_j} = (\prod_{k=1}^{t} m_{i_k})/m_{i_j}$ a $N_{i_j} = M_{i_j}^{-1} \bmod m_{i_j}$. Podľa Čínskej zvyškovej vety má sústava práve jedno riešenie v množine $\mathbb{Z}_{M'}$. Toto riešenie musí byť rovné s, lebo $M' \geq M > s$.

Príklad. Skonštruujme $(3, 5)$ schému. Zvoľme tieto hodnoty modulov: $m_1 = 79$, $m_2 = 81$, $m_3 = 83$, $m_4 = 85$, $m_5 = 89$. Teda tajnú zdieľanú informáciu s vyberáme z množiny $\mathbb{Z}_{m_1 m_2 m_3} = \mathbb{Z}_{531117}$. Nech $s = 250000$. Potom podiely účastníkov, spolu s hodnotami modulov, sú takéto:

P_1	P_2	P_3	P_4	P_5
$\langle 79, 44 \rangle$	$\langle 81, 34 \rangle$	$\langle 83, 4 \rangle$	$\langle 85, 15 \rangle$	$\langle 89, 88 \rangle$

Bezpečnosť schémy

Predpokladajme, že $t - 1$ účastníkov chce vypočítať s. Skupina s menším počtom účastníkov určite o tajnej zdieľanej informácii nemôže zistiť viac. Účastníci $P_{i_1}, \ldots, P_{i_{t-1}}$ majú k dispozícii $t - 1$ kongruencií:

$$\begin{aligned} s' &\equiv k_{i_1} \pmod{m_{i_1}} \\ &\vdots \\ s' &\equiv k_{i_{t-1}} \pmod{m_{i_{t-1}}} \end{aligned}$$

Tajná zdieľaná informácia s je tiež riešením sústavy. Po riešení pomocou Čínskej zvyškovej vety teda účastníci získajú s', pre ktoré platí:

$$s \equiv s' \pmod{m_{i_1} m_{i_2} \dots m_{i_{t-1}}}.$$

Skupina účastníkov vie redukovať počet potenciálnych hodnôt s. Všetci kandidáti na hodnotu s sú v tvare $s' + kM'$, kde $M' = m_{i_1} m_{i_2} \dots m_{i_{t-1}}$ a $k \in \mathbb{N}$. Keďže

$$M' \leq m_n m_{n-1} \dots m_{n-t+2} < M,$$

počet kandidátov na hodnotu s je $\lfloor M/M' \rfloor$. Najhorší prípad z hľadiska bezpečnosti nastane vtedy, keď je kandidátov najmenej, teda v skupine účastníkov P_{n-t+2}, ..., P_n. Počet kandidátov je vtedy

$$\left\lfloor \frac{M}{m_{n-t+2} \dots m_{n-1} m_n} \right\rfloor.$$

Táto analýza poukazuje na potrebu voliť moduly $m_1, \dots, m_n$ tak, aby bol rozdiel súčinu t najmenších členov a $t-1$ najväčších členov čo najväčší.

Poznámka. V predchádzajúcom príklade $(3, 5)$ schémy je počet kandidátov na hodnotu s v najhoršom prípade $\lfloor (79 \cdot 81 \cdot 83)/(85 \cdot 89) \rfloor = 70$.

Ak sa na rekonštrukcii zúčastní nečestný účastník a poskytne nekorektnú hodnotu svojho podielu, skupina rekonštruuje nesprávnu hodnotu. Navyše, podobne ako v Shamirovej schéme, tento nečestný účastník dokáže získať správnu hodnotu tajnej zdieľanej informácie s. Tento fakt vyplýva z toho, že v oboch schémach je rekonštrukcia tajnej zdieľanej informácie lineárnou kombináciou podielov účastníkov.

12.3 Hierarchické schémy

Bezpečnosť schémy na zdieľanie tajomstva závisí na možnosti neoprávnenej skupiny účastníkov (nepatriacej do prístupovej štruktúry) vypočítať niečo o tajnej zdieľanej informácii s. V ideálnom prípade nie sú schopní o hodnote s zistiť nič. Takú schému nazývame perfektná.

Definícia 12. Nech $\Gamma \subseteq 2^{\mathcal{P}}$ je prístupová štruktúra a nech K je množina, z ktorej sa vyberá s. Schéma na zdieľanie tajomstva realizujúca prístupovú štruktúru Γ sa nazýva perfektná, ak:

1. $\forall A \in \Gamma$: skupina vie zo svojich podielov rekonštruovať s;
2. $\forall A \notin \Gamma$: skupina nevie vypočítať o hodnote s nič.

Príklad. Nech $K = \{0,1\}^{10}$. Realizujme $(2, 2)$ schému tak, že pre $s \in K$ pridelíme účastníkovi P_1 prvých päť bitov s a P_2 druhú polovicu s. Táto schéma nie je perfektná, lebo samotný P_1 nepatrí do prístupovej štruktúry, ale vie zúžiť množinu potenciálnych hodnôt s na 32 prvkov.

V časti 12.1 sme ukázali, že Shamirova schéma pre (t, n)-prahovú prístupovú štruktúru je perfektná. Na druhej strane Asmuthova-Bloomova schéma perfektná nie je.

Navrhneme metódu konštrukcie perfektnej schémy na zdieľanie tajomstva pre ľubovoľnú monotónnu prístupovú štruktúru. Hlavná myšlienka je v priradení monotónnej booleovskej funkcie n premenných k monotónnej prístupovej štruktúre s n účastníkmi.

Pripomeňme, že n-árna booleovská funkcia $f : \{0,1\}^n \to \{0,1\}$ je monotónna, ak pre všetky $x, y \in \{0,1\}^n$ platí ($x = (x_1, \ldots, x_n)$, $y = (y_1, \ldots, y_n)$):

$$x \preceq y \quad \Longrightarrow \quad f(x) \leq f(y),$$

kde $x \preceq y$ práve vtedy, keď $x_1 \leq y_1, \ldots, x_n \leq y_n$.

Nech $\Gamma \in 2^{\mathcal{P}}$ je prístupová štruktúra pre účastníkov $\mathcal{P} = \{P_1, \ldots, P_n\}$. Potom k Γ priradíme monotónnu booleovskú funkciu takto:

$$f(x_1, \ldots, x_n) = 1 \quad \Longleftrightarrow \quad \{P_i \mid x_i = 1\} \in \Gamma.$$

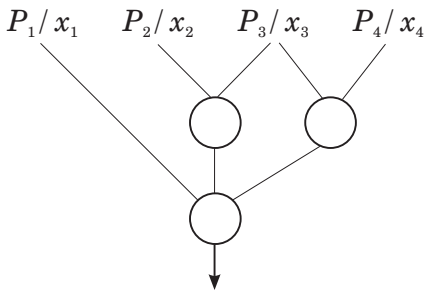
Toto priradenie je jednoznačné – prístupová štruktúra určuje funkciu a naopak. Monotónne booleovské funkcie môžeme vyjadriť ako formuly nad množinou spojok $\{\vee, \wedge\}$. Funkciou $f(x) = 0$ sa nezaoberáme, keďže zodpovedá prázdnej prístupovej štruktúre.

Príklad. Nech $n = 4$ a prístupová štruktúra je definovaná takto:

$$\Gamma = \{A \subseteq \mathcal{P} \mid P_1 \in A\} \cup \{A \subseteq \mathcal{P} \mid P_2, P_3 \in A\} \cup \{A \subseteq \mathcal{P} \mid P_3, P_4 \in A\}\}.$$

Teda prístup k zdieľanej informácii má každá skupina účastníkov, ktorá obsahuje P_1 alebo súčasne P_2 a P_3 alebo súčasne P_3 a P_4. Booleovská funkcia priradená ku Γ (pozri aj obrázok 12.2):

$$f(x_1, x_2, x_3, x_4) = x_1 \vee x_2 x_3 \vee x_3 x_4$$



Obrázok 12.2: Booleovská funkcia pre prístupovú štruktúru Γ

Pre každú monotónnu booleovskú funkciu vieme nakresliť obvod na jej vyhodnotenie tak, ako sme to urobili pre funkciu v príklade na obrázku 12.2. Pri konštrukcii schémy budeme postupovať od výstupu, kde má byť hodnota tajnej zdieľanej informácie s, k vstupom, teda súkromným podielom účastníkov. Pre každý typ „hradla“ v obvode vytvoríme samostatnú perfektnú schému na zdieľanie tajomstva (napríklad Shamirovu):

– hradlo $\wedge$ (obrázok 12.3a) – použijeme perfektnú (t, t) schému, kde t je počet vstupov hradla;

– hradlo $\vee$ (obrázok 12.3b) – použijeme perfektnú $(1, t)$ schému, kde t je počet vstupov hradla. Takáto schéma je len rozdaním zdieľanej informácie účastníkom.

(a) (b)

Obrázok 12.3: Hradlá $\wedge$ a $\vee$

Každý účastník dostane sadu toľkých podielov, koľko vstupov hradiel bezprostredne ovplyvňuje (inak povedané, koľko hrán v obvode od neho vychádza). V našom príklade dostane účastník P_3 dva podiely a ostatní účastníci jeden podiel. Indukciou vzhľadom na počet hrán možno ľahko dokázať, že takto skonštruovaná schéma je perfektná. Tým dostaneme konštruktívny dôkaz nasledujúcej vety.

Veta 6. *Pre každú monotónnu prístupovú štruktúru existuje perfektná schéma na zdieľanie tajomstva.*

Pri vyhodnocovaní schémy potrebujú účastníci poznať príslušný obvod a postupovať podľa neho. Schéma, ktorú týmto postupom získame, môže byť rozsiahla – účastníci pre niektoré prístupové štruktúry dostanú až exponenciálne veľa podielov (vzhľadom na počet účastníkov). Napríklad (t, n) schéma by sa realizovala obvodom s $\binom{n}{t} + 1$ hradlami. V konštrukcii schémy môžeme zaviesť nové booleovské spojky, pre ktoré vieme efektívne zostrojiť perfektnú schému a tým znížiť počet hradiel v obvode. Takou spojkou môže byť napríklad prahová spojka „aspoň t z n“. Napriek novým spojkám je vo všeobecnosti počet hradiel obvodu pre monotónnu prístupovú štruktúru až exponenciálne veľký.

Dôležitý parameter schémy na zdieľanie tajomstva je rozsah súkromných podielov, ktoré si účastníci musia pamätať. Tento parameter sa nazýva informačná miera.

Definícia 13. Nech $P_1, \ldots, P_n$ sú účastníci schémy na zdieľanie tajomstva. Nech K je množina, z ktorej sa vyberá tajná zdieľaná informácia a nech $S(P_i)$ (pre $i = 1, \ldots, n$) sú množiny potenciálnych podielov účastníka P_i. Potom informačná miera schémy pre účastníka P_i je definovaná takto:

$$\rho_i = \frac{\lg |K|}{\lg |S(P_i)|}.$$

Informačná miera schémy je minimum z informačných mier pre všetkých účastníkov:

$$\rho = \min\{\rho_i \mid 1 \leq i \leq n\}.$$

Poznámka. Predchádzajúca definícia predpokladá, že každý prvok z K aj z ľubovoľnej množiny $S(P_i)$ je rovnako pravdepodobný. Precíznejšia definícia používa entropie príslušných náhodných premenných.

Informačná miera vyjadruje podiel dĺžky zápisu (v bitoch) podielov účastníkov a zdieľanej informácie. V prípade že pre informačnú mieru schémy platí $\rho = 1$ hovoríme o *ideálnej* schéme. Ľahko vidieť, že Shamirova schéma je ideálna. Pre perfektné schémy vieme informačnú mieru odhadnúť zhora.

Veta 7. *Pre ľubovoľnú perfektnú schému na zdieľanie tajomstva platí* $\rho \leq 1$.

Dôkaz. Sporom. Nech $\exists i \in \{1, \ldots, n\} : \rho_i > 1$. Teda $|S(P_i)| < |K|$. Označme Γ prístupovú štruktúru schémy. Nech A je množina účastníkov, pre ktorú platí:

$$A \in \Gamma \quad \wedge \quad P_i \in A \quad \wedge \quad A \setminus \{P_i\} \notin \Gamma.$$

Ak by taká množina účastníkov neexistovala, účastník P_i je v schéme zbytočný. Skupina účastníkov $A \setminus \{P_i\}$ dokáže preskúšaním všetkých možností podielu P_i z $S(P_i)$ vylúčiť aspoň jednu hodnotu z K. To je v spore s tým, že schéma je perfektná. □

Existujú také prístupové štruktúry, pre ktoré sa dá dokázať, že ľubovoľná schéma s takouto prístupovou štruktúrou nemôže byť ideálna [Sti95, kapitola 11]. Príkladom takej prístupovej štruktúry je (pre $\mathcal{P} = \{P_1, P_2, P_3, P_4\}$):

$$\Gamma = \{A \in \mathcal{P} \mid P_1, P_2 \in A\} \cup \{A \in \mathcal{P} \mid P_2, P_3 \in A\} \cup \{A \in \mathcal{P} \mid P_3, P_4 \in A\}.$$

13 Kryptografické protokoly

Úlohou kryptografických protokolov je napĺňať komunikačné ciele účastníkov a zároveň zabezpečovať aj ich bezpečnostné potreby. Ciele kryptografických protokolov môžu byť rozmanité: manažment kľúčov, autentifikácia, digitálne peniaze, voľby v elektronickom prostredí, atď. Protokoly využívajú a vytvárajú rámec použitia základných kryptografických konštrukcií ako sú napríklad šifrovanie, hašovacie funkcie, digitálne podpisy, schémy na zdieľanie tajomstva.

Najdôležitejšou triedou kryptografických protokolov sú protokoly pre manažment kľúčov a autentifikáciu. Najmä týmto protokolom sa v nasledujúcich statiach budeme venovať.

Útoky na kryptografické protokoly môžeme rozdeliť na pasívne a aktívne. Pri pasívnych útokoch útočník komunikáciu len odpočúva, pri aktívnych môže urobiť čokoľvek – zadržať správu, vymeniť ju za inú, iniciovať paralelný beh protokolu a podobne. Obvykle predpokladáme, že útočník môže byť legitímnym účastníkom komunikácie.

Klasické typy útokov sú (podrobnejšie sa útokom na kryptografické protokoly venujeme v kapitole 14):

- Útok opakovaním (replay attack) – využitie starších správ v aktuálnom behu protokolu ich zopakovaním. Protiopatrenie spočíva v zaručení čerstvosti správ, či už použitím časových pečiatok alebo tzv. „príležitostných slov“ (nonces). Podrobnejšie sa im budeme venovať v ďalších častiach.
- Útočník uprostred (man in the middle) – útočník ako (nepozorovaný) prostredník v komunikácii medzi účastníkmi. Protiopatrením je napríklad zabezpečenie autentickosti správ prostredníctvom digitálnych podpisov alebo MAC.

- Využitie slabín použitých primitív – šifrovanie, digitálne podpisy a podobne. Pri návrhu kryptografických protokolov sa zvyčajne abstrahuje od konkrétnej voľby primitív a predpokladá sa, že sú zvolené dostatočne odolné algoritmy.

Častým cieľom protokolov je dohodnúť medzi účastníkmi kľúč, ktorým budú šifrovať svoju následnú komunikáciu. Tento kľúč sa nazýva kľúč pre spojenie alebo kľúč spojenia (session key). Význam kľúčov pre spojenie, oproti dlhodobým kľúčom, spočíva v nasledujúcom:

- obmedzenie rozsahu šifrových textov pre potenciálnu kryptoanalýzu,
- v prípade kompromitácie kľúča obmedzený rozsah kompromitovaných dát,
- nezávislosť (z pohľadu šifrovania) medzi aplikáciami a/alebo spojeniami,
- redukcia potreby uchovávať dlhodobo veľa rôznych kľúčov.

Účastníci protokolu si nemusia dôverovať, prípadne spolu ešte ani nekomunikovali. Dôveru v správy prenášané protokolom sprostredkúva tzv. dôveryhodná tretia strana (v niektorých protokoloch označovaná ako notár, certifikačná autorita a podobne).

Pri popise protokolov budeme skutočnosť, že účastník A poslal účastníkovi B správu m, zapisovať $A \to B$: m.

13.1 Diffieho-Hellmanov protokol

Cieľom Diffieho-Hellmanovho (skrátene DH) protokolu je dohodnúť medzi dvoma účastníkmi (označme ich A a B) kľúč K, ktorým môžu šifrovať následnú komunikáciu. Protokol predpokladá spoločné hodnoty p a g pre všetkých potenciálnych účastníkov protokolu. Hodnota p je dostatočne veľké prvočíslo a $g \in \mathbb{Z}_p^*$ môže ale nemusí byť generátor grupy $(\mathbb{Z}_p^*, \cdot)$.

1. $A \to B$: X, kde $X = g^x \bmod p$ a $x \in_R \mathbb{Z}_p^*$ zvolí účastník A náhodne
2. $B \to A$: Y, kde $Y = g^y \bmod p$ a $y \in_R \mathbb{Z}_p^*$ zvolí účastník B náhodne
3. A vypočíta $K = Y^x \bmod p$
4. B vypočíta $K = X^y \bmod p$

Ľahko možno ukázať, že A aj B vypočítajú rovnaký kľúč:

$$Y^x \bmod p = g^{xy} \bmod p = X^y \bmod p.$$

Pokiaľ uvažujeme o pasívnych útokoch na DH protokol, útočník má k dispozícii správy, ktoré si účastníci v protokole poslali: X a Y. Predpokladáme, že zároveň pozná verejne známe hodnoty p a g. Cieľom útočníka je vypočítať hodnotu K. Tento problém, nazývaný Diffieho-Hellmanov problém, možno sformulovať o čosi všeobecnejšie takto:

> Nech $(G, \cdot)$ je konečná cyklická grupa rádu $n \in \mathbb{N}$ a nech $g \in G$ je jej generátor. Pre dané $X, Y \in G$ je potrebné vypočítať $K \in G$ také, že $K = g^{xy}$, kde $X = g^x$ a $Y = g^y$, pre nejaké $x, y \in \mathbb{Z}_n$.

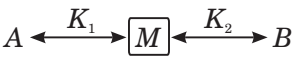
Prirodzene, ak vie útočník vypočítať diskrétny logaritmus X alebo Y pri základe g, teda hodnotu x alebo y, vie vypočítať aj K rovnako ako A alebo B. To znamená, že schopnosť riešiť problém diskrétneho logaritmu umožňuje riešiť DH problém. Opačná implikácia je vo všeobecnosti otvorený problém.

Poznámka. V časti 9.1 sme dokázali, že bezpečnosť ElGamalovho asymetrického šifrovacieho systému je ekvivalentná s Diffieho-Hellmanovým problémom.

V prípade aktívnych útokov možno DH protokol úspešne napadnúť s útočníkom uprostred, pozri obrázok 13.1. Označme útočníka M. Ten zachytí prvú správu a namiesto X pošle B hodnotu $W = g^w \bmod p$, kde w zvolí náhodne zo $\mathbb{Z}_p^*$. Podobne zachytí správu Y a namiesto nej pošle účastníkovi A hodnotu $Q = g^q \bmod p$, kde $q \in_R \mathbb{Z}_p^*$. Priebeh útoku je teda takýto:

1. $A \to M(B)$: $X = g^x \bmod p$
2. $M(A) \to B$: $W = g^w \bmod p$
3. $B \to M(A)$: $Y = g^y \bmod p$
4. $M(B) \to A$: $Q = g^q \bmod p$
5. A vypočíta $K_1 = Q^x \bmod p$
6. B vypočíta $K_2 = W^y \bmod p$

Pri zápise útoku sme použili označenie, ktoré využijeme aj v ďalších kapitolách. Zápis $A \to M(B)$: m znamená, že správu m posiela A účastníkovi B, ale zachytí ju M. Označenie $M(A) \to B$: m znamená, že M poslal namiesto (v mene) účastníka A správu m účastníkovi B.



Obrázok 13.1: Útočník uprostred komunikácie

Dôležité pre útočníka je to, že A a B nemajú ako odhaliť jeho prítomnosť v protokole a oba kľúče K_1 aj K_2 dokáže vypočítať:

$$K_1 = X^q \bmod p = g^{xq} \bmod p$$
$$K_2 = Y^w \bmod p = g^{yw} \bmod p$$

Následne útočník sprostredkúva komunikáciu medzi A a B, ku ktorej má neobmedzený prístup. Smerom k A sa správy šifrujú pomocou kľúča K_1, smerom k B pomocou kľúča K_2.

Certifikáty a autentifikované správy

Jeden zo spôsobov ako zamedziť útoku s útočníkom uprostred, je použitie certifikátov. Certifikáty slúžia na previazanie identity účastníka protokolu (alebo akéhokoľvek subjektu) s jeho verejným kľúčom. Toto previazanie zabezpečuje dôveryhodná tretia

strana – certifikačná autorita. Certifikát účastníka U, označme ho $C(U)$, je dvojica dát $ID(U)$, y_U, pričom táto je digitálne podpísaná certifikačnou autoritou (CA). Teda:

$$C(U) = ID(U), y_U, \text{podpis}_{CA}(ID(U), y_U),$$

kde $ID(U)$ je identifikácia subjektu a certifikátu (meno, adresa, platnosť certifikátu, identifikácia CA a podobne) a y_U je verejný kľúč U. Predpokladá sa, že každý účastník protokolu má k dispozícii verejný kľúč certifikačnej autority a môže overovať ňou podpísané certifikáty.

Poznamenajme, že našim cieľom nie je v tejto časti rozoberať problematiku certifikačných autorít, certifikátov a ich správy, ale načrtnúť hlavnú myšlienku ich významu pre zabezpečenie autentickosti prenášaných správ. Konkrétne detaily súvisiace s certifikátmi a certifikačnými službami možno nájsť v príslušných štandardoch.

Pre naše potreby v DH protokole ďalej predpokladajme, že každý účastník U má verejný kľúč zvolený v tvare $y_U = g^{x_U} \bmod p$, kde $x_U \in \mathbb{Z}_p^*$. Potom by DH protokol mohol prebehnúť len výmenou certifikátov, ich overením a následným výpočtom kľúča K:

$$\begin{aligned} A: &\quad K = y_B^{x_A} \bmod p = g^{x_A x_B} \bmod p \\ B: &\quad K = y_A^{x_B} \bmod p = g^{x_A x_B} \bmod p \end{aligned}$$

Útočník uprostred by neuspel, keďže by nebol schopný vyrobiť korektné certifikáty pre svoje „vymyslené“ verejné kľúče tak, aby boli prepojené s identitami účastníkov A a B. Nevýhodou takejto úpravy je fakt, že kľúč K je pre dvojicu účastníkov stále rovnaký (až do zmeny niektorého z certifikátov). Flexibilnejšiu zmenu kľúča pre spojenie poskytuje tzv. *Station to Station* protokol (funkcia E v protokole označuje symetrický šifrovací algoritmus):

1. $A \to B$: $X = g^x \bmod p$, kde $x \in_R \mathbb{Z}_p^*$
2. B zvolí náhodné $y \in_R \mathbb{Z}_p^*$ a vypočíta hodnotu $Y = g^y \bmod p$ a kľúč spojenia $K = X^y \bmod p$. Dvojicu (X, Y) podpíše použijúc svoj súkromný kľúč. Podpis zašifruje s použitím kľúča K a pošle A správu:
   $B \to A$: $Y, E_K(\text{podpis}_B(X, Y)), C(B)$
3. A vypočíta kľúč $K = Y^x \bmod p$ a dešifruje digitálny podpis. Potom A overí certifikát $C(B)$, vyberie z neho verejný kľúč B a overí digitálny podpis. Ak súhlasí, kľúč K je v poriadku a ako potvrdenie môže účastníkovi B poslať správu:
   $A \to B$: $E_K(\text{podpis}_A(X, Y)), C(A)$
4. B overí certifikát, dešifruje a následne overí podpis.

Útočník uprostred neuspeje, lebo nie je schopný falšovať digitálne podpisy v protokole.

13.2 Interlock protokol

Cieľom Interlock protokolu je detekcia útočníka uprostred komunikačného kanála. Predpokladajme, že účastníci A a B šifrujú svoju komunikáciu s použitím kľúča

K, dohodnutého DH protokolom. To znamená, že útočník mohol „sprostredkovať“ pre A kľúč K_1 a pre B kľúč K_2, pričom účastníci A a B nevedia o jeho existencii. Pripomeňme, že bez schopnosti riešiť Diffieho-Hellmanov problém nedokáže útočník zabezpečiť rovnosť kľúčov K_1 a K_2.

Predpokladajme, že účastníci A a B majú pripravené (resp. zvolia) správy m_A a m_B. Interlock protokol pozostáva z týchto krokov:

1. A chce poslať účastníkovi B správu m_A. Zašifruje ju použitím svojho kľúča pre spojenie a získa šifrový text c_A. Ten rozdelí na dve časti c_{A1}, c_{A2} tak, aby žiadnu časť m_A nebolo možné, ani pri znalosti šifrovacieho kľúča, určiť z jednej z nich. Šifrový text možno rozdeliť napríklad na párne a nepárne bity, alebo prvú a druhú polovicu.
   $A \to B$: c_{A1}
2. B zašifruje svoju správu m_B použitím svojho kľúča pre spojenie. Šifrový text c_B rozdelí (podobne ako A) na dve časti c_{B1} a c_{B2}.
   $B \to A$: c_{B1}
3. Po získaní časti c_{B1} účastník A odpovie:
   $A \to B$: c_{A2}
4. Po získaní časti c_{A2} účastník B rekonštruuje šifrový text c_A a dešifruje správu m_A. Zároveň pošle:
   $B \to A$: c_{B2}
5. Účastník A zloží šifrový text c_B a dešifruje správu m_B.

Útočník uprostred musí zvoliť vlastné správy m'_A alebo m'_B, lebo prvá časť šifrového textu je bez druhej nepoužiteľná. Zároveň, keďže $K_1 \neq K_2$, nie je možné poslať časti správ nezmenené – po dešifrovaní iným kľúčom by nedávali zmysel. Úspech útočníka potom závisí na schopnosti „tváriť sa“ ako skutočný účastník A alebo B v komunikácii.

Poznámka. Útočníkovi stačí oklamať jednu stranu, pri Interlock protokole totiž môže najskôr prebehnúť celú komunikáciu napríklad s účastníkom A (s vymyslenou správou m'_B). Tým získa od A správu m_A, s ktorou opäť vykoná celý beh protokolu s účastníkom B.

Význam Interlock protokolu spočíva v tom, že útočník nemôže nečinne (pasívne) odpočúvať komunikáciu účastníkov, ale musí sa do nej aktívne zapojiť, čím zvyšuje pravdepodobnosť svojho odhalenia.

Rovnako ako v prípade DH protokolu sa dá Interlock protokol použiť aj v situácii, keď si účastníci A a B cez nezabezpečený kanál vymenia svoje verejné kľúče a následne používajú asymetrické (prípadne hybridné) šifrovanie na zabezpečenie dôvernosti svojej komunikácie. Útočník môže v prvom kroku zachytiť posielané verejné kľúče a nahradiť ich vlastnými, čím získa prístup ku komunikácii účastníkov.

13.3 Protokoly s dôveryhodnou treťou stranou

Mnohé protokoly predpokladajú aktívnu účasť dôveryhodnej tretej strany pri autentifikácii a distribúcii kľúča pre spojenie.

Konštrukcia rôznorodých protokolov riešiacich rovnaké (alebo veľmi podobné) úlohy vyplýva z rôznych podmienok v ktorých sa má konkrétny protokol použiť. Základné počiatočné podmienky (predpoklady) pri návrhu protokolov sú:

- použité kryptografické primitíva – symetrické, asymetrické šifrovanie, atď.
- predpoklady o znalosti a vlastníctve kľúčov – verejné kľúče, zdieľanie kľúčov, atď.
- použitie časových pečiatok alebo príležitostných slov na zabezpečenie čerstvosti správ;
- počet kôl protokolu, rozsah správ, nároky na komunikačnú a výpočtovú zložitosť;
- predpoklady o schopnostiach zúčastnených subjektov – napríklad generovanie kľúčov.

Časové pečiatky a príležitostné slová

Časové pečiatky a príležitostné slová sú prostriedkom na zamedzenie útoku opakovaním. Časové pečiatky sú prostým zápisom aktuálneho času. Pokiaľ príjemca v správe, ktorá šifrovaná s použitím kľúča známemu len jemu a odosielateľovi, obdrží časovú pečiatku s dostatočne aktuálnym časom, môže správu považovať za čerstvú. Časové pečiatky sú pomerne „luxusné“ objekty, ktoré pre správne spracovanie v protokole vyžadujú synchronizáciu vnútorných hodín účastníkov. Pri použití časových pečiatok je potrebné venovať pozornosť aj nastaveniu intervalu, v ktorom sa časový údaj ešte berie ako aktuálny. Príliš krátky interval zvyšuje nároky na synchronizáciu, pridlhý zas otvára väčšie možnosti pre potenciálny útok opakovaním.

Jednoduchším objektom na zaručenie čerstvosti správ sú príležitostné slová. Príležitostné slovo je dostatočne dlhý náhodný reťazec bitov. Účastník si tento reťazec vygeneruje (a pošle ostatným) počas behu protokolu. Neskôr, keď dostane šifrovanú správu obsahujúcu svoje príležitostné slovo, vie, že táto správa nie je opakovaním správy zo staršieho behu protokolu. Príležitostné slovo nemôže byť príliš krátke, vzhľadom na riziko opakovania reťazcov medzi behmi protokolu. Príliš dlhé slovo zasa zvyšuje komunikačnú aj výpočtovú zložitosť.

Príklady protokolov využívajúcich časové pečiatky a príležitostné slová sú v nasledujúcich statiach. V zápisoch protokolov budeme abstrahovať od použitia konkrétnych šifrovacích algoritmov a správu m šifrovanú s použitím kľúča K budeme zapisovať $\{m\}_K$.

Upravený Wide Mouthed Frog protokol

Cieľom upraveného Wide Mouthed Frog (WMF) protokolu je distribúcia kľúča pre spojenie K a jednostranná autentifikácia (dôkaz prítomnosti v protokole) účastníka A. Protokol predpokladá, že účastníci A a B zdieľajú s dôveryhodnou treťou stranou (serverom) T kľúče na šifrovanie vzájomnej komunikácie. Kľúč pre účastníka A označme K_A a kľúč pre B označme K_B. Upravený WMF protokol využíva časové pečiatky T_A, resp. T_T, v danej chvíli vytvorené účastníkom A, resp. serverom T.

1. $A \to T$: $A, \{T_A, B, K\}_{K_A}$

2. $T \rightarrow B$: $\{T_T, A, B, K\}_{K_B}$

Kľúč pre spojenie generuje účastník A a prostredníctvom T ho oznamuje účastníkovi B. V prvom kroku A posiela dôveryhodnému serveru T šifrovanú správu obsahujúcu aktuálnu časovú pečiatku T_A, vygenerovaný kľúč K a identifikátor účastníka, s ktorým chce komunikovať.

Server T správu dešifruje, skontroluje aktuálnosť časovej pečiatky a ak je v poriadku, oznámi účastníkovi B zistené skutočnosti. V druhom kroku pošle B správu šifrovanú s použitím kľúča K_B, obsahujúcu čerstvú časovú pečiatku T_T, identifikátory oboch účastníkov (budúcej) komunikácie a kľúč spojenia K.

Účastník B dešifruje správu, overí aktuálnosť časovej pečiatky a svoj identifikátor. Z dešifrovanej správy získa zároveň kľúč spojenia a zistí identitu účastníka, ktorý s ním chce komunikovať. O prítomnosti A v protokole je presvedčený prostredníctvom T, pretože:

- T overil korektnosť (a čerstvosť) správy v prvom kroku – inak by nebol vykonal druhý krok protokolu
- kľúč K_B pozná len T (a samozrejme B) a správa od neho je čerstvá

Účastník A bude o prítomnosti B v komunikácii presvedčený až vtedy, keď dostane správu šifrovanú kľúčom K. V prípade potreby možno upravený WMF protokol rozšíriť o tretí krok (opäť s časovou pečiatkou):

3. $B \rightarrow A$: $\{T_B, A, B\}_K$

Poznámka. Pôvodný Wide Mouthed Frog protokol má niektoré bezpečnostné nedostatky (slabiny), o ktorých podrobnejšie hovoríme v kapitole 14.

Yahalomov protokol

Yahalomov protokol predpokladá, že účastníci A a B zdieľajú s dôveryhodnou treťou stranou (serverom) T kľúče K_A a K_B na šifrovanie vzájomnej komunikácie. Nech N_A a N_B sú príležitostné slová vygenerované účastníkmi A a B. Cieľom protokolu je vzájomná autentifikácia účastníkov (dôkaz prítomnosti v protokole) a distribúcia kľúča pre spojenie generovaného T.

1. $A \rightarrow B$: A, N_A
2. $B \rightarrow T$: $B, \{A, N_A, N_B\}_{K_B}$
3. $T \rightarrow A$: $\{B, K, N_A, N_B\}_{K_A}, \{A, K\}_{K_B}$
4. $A \rightarrow B$: $\{A, K\}_{K_B}, \{N_B\}_K$

Na začiatku chce A komunikovať s B. Pošle mu svoj identifikátor a príležitostné slovo. Účastník B pripraví správu (žiadosť o kľúč pre spojenie) pre T. Vlastné príležitostné slovo N_B je spolu s N_A a identitou A šifrované s použitím kľúča K_B, ktorý B zdieľa s T.

Dôveryhodný server T správu dešifruje a pripraví pre A odpoveď zloženú z dvoch častí. Jedna je šifrovaná kľúčom K_A a je teda určená pre účastníka A. Druhá, šifrovaná kľúčom K_B je určená účastníkovi B. Obe správy obsahujú vygenerovaný kľúč pre spojenie K. Správa pre A navyše obsahuje príležitostné slovo N_A, čím presvedča

A o svojej čerstvosti (lebo len A a T poznajú kľúč K_A). Takže A po prijatí správy v treťom kroku dešifruje svoju časť, skontroluje prítomnosť N_A, identifikátora B a pripraví správu pre B.

Prvá časť správy vo štvrtom kroku je zopakovaním časti odpovede T určenej B. Druhú časť vytvorí účastník A a je to kľúčom pre spojenie šifrované príležitostné slovo N_B. Účastník B dešifruje prvú časť správy a overí identifikátor A. Získaný kľúč použije na dešifrovanie druhej časti a následne skontroluje prítomnosť N_B. Keďže príležitostné slovo N_B je v protokole posielané výlučne šifrované, môžu ho poznať (okrem B) len T a A. Jeho prítomnosť v kontexte štvrtého kroku protokolu preto dokazuje, že A verí v čerstvosť kľúča K. To, spolu s faktom, že prvá časť správy je od T (lebo len T pozná K_B), presviedča B o vhodnosti použitia K ako kľúča pre spojenie.

Účastník A je o prítomnosti B presvedčený sprostredkovane cez dôveryhodný server T, po treťom kroku protokolu. Účastník B je o prítomnosti A v protokole presvedčený po úspešnom štvrtom kroku.

Poznámka. Niektoré varianty Yahalomovho protokolu obsahujú bezpečnostné slabiny využiteľné útočníkom [Syv94].

14 Útoky na kryptografické protokoly

Predpoklady aj ciele protokolov na autentifikáciu a manažment kľúčov možno ľahko sformulovať. Protokoly obvykle pozostávajú z výmeny niekoľkých, nie veľmi komplikovaných správ. Napriek tomu nie je jednoduché navrhnúť bezpečný protokol napĺňajúci zvolené ciele. Táto časť je venovaná rôznym triedam útokov na kryptografické protokoly. Niektoré útoky kombinujú viacero techník, preto sú uvedené triedy skôr predstavením rizík v návrhoch protokolov ako presnou kategorizáciou útokov. Jednotlivé útoky ilustrujeme na konkrétnych príkladoch.

Útočníka budeme v protokoloch označovať ako E. Pripomeňme, že zápisom $E(A)$ označíme útočníka vystupujúceho v roli účastníka A – pri posielaní alebo pri zachytení správy.

14.1 Útok opakovaním

Útok spočíva v zopakovaní správy alebo jej časti v aktuálnom behu protokolu. Zopakovaná správa môže pochádzať zo súčasného behu protokolu, alebo môže byť staršia. Ilustrujme tento útok na dvoch protokoloch.

Needhamov-Schroederov protokol

Needhamov-Schreoderov protokol (1978) je protokol s dvoma účastníkmi A, B a dôveryhodnou treťou stranou (označme ju S). Cieľom protokolu je vzájomná autentifikácia A a B, spolu s distribúciou kľúča pre spojenie K.

1. $A \rightarrow S$: A, B, N_A
2. $S \rightarrow A$: $\{N_A, B, K, \{K, A\}_{K_B}\}_{K_A}$

3. $A \rightarrow B$: $\{K, A\}_{K_B}$
4. $B \rightarrow A$: $\{N_B\}_K$
5. $A \rightarrow B$: $\{N_B - 1\}_K$

Kľúč K_A (resp. K_B) zdieľa A (resp. B) s S a slúži na šifrovanie vzájomnej komunikácie. Kľúč pre spojenie K generuje S a je určený na šifrovanie následnej komunikácie medzi A a B. Príležitostné slová N_A a N_B slúžia účastníkom A a B na overenie čerstvosti prijatej správy a sú vygenerované samotnými účastníkmi ako dostatočne dlhé náhodné reťazce bitov.

Slabinou Needhamovho-Schroederovho protokolu je nedostatočné zabezpečenie čerstvosti správy posielanej v treťom kroku protokolu. Predpokladajme, že útočník E odpočuje celý priebeh protokolu. Následne prebieha medzi účastníkmi A a B komunikácia šifrovaná s použitím kľúča K. Predpokladajme, že neskôr je K kompromitovaný – účastníci ho prezradia, prípadne E získa K kryptoanalýzou. Po kompromitácii môže útočník kľúč K opäť vnútiť účastníkovi B, predstierajúc identitu A, zopakovaním správy zo starej inštancie protokolu:

3'. $E(A) \rightarrow B$: $\{K, A\}_{K_B}$
4'. $B \rightarrow E(A)$: $\{N'_B\}_K$

Na správu v štvrtom kroku dokáže útočník správne odpovedať, keďže pozná K:

5'. $B \rightarrow E(A)$: $\{N'_B - 1\}_K$

Účastník B je (mal by byť) presvedčený o prítomnosti A v protokole a o vhodnosti kľúča K na komunikáciu s ním. Opak je však pravdou, komunikovať s ním bude útočník E.

Problém Needhamovho-Schroederovho protokolu je v tom, že ku kľúču K v tretej správe sa z pohľadu B neviaže nič čerstvé (príležitostné slovo alebo časová pečiatka). Preto nie je zaručená ani čerstvosť kľúča K. Protokol je možné upraviť viacerými spôsobmi tak, aby túto slabinu neobsahoval. Jedna z možných úprav je táto:

1. $A \rightarrow B$: A, B, N_A
2. $B \rightarrow S$: A, B, N_A, N_B
3. $S \rightarrow A$: $\{N_A, B, K, \{N_B, A, K\}_{K_B}\}_{K_A}$
4. $A \rightarrow B$: $\{N_B, A, K\}_{K_B}$

Účastník B svoje príležitostné slovo poskytne S ešte na začiatku protokolu. Neskôr, po obdržaní a dešifrovaní správy $\{N_B, A, K\}_{K_B}$ vie, vďaka prítomnosti N_B, že kľúč K je čerstvý.

Wide Mouthed Frog protokol

Upravená verzia Wide Mouthed Frog protokolu (WMF) bola popísaná v časti 13.3. Pôvodná verzia WMF prebieha takto:

1. $A \rightarrow T$: $A, \{T_A, B, K\}_{K_A}$
2. $T \rightarrow B$: $\{T_T, A, K\}_{K_B}$

Ciele protokolu sú distribúcia kľúča pre spojenie K medzi účastníkmi A a B prostredníctvom dôveryhodnej tretej strany T a jednostranná autentifikácia účastníka A. Kľúč K je generovaný účastníkom A. WMF na zaručenie čerstvosti prenášaných správ využíva časové pečiatky T_A (vytvára A) a T_T (vytvára T). Šifrovanie komunikácie účastníkov s T je zabezpečené s pomocou kľúčov K_A a K_B.

Predpokladajme, že A zaháji protokol s úmyslom dôverne komunikovať s B. Útočník E zachytí správu určenú B v druhom kroku (správu prepustí aj k B, takže pre účastníkov A a B sa nič nezmenilo a po dokončení protokolu používajú kľúč K):

1. $A \to T$: $A, \{T_A, B, K\}_{K_A}$
2. $T \to E(B)$: $\{T_T, A, K\}_{K_B}$
   $E(T) \to B$: $\{T_T, A, K\}_{K_B}$

Zachytená správa má rovnakú štruktúru ako správa v prvom kroku WMF a časová pečiatka v nej je dostatočne čerstvá. Útočník správu použije ako falošný prvý krok novej inštancie WMF a následne opäť zachytí odpoveď T:

1'. $E(B) \to T$: $B, \{T_T, A, K\}_{K_B}$

2'. $T \to E(A)$: $\{T'_T, B, K\}_{K_A}$

Zachytená správa má opäť vhodnú štruktúru a E iniciuje ďalšiu inštanciu WMF:

1". $E(A) \to T$: $A, \{T'_T, B, K\}_{K_A}$

2". $T \to E(B)$: $\{T''_T, A, K\}_{K_B}$

Takto neustále aktualizuje časovú pečiatku vnútri šifrovanej správy a medzitým pracuje na kompromitácii kľúča K. Po získaní K útočník využije správu z aktuálneho behu protokolu a dokáže vnútiť B opäť kľúč K ako vhodný kľúč na šifrovanie spojenia s účastníkom A:

$1^{(k)}$. $E(A) \to T$: $A, \{T_T^{(k-1)}, B, K\}_{K_A}$

$2^{(k)}$. $T \to B$: $\{T_T^{(k)}, A, K\}_{K_B}$

Ochrana pred útokom spočíva v porušení symetrie (rovnakej štruktúry) správ. Vtedy nie je možné použiť správu z druhého kroku WMF v prvom kroku. Príkladom je upravený WMF v časti 13.3.

Denningovej-Saccov protokol

Cieľom protokolu (1981) je jednostranná autentifikácia účastníka A na základe certifikátov poskytnutých dôveryhodnou treťou stranou T a ustanovenie kľúča pre spojenie K. Označme C_A a C_B certifikáty verejných kľúčov účastníkov A a B, teda (v podstate) dôveryhodným T digitálne podpísané verejné kľúče týchto účastníkov. Účastník A generuje kľúč K a časovú pečiatku T_A. Zápis $\{\{K, T_A\}_{K_A^{-1}}\}_{K_B}$ znamená, že správa K, T_A je digitálne podpísaná účastníkom A a následne šifrovaná pre B s použitím jeho verejného kľúča.

1. $A \to T$: A, B
2. $T \to A$: C_A, C_B

3. $A \rightarrow B$: $C_A, C_B, \{\{K, T_A\}_{K_A^{-1}}\}_{K_B}$

Útočník môže využiť situáciu, keď s ním chce A komunikovať, na predstieranie falošnej identity pred účastníkom B [AN94]:

1. $A \rightarrow T$: A, E
2. $T \rightarrow A$: C_A, C_E
3. $A \rightarrow E$: $C_A, C_E, \{\{K, T_A\}_{K_A^{-1}}\}_{K_E}$

3'. $E(A) \rightarrow B$: $C_A, C_B, \{\{K, T_A\}_{K_A^{-1}}\}_{K_B}$

Po prijatí správy v treťom kroku protokolu útočník dešifruje správu, získa kľúč K, overí časovú pečiatku a digitálny podpis A. Podpísaný kľúč s časovou pečiatkou útočník zašifruje s použitím verejného kľúča B a okamžite pošle ako tretí krok protokolu. Časová pečiatka T_A je ešte stále čerstvá, preto účastník B považuje prijatú správu za korektný krok protokolu a akceptuje autentifikáciu A aj kľúč K. Útočník získa certifikát C_B, pokiaľ ho ešte nemá, komunikáciou s T.

Ochranu pred útokom zabezpečí napríklad pridanie identifikátorov účastníkov do podpísanej správy v treťom kroku protokolu:

3. $A \rightarrow B$: $C_A, C_B, \{\{A, B, K, T_A\}_{K_A^{-1}}\}_{K_B}$

14.2 Útočník uprostred

„Man in the middle“ útok je v širšom ponímaní každý útok, kde útočník aktívne vstúpi do komunikácie medzi účastníkmi protokolu. Teda väčšinu útokov prezentovaných v tejto kapitole (14) môžeme považovať za útok s útočníkom uprostred.

Iný, zúžený pohľad na útok s útočníkom uprostred je ten, že útočník správy posielané medzi účastníkmi protokolu zachytí, ale následne ich (možno pozmenené) pošle pôvodnému adresátovi. Prominentným príkladom útoku tohto typu je útok na Diffieho-Hellmanov protokol, popísaný v časti 13.1.

14.3 Nepresný popis protokolu

Nepresný popis protokolu môže viesť k implementácii s bezpečnostnými slabinami. Nepresnosť popisu sa môže týkať rôznych častí protokolu: dĺžky objektov (napr. kľúčov, príležitostných slov), požadovaných parametrov použitia šifrovacieho algoritmu (napr. mód, dĺžka bloku), postupnosti operácií jednotlivých účastníkov po každom kroku protokolu (najmä testov), atď. Táto trieda útokov sa do značnej miery prekrýva s útokmi súvisiacimi s implementačnými problémami, ktoré sú predmetom ďalšej časti.

Na ilustráciu bezpečnostných problémov pri nepresnom popise potrebných testov v protokole použijeme Otwayov-Reesov protokol. Neskôr na tomto protokole demonštrujeme aj útoky iných typov.

Otwayov-Reesov protokol

Cieľom protokolu (1987) je distribúcia kľúča pre spojenie K účastníkom A a B, spolu s autentifikáciou účastníka A (potvrdenie jeho prítomnosti v protokole). Autentifikácia B je dokončená až po prvom použití kľúča K. Kľúč pre spojenie generuje dôveryhodná tretia strana T. Každý účastník zdieľa s T kľúč na šifrovanie vzájomnej komunikácie. Pre účastníka A označíme tento kľúč K_A, pre B to bude K_B. Na zabezpečenie čerstvosti prenášaných správ sa používajú príležitostné slová N_A, resp. N_B, generované účastníkom A, resp. B. Protokol využíva na identifikáciu aktuálneho behu protokolu náhodný identifikátor M, s cieľom zabrániť opakovaniu správ zo starších inštancií protokolu. Tento identifikátor je volený účastníkom A.

1. $A \to B : M, A, B, \{N_A, M, A, B\}_{K_A}$
2. $B \to T : M, A, B, \{N_A, M, A, B\}_{K_A}, \{N_B, M, A, B\}_{K_B}$
3. $T \to B : M, \{N_A, K\}_{K_A}, \{N_B, K\}_{K_B}$
4. $B \to A : M, \{N_A, K\}_{K_A}$

Predpokladajme, že dôveryhodná tretia strana T v druhom kroku neskontroluje zhodu identifikátorov účastníkov (A, B) v otvorenom a šifrovom texte, ale len zhodu v šifrovom texte. Útočník E potom môže postupovať takto:

1. $A \to B : M, A, B, \{N_A, M, A, B\}_{K_A}$
2. $B \to E(T) : M, A, B, \{N_A, M, A, B\}_{K_A}, \{N_B, M, A, B\}_{K_B}$
   $E \to T : M, A, E, \{N_A, M, A, B\}_{K_A}, \{N_E, M, A, B\}_{K_E}$
3. $T \to E : M, \{N_A, K\}_{K_A}, \{N_E, K\}_{K_E}$
4. $E(B) \to A : M, \{N_A, K\}_{K_A}$

Útočník po zachytení správy v druhom kroku protokolu pošle T vlastnú správu, v ktorej vystupuje ako legitímny účastník protokolu. Odpoveď potom umožní E získať kľúč pre spojenie a navyše aj správu, ktorú potrebuje poslať A namiesto účastníka B.

14.4 Implementačné problémy

Popisy kryptografických protokolov abstrahujú od niektorých konkrétnych parametrov, ktoré je nevyhnutné pri implementácii zvoliť: šifrovací algoritmus a jeho mód, dĺžka jednotlivých objektov a pod. Vo väčšine kryptografických protokolov môže nevhodná voľba parametrov pri implementácii vytvoriť bezpečnostnú slabinu.

Variabilná alebo nevhodná dĺžka objektov

Opäť uvažujme Otwayov-Reesov protokol. Predpokladajme, že dĺžka generovaného kľúča pre spojenie K zodpovedá súčtu dĺžok M, A a B: $|K| = |M, A, B|$. Útočník zachytí prvú správu protokolu:

1'. $A \to E(B) : M, A, B, \{N_A, M, A, B\}_{K_A}$

Táto správa má rovnakú štruktúru ako odpoveď, ktorú má A dostať vo štvrtom kroku. Útočník teda zopakuje zachytený šifrový text ($K = M, A, B$):

4'. $E(B) \rightarrow A$: $M, \{N_A, K\}_{K_A}$

Účastník A získa kľúč pre spojenie ako predpisuje protokol. Útočník kľúč pozná a pokračuje v komunikácii s A, predstierajúc identitu B.

Nevhodný mód šifrovacieho algoritmu

Popisy protokolov abstrahujú od konkrétnych šifrovacích algoritmov a módov ich použitia. Z hľadiska bezpečnosti však nevhodná voľba algoritmu alebo módu môže znamenať problém. Napríklad ECB mód (Electronic Codebook, pozri časť 3.3), umožňuje preusporiadavať a vynechávať bloky šifrového textu bez ovplyvnenia zodpovedajúcich blokov otvoreného textu.

Predpokladajme, že v Needhamovom-Schroederovom protokole (časť 14.1) použijeme ECB mód blokovej šifry (na šifrovacom algoritme nezáleží). Druhá správa v protokole má tvar:

2. $S \rightarrow A : \{N_A, B, \underline{K}, \{K, A\}_{K_B}\}_{K_A}$

Ak podčiarknutý kľúč K v otvorenom texte zaberá celý blok (prípadne viac blokov) samostatne, môže útočník nahradiť tento blok zodpovedajúcim blokom zo staršieho behu protokolu, pre ktorý pozná zašifrovaný kľúč. Takýto blok vie dokonca získať priamo od S tým, že iniciuje samostatnú inštanciu protokolu:

1'. $E \rightarrow S$: E, A, N_E

2'. $S \rightarrow E$: $\{N_E, A, K', \{\underline{K'}, E\}_{K_A}\}_{K_E}$

Po dešifrovaní útočník získa nielen požadovaný blok z vnútornej správy, ale aj hodnotu kľúča K'.

Iné módy podobné presúvania blokov neumožňujú. Napriek tomu si použitie konkrétneho módu vyžaduje opatrnosť. Ilustrujme to na použití CBC módu (Cipher Block Chaining, časť 3.3), v Otwayovom-Reesovom protokole. Predpokladajme, že správa N_B, M, A, B, fragment z druhého kroku protokolu, sa rozdelí na bloky otvoreného textu takto:

$$\underbrace{N_B}_{P_1}, \underbrace{M}_{P_2}, \underbrace{A, B}_{P_3}.$$

Útočník E iniciuje protokol pre komunikáciu s B a zachytí druhú správu:

1. $E \rightarrow B : M', E, B, \{N_E, M', E, B\}_{K_E}$

2. $B \rightarrow E(T) : M', E, B, \{N_E, M', E, B\}_{K_E}, \underbrace{\{N'_B, M', E, B\}_{K_B}}_{(*)}$

Vzhľadom na uvedené rozdelenie druhého otvoreného textu na bloky, má šifrový text $(*)$ tvar $\{IV'\}_{K_B}, C'_1, C'_2, C'_3$, kde IV je inicializačný vektor použitý pre CBC mód. Útočník zároveň začne novú inštanciu protokolu s B, predstierajúc identitu A.

1'. $E(A) \rightarrow B : M, A, B, \{hocičo\}_{K_E}$

2'. $B \to E(T) : M, A, B, \{hocičo\}_{K_E}, \underbrace{\{N_B, M, A, B\}_{K_B}}_{(\triangle)}$

Samozrejme, text prvej správy by mal byť šifrovaný kľúčom K_A, avšak B ho aj tak nevie dešifrovať a v ďalšom priebehu útoku nehrá žiadnu rolu. Preto tento text môže byť hocičo, aj náhodný reťazec bitov (vhodnej dĺžky). Bloky šifrového textu ($\triangle$) označíme $\{IV\}_{K_B}, C_1, C_2, C_3$, kde IV je opäť inicializačný vektor (nemusí byť rovnaký ako v predchádzajúcom prípade). Útočník pošle namiesto zachytenej správy novú správu, pri konštrukcii ktorej využije časti správ z oboch inštancií protokolu:

2'. $E(B) \to T : S, E, B, \{N_E, S, E, B\}_{K_E}, X,$

kde $X = \{IV\}_{K_B}, C_1, C_2', C_3'$ a $S = C_1 \oplus M' \oplus C_1'$. Následne T dešifruje X takto:

$$\begin{aligned} D_{K_B}(X) &= IV \oplus D_{K_B}(C_1),\ C_1 \oplus D_{K_B}(C_2'),\ C_2' \oplus D_{K_B}(C_3') \\ &= N_B,\ \underbrace{C_1 \oplus M' \oplus C_1'}_{S},\ E,\ B. \end{aligned}$$

Z pohľadu T je správa presne taká, akú očakáva v druhom kroku Otwayovho-Reesovho protokolu. Vygeneruje kľúč pre spojenie K a pošle odpoveď účastníkovi B. Odpoveď zachytí útočník E a priamočiaro pokračuje v útoku:

3'. $T \to E(B) : S, \{N_E, K\}_{K_E}, \{N_B, K\}_{K_B}$
$E(T) \to B : M, \{N_E, K\}_{K_E}, \{N_B, K\}_{K_B}$

4'. $B \to E(A) : M, \{N_E, K\}_{K_E}$

Útočník dosiahol svoj cieľ a B nezistí, že nekomunikuje s A. Ochrana v tomto prípade môže spočívať v zabezpečení integrity šifrovanej správy, pridaní redundantného textu (vtedy nebude možné vhodne „zlepiť“ bloky šifrového textu), prípade vo výbere iného módu blokového šifrovacieho algoritmu.

14.5 Symetria správ

Správy s rovnakou štruktúrou z pohľadu dvoch alebo viacerých účastníkov protokolu sú potenciálnym zdrojom bezpečnostných problémov. Takéto správy útočník môže použiť v iných častiach alebo behoch protokolu, napríklad aj bez nutnosti správu dešifrovať. Útok na Wide Mouthed Frog protokol v časti 14.1 je príkladom útoku, ktorý využíva symetriu správ.

Uvažujme takýto protokol na vzájomnú autentifikáciu dvojice účastníkov:

1. $A \to B : N_A$
2. $B \to A : \{N_A, N_B\}_K$
3. $A \to B : N_B$

Protokol predpokladá, že dvojica účastníkov zdieľa spoločne kľúč K. Príležitostné slová N_A a N_B slúžia ako výzvy pre druhého účastníka protokolu. Schopnosť správne dešifrovať, príp. zašifrovať náhodný reťazec bitov zodpovedá znalosti kľúča K a dokazuje prítomnosť druhej strany.

Útok využíva paralelný beh dvoch inštancií protokolu. Útočník použije druhú inštanciu na konštrukciu vhodnej správy v druhom kroku a úspešne predstiera prítomnosť B v protokole.

1. $A \rightarrow E(B) : N_A$
1'. $E(B) \rightarrow A : N_A$
2'. $A \rightarrow E(B) : \{N_A, N'_A\}_K$
2. $E(B) \rightarrow A : \{N_A, N'_A\}_K$
3. $A \rightarrow E(B) : N'_A$
3'. $E(B) \rightarrow A : N'_A$

Bezpečnostný problém protokolu je spôsobený symetriou správy v druhom kroku protokolu. To znamená, že správu bolo možné použiť v inej inštancii bez znalosti K. Ochranou pred útokom je napríklad pridanie identifikátorov účastníkov do druhej správy:

2. $B \rightarrow A : \{A, B, N_A, N_B\}_K$

Needhamov-Schreoderov protokol s verejnými kľúčmi

Cieľom Needhamovho-Schreoderovho protokolu s verejnými kľúčmi (skrátene NSVK, 1978) je vzájomná autentifikácia účastníkov, s prípadnou dohodou o kľúči pre šifrovanie následnej komunikácie. NSVK nepoužíva dôveryhodnú tretiu stranu, predpokladá však, že účastníci navzájom poznajú svoje verejné kľúče. Označme K_A a K_B verejné kľúče účastníkov A a B. Príležitostné slovo N_A generuje účastník A, N_B účastník B.

1. $A \rightarrow B : \{N_A, A\}_{K_B}$
2. $B \rightarrow A : \{N_A, N_B\}_{K_A}$
3. $A \rightarrow B : \{N_B\}_{K_B}$

Príležitostné slová sú v správach prenášané šifrovane a po dokončení protokolu ich poznajú len účastníci A a B. To znamená, že z nich následne môžu skonštruovať kľúč pre spojenie.

Napriek jednoduchosti NSVK trvalo 17 rokov, kým bol objavený bezpečnostný problém využívajúci symetriu správy v druhom kroku protokolu [Low95]. Útočník využije fakt, že s ním účastník A nadviaže spojenie a zároveň začne komunikovať s B:

1. $A \rightarrow E : \{N_A, A\}_{K_E}$
1'. $E(A) \rightarrow B : \{N_A, A\}_{K_B}$
2'. $B \rightarrow E(A) : \{N_A, N_B\}_{K_A}$
2. $E \rightarrow A : \{N_A, N_B\}_{K_A}$
3. $A \rightarrow E : \{N_B\}_{K_E}$
3'. $E(A) \rightarrow B : \{N_B\}_{K_B}$

Obe inštancie NSVK sú úspešne dokončené, pričom na dokončenie „čiarkovanej“ využil E účastníka A ako orákulum. Na záver je B presvedčený, že komunikuje s A a útočník má k dispozícii obe príležitostné slová, teda môže skonštruovať aj kľúč pre spojenie.

Ochrana pred útokom môže spočívať napríklad v takomto porušení symetrie:

1. $A \rightarrow B : \{N_A, A\}_{K_B}$
2. $B \rightarrow A : \{N_A, N_B, B\}_{K_A}$
3. $A \rightarrow B : \{N_B\}_{K_B}$

14.6 Interakcia protokolov

V reálnom prostredí účastníci vystupujú v rôznych kryptografických protokoloch. Autentifikácia, distribúcia kľúča, platobné protokoly, slepé podpisy a ďalšie protokoly sú použité v aplikáciach, ktoré účastník musí alebo chce využívať. Úspešný beh protokolu obvyle vyžaduje prácu so zdieľaným symetrickým alebo súkromným kľúčom účastníka. Pokiaľ sú rovnaké kľúče používané v rôznych protokoloch, potenciálne môže byť jeden protokol použitý na útok proti druhému. Takáto nevhodná interakcia protokolov môže nastať napriek tomu, že samostatne sú protokoly bezpečné [KSW98].

Mnohé doteraz prezentované útoky využívali paralelný beh viacerých inštancií protokolu. Iný pohľad na tieto útoky je, že ide o nevhodnú interakciu protokolu, keď „útočiaci“ protokol je rovnaký ako ten, na ktorý sa útočí.

Bez ohľadu na to, či je nový protokol navrhnutý potenciálnym útočníkom alebo už v danom prostredí existuje, ak sú kľúče zdieľané medzi protokolmi, treba analyzovať bezpečnosť protokolov aj z hľadiska možnej interakcie. Ilustrujme nevhodnú interakciu protokolov na dvoch príkladoch.

Agora a protokol na overenie veku

Agora je protokol navrhnutý na jednoduché platenie za informácie na webových stránkach. Protokol využíva certifikáty a digitálne podpisy na zabezpečenie autenticity posielaných správ. Nakupujúceho účastníka označíme A, predávajúceho B. Symbolmi C_A a C_B označíme certifikáty ich verejných kľúčov. Predpokladáme, že certifikáty vydala dôveryhodná tretia strana. Nech M je požiadavka na zaslanie ceny, N je počítadlo požiadaviek a P cena za informáciu.

1. $A \rightarrow B : A, M$
2. $B \rightarrow A : \{C_B, N, P\}_{K_B^{-1}}$
3. $A \rightarrow B : \{C_A, N, P\}_{K_A^{-1}}$

V druhom a treťom kroku sú správy digitálne podpísané účastníkmi s použitím ich súkromných kľúčov (nie sú však šifrované).

K protokolu Agora je možné skonštruovať protokol, ktorý naruší jeho bezpečnosť. Tento protokol bude slúžiť na overovanie veku – ako ochrana prístupu k niektorým webovým stránkam. Predpokladajme, že certifikát obsahuje dátum narodenia, prípadne certifikát je vydaný len účastníkom s požadovaným vekom. Účastník dokáže

svoj vek znalosťou súkromného kľúča, teda schopnosťou podpísať náhodnú výzvu R.

1. $A \rightarrow B : A$
2. $B \rightarrow A : R$
3. $A \rightarrow B : \{C_A, R\}_{K_A^{-1}}$

Ak je dĺžka náhodnej výzvy R rovná súčtu dĺžok N a P, útočník postupuje takto (čiarkované kroky označujú inštanciu protokolu Agora):

1. $A \rightarrow E(D) : A$
1'. $E(A) \rightarrow B : A, M$
2'. $B \rightarrow E(A) : \{C_B, N, P\}_{K_B^{-1}}$
2. $E(D) \rightarrow A : R \qquad (R = N, P)$
3. $A \rightarrow E(D) : \{C_A, R\}_{K_A^{-1}}$
3'. $E(A) \rightarrow B : \{C_A, N, P\}_{K_A^{-1}}$

Útočník použije zreťazenie N, P ako náhodnú výzvu v protokole na overenie veku. Následná odpoveď A je potom priamo použiteľná ako odpoveď potvrdzujúca nákup v protokole Agora. Poznamenajme, že D môže byť akýkoľvek účastník.

Wide Mouthed Frog a bezpečné prihlasovanie

Wide Mouthed Frog protokol (WMF) bol popísaný v časti 14.1. Opäť môžeme skonštruovať protokol, pomocou ktorého sa dá naň zaútočiť. Tento protokol bude slúžiť na bezpečné prihlasovanie účastníkov, využijúc prítomnosť dôveryhodnej tretej strany T. Účastník A sa snaží prihlásiť na systém B pomocou hesla p. Symbolom R označíme náhodnú výzvu a H je hašovacia funkcia:

1. $A \rightarrow B : A$
2. $B \rightarrow A : R$
3. $A \rightarrow T : A, \{T_A, R, H(p), B\}_{K_A}$
4. $T \rightarrow B : \{T_T, H(R, H(p)), A\}_{K_B}$

Protokol využíva kľúče K_A a K_B, zdieľané medzi účastníkmi a T rovnako, ako vo WMF. Účastník B po dešifrovaní správy zo štvrtého kroku protokolu overí časovú pečiatku a správnosť hodnoty hašovacej funkcie (B pozná výzvu R aj hodnotu $H(p)$). Prirodzene, T tiež overí čerstvosť časovej pečiatky v treťom kroku.

Útočník použije protokol pre bezpečné prihlasovanie na predstieranie identity účastníka A vo WMF (kroky inštancie WMF sú označené čiarkovane):

1. $A \rightarrow E : A$
2. $E \rightarrow A : R \quad (R = B)$
3. $A \rightarrow E(T) : A, \{T_A, R, H(p), B\}_{K_A}$

1'. $E(A) \rightarrow T$: $A, \{T_A, B, K\}_{K_A}$ $\quad (K = H(p), B)$

2'. $T \rightarrow B$: $\{T_T, A, K\}_{K_B}$

Útok využíva tieto predpoklady o dĺžke objektov: $|R| = |B|$ a $|K| = |H(p), B|$. Útočník pozná kľúč $(K = H(p), B)$ a môže komunikovať s B, predstierajúc identitu A.

Eliminácia útokov využívajúcich interakciu protokolov nie je jednoduchá. Použitie každého kľúča len pre jeden protokol (aplikáciu) nie je vždy možné. Na druhej strane pomôže, ak je každé použitie kľúča zviazané s identifikátorom protokolu a konkrétneho kroku v rámci protokolu. Takéto protokoly potom sťažujú použitie správy z jedného protokolu v inom. Vo všeobecnosti si situácia, keď protokoly (aplikácie) majú neprázdny prienik používaných kľúčov vyžaduje dôkladnú analýzu.

15 Praktické doporučenia pre návrh protokolov

Mnohým útokom prezentovaným v časti 14 je možné zabrániť implementačnými opatreniami. Pamätanie si starých kľúčov pre spojenie a overovanie „inakosti“ nových znemožní útočníkovi vnútiť na komunikáciu starý kľúč. Na druhej strane však nutnosť pamätať si, chrániť (lebo aj staré kľúče treba chrániť pred prezradením) a overovať kľúče kladie zvýšené nároky na prevádzku protokolu. Podobne kontrola paralelných spojení síce znemožní útoky vedené prostredníctvom paralelných behov protokolov, avšak znižuje výkon (z hľadiska počtu realizácií protokolu) a komplikuje implementáciu protokolu.

Cieľom návrhu kryptografického protokolu má byť taký protokol, ktorého bezpečnostné vlastnosti sú garantované vlastnou konštrukciou a postupnosťou správ, spolu s presne sformulovanými predpokladmi. Protokoly vyžadujúce špeciálne implementačné opatrenia na elimináciu útokov považujeme za slabé.

V tejto časti uvedieme niektoré princípy a doporučenia pre konštrukciu kryptografických protokolov [AN94, AN95]. Uvedené princípy nie sú nutnou ani postačujúcou podmienkou návrhu bezpečných protokolov [Syv96]. Pomáhajú však precíznejšie formulovať návrh a vyvarovať sa najčastejších chýb.

1. Explicitnosť
Význam správy má závisieť len na správe samotnej. Správa má obsahovať všetky údaje potrebné pre jej interpretáciu, vrátane identifikátorov účastníkov.

Nech dôveryhodný server posiela správu účastníkovi A s významom „K je dobrý kľúč na šifrovanie spojenia s účastníkom B“. Potom v posielanej správe majú byť okrem K uvedené aj identifikátory účastníkov A a B. Príkladom protokolov, kde tento princíp nebol dodržaný, sú Dennigovej-Saccov protokol (časť 14.1) alebo Needhamov-Schroederov protokol s verejnými kľúčmi (časť 14.5). Samozrejme, princíp sa netýka len identifikátorov účastníkov, ale všetkých údajov – príležitostných slov, časových pečiatok, kľúčov a podobne.

Ďalšia požiadavka vyžaduje formuláciu predpokladov pre všetky časti protokolu. Uvedenie predpokladov dovolí komukoľvek overiť ich správnosť a dostatočnosť. Ob-

vyklé predpoklady sú o zdieľaní (znalosti) kľúčov, o schopnosti generovať kľúče (alebo príležitostné slová), alebo o synchronizácii vnútorných hodín účastníkov.

2. Predpoklady
Pre každú správu, na základe ktorej sa má vykonať nejaká akcia, je potrebné uviesť všetky vyžadované predpoklady.

3. Použitie šifrovania
Pri použití šifrovania je potrebné uviesť, aký význam má šifrovanie konkrétneho textu.

Význam šifrovania správ v protokole môže byť v rôznych kontextoch odlišný. V prípade asymetrického šifrovania s použitím verejného kľúča je význam obvykle zrejmý. Symetrické šifrovanie môže mať v protokole nasledujúce ciele (prípadne ich kombináciu):

- Dôvernosť – primárny cieľ šifrovania. Ak účastník získa $\{X\}_K$, môže usúdiť, že správa bola určená pre niekoho, kto pozná dešifrovací kľúč. Ak účastník pozná dešifrovací kľúč, môže (za istých podmienok) dedukovať, že správa je určená práve jemu.
- Autentickosť – pokiaľ je použitý symetrický kľúč známy len dvom účastníkom a títo dokážu rozpoznať vlastné správy; prípadne, ak účastník použije svoj súkromný kľúč. Príkladom šifrovania z dôvodov autentickosti je posielanie šifrovanej časovej pečiatky v protokole na vyžiadanie aktuálneho času (pozri neskôr v tejto kapitole).
- Previazanie správ – šifrovanie „drží pokope“ časti textu. Obvykle $\{X, Y\}_K$ nie je to isté ako $\{X\}_K, \{Y\}_K$. V opačnom prípade je možné správy rozdeľovať a opätovne lepiť dokopy, čo vytvára možnosti pre potenciálny útok (pozri napríklad útok na Needhamov-Schroederov protokol pri použití ECB módu v časti 14.4)
- Náhodnosť – šifrovanie je možné použiť na generovanie pseudonáhodných čísel, prípadne na znižovanie možnosti predikovať údaje („znáhodňovanie“). Príkladom druhého použitia je úprava protokolu na vyžiadanie aktuálneho času (pozri neskôr v tejto kapitole)

4. Podpisovanie a šifrovanie
Digitálny podpis šifrovanej správy nezaručuje, že podpisujúci účastník pozná otvorený text správy.

Doporučovaný postup, ak je potrebné zabezpečiť autentickosť správy podpisom a dôvernosť šifrovaním, je najskôr podpísať otvorený text a následne správu zašifrovať. Na druhej strane, ani tento postup postup nie je zárukou bezpečnosti, ako ukazuje príklad Denningovej-Saccovho protokolu (časť 14.1).

5. Príležitostné slová
Pre každé príležitostné slovo je potrebné uviesť význam použitia a očakávané vlastnosti.

Obvyklý význam použitia príležitostného slova je zabezpečiť časovú súslednosť správ a tým presvedčiť účastníka o čerstvosti prijatých údajov. Niekedy sa príležitostné slovo viaže s iným údajom a slúži ako jeho náhrada. V Otwayovom-Reesovom protokole (pozri časť 14.3) slúžia N_A, N_B aj ako náhrada za identifikátory účastníkov A, B (sú s nimi previazané pomocou šifrovania):

1. $A \rightarrow B : M, A, B, \{N_A, M, A, B\}_{K_A}$
2. $B \rightarrow T : M, A, B, \{N_A, M, A, B\}_{K_A}, \{N_B, M, A, B\}_{K_B}$
3. $T \rightarrow B : M, \{N_A, K\}_{K_A}, \{N_B, K\}_{K_B}$
4. $B \rightarrow A : M, \{N_A, K\}_{K_A}$

Ak by boli N_A a N_B použité len na zabezpečenie čerstvosti správ v treťom a štvrtom kroku, tak by v prvých dvoch krokoch nemuseli byť posielané šifrovane. Prvé dva kroky protokolu by potom vyzerali takto (šifrové texty vypadli úplne, lebo zodpovedajúce otvorené texty sú známe a šifrovanie tu nie je použité na zabezpečenie autentickosti):

1. $A \rightarrow B : M, A, B, N_A$
2. $B \rightarrow T : M, A, B, N_A, N_B$

V upravenom protokole útočník dokáže predstierať identitu B a vnútiť účastníkovi A kľúč pre spojenie, ktorý bude poznať:

1. $A \rightarrow E(B) : M, A, B, N_A$

1'. $E \rightarrow A : M', E, A, N_E$

2'. $A \rightarrow E(T) : M', E, A, N_E, N'_A$
$E(A) \rightarrow T : M', E, A, N_E, N_A$

3'. $T \rightarrow E(A) : M', \{N_E, K\}_{K_E}, \{N_A, K\}_{K_A}$

4. $E(B) \rightarrow A : M, \{N_A, K\}_{K_A}$

Keďže príležitostné slová slúžili ako náhrada za identifikátory účastníkov, doplňme tieto do správ v treťom a štvrtom kroku protokolu. Dostaneme jednoduchší a logicky prehľadnejší protokol (identifikátor M bol odstránený, lebo neplnil žiadnu funkciu):

1. $A \rightarrow B : A, B, N_A$
2. $B \rightarrow T : A, B, N_A, N_B$
3. $T \rightarrow B : \{N_A, A, B, K\}_{K_A}, \{N_B, A, B, K\}_{K_B}$
4. $B \rightarrow A : \{N_A, A, B, K\}_{K_A}$

Pri návrhu protokolu je dôležité určiť, či konkrétne príležitostné slovo musí byť nepredikovateľné (dopredu „neuhádnuteľné“ útočníkom), alebo by ho stačilo nahradiť počítadlom. Spôsob zabezpečenia nepredikovateľnosti môže spočívať v použití kvalitného (pseudo)náhodného generátora, šifrovaní a podobne. Ilustrujme to na ďalšom protokole.

Protokol na vyžiadanie aktuálneho času

Účastník A v protokole získa aktuálny čas od dôveryhodného servra S, s ktorým zdieľa kľúč K_A. Aktuálny čas je reprezentovaný časovou pečiatkou T_S. Čerstvosť prijatej správy z pohľadu účastníka A zabezpečuje príležitostné slovo N_A, ktoré si A vygeneruje na začiatku protokolu.

1. $A \rightarrow S : A, N_A$
2. $S \rightarrow A : \{T_S, N_A\}_{K_A}$

Pre bezpečnosť protokolu je nevyhnutné, aby bol reťazec N_A náhodný a nepredikovateľný, čo príležitostné slová za normálnych okolností sú. V opačnom prípade, teda ak by N_A bolo predikovateľné (napr. počítadlo), tak útočník E dokáže poskytnúť A starú časovú pečiatku. Najskôr E určí hodnotu N_A, ktorú A použije v ďalšej inštancii protokolu a vypýta si od S šifrovanú časovú pečiatku:

1. $E(A) \rightarrow S : A, N_A$
2. $S \rightarrow E(A) : \{T_S, N_A\}_{K_A}$

Keď chce neskôr A získať aktuálny čas, útočník poskytne starú správu:

1. $A \rightarrow E(S) : A, N_A$
2. $E(S) \rightarrow A : \{T_S, N_A\}_{K_A}$

Ak je účastníkom generované príležitostné slovo nepredikovateľné, tento útok nie je realizovateľný. Miernou úpravou protokolu sa dá podmienka nepredikovateľnosti odstrániť:

1. $A \rightarrow S : A, \{N_A\}_{K_A}$
2. $S \rightarrow A : \{T_S, \{N_A\}_{K_A}\}_{K_A}$

Keďže kľúč K_A poznajú len A a S, hodnota $\{N_A\}_{K_A}$ je nepredikovateľná pre kohokoľvek mimo A a S, aj keď je samotné N_A ľahko uhádnuteľné. Stačia k tomu obvyklé predpoklady o kvalite použitého šifrovacieho algoritmu.

> *6. Ochrana predikovateľných údajov*
> Predikovateľné údaje (počítadlá) použité na zabezpečenie čerstvosti správ musia byť v protokole chránené.

Siedmy princíp je formuláciou prirodzenej požiadavky na synchronizáciu hodín účastníkov pri používaní časových pečiatok.

> *7. Časové pečiatky*
> Ak sú na zabezpečenie čerstvosti použité časové pečiatky, je potrebné synchronizovať lokálne časy. Navyše, systém „správy času“ sa stáva kritickým komponentom protokolu.

Ilustráciou ďalšieho princípu je Needhamov-Schroederov protokol (časť 14.1), v ktorom je kľúč pre spojenie použitý na šifrovanie nového príležitostného slova. Toto čerstvé použitie však nezaručuje čerstvosť samotného kľúča.

8. Čerstvosť vs. použitie
Aktuálne použitie entity (napr. kľúča na šifrovanie) nie je to isté ako čerstvosť entity.

9. Jednoznačnosť
Správa protokolu má byť jednoznačne dekódovateľná – účastník je schopný určiť príslušnosť správy do protokolu, beh protokolu ako aj poradie správy v protokole.

Dodržaním deviateho princípu zamedzíme útokom, ktoré použijú správy získané v konkrétnom kroku inštancie protokolu v inom kroku, v inej inštancii alebo v inom protokole. Príkladom je útok na Wide Mouthed Frog protokol (časť 14.1), v ktorom bola správa z druhého kroku použitá v prvom kroku protokolu. Útoky umožnené interakciou protokolov (časť 14.6) využívali správy z jedného protokolu pre útok na druhý protokol. Podobná situácia je aj pri útokoch využívajúcich paralelné inštancie protokolu.

10. Dôvera
Je potrebné uviesť a zdôvodniť oprávnenosť (ako aj vhodnosť) všetkých predpokladov o dôvere, ktoré protokol vyžaduje.

Dôvera, ktorú účastníci majú v konanie a schopnosti ostatných účastníkov, musí byť špecifikovaná. Niekedy sa dôvera prekrýva s predpokladmi, napríklad schopnosť generovať kvalitný kľúč pre spojenie. Inokedy protokol vyžaduje dôveru v to, že iný účastník dodržiava protokol a nebude sa od neho odchylovať – inými slovami, účastník nebude nadväzovať spojenie s potenciálnym útočníkom.

11. Použitie súkromného kľúča
Ak je to možné, treba sa vyhnúť použitiu súkromného kľúča na rôzne účely, napríklad na podpisovanie a dešifrovanie.

Použitie súkromného kľúča na rôzne účely predstavuje riziko, že použitie jedným spôsobom bude zneužiteľné voči druhému použitiu. Napríklad v prípade RSA dešifrovanie a zverejnenie dešifrovaných dát môže byť využité na získanie digitálneho popisu. Vo všeobecnosti treba opatrne pristupovať k podpisovaniu a dešifrovaniu s použitím súkromného kľúča, aby sa účastník nestal orákulom pre útočníka. Využitie účastníka ako dešifrovacieho orákula je demonštrované na protokole TMN.

TMN protokol

Autormi TMN protokolu sú Tatebayashi, Matsuzaki a Newman (1989). Cieľom protokolu je distribúcia kľúča pre spojenie, prostredníctvom dôveryhodnej tretej strany S. Označme verejný kľúč servra K_S. Účastníci A a B v každej inštancii protokolu vytvárajú vlastné symetrické kľúče K_A a K_B, z ktorých sa K_B stane kľúčom na šifrovanie ich následnej vzájomnej komunikácie.

1. $A \rightarrow S : B, \{K_A\}_{K_S}$
2. $S \rightarrow B : A$

3. $B \rightarrow S : A, \{K_B\}_{K_S}$
4. $S \rightarrow A : B, \{K_B\}_{K_A}$

Správy v prvom a treťom kroku protokolu sú šifrované asymetrickým systémom s použitím verejného kľúča S, správa vo štvrtom kroku je šifrovaná symetrickým systémom. TMN protokol nezabezpečuje autentifikáciu, lebo server ani žiadny z účastníkov nedokáže zistiť, kto je pôvodcom konkrétnej správy. Chýbajúca autentifikácia je príčinou útoku, v ktorom útočník E získa kľúč pre spojenie:

1. $A \rightarrow E(S) : B, \{K_A\}_{K_S}$
   $E(A) \rightarrow S : B, \{K_A\}_{K_S}$

1'. $E(A) \rightarrow S : B, \{K_E\}_{K_S}$

2'. $S \rightarrow B : A$

3'. $B \rightarrow S : A, \{K_B\}_{K_S}$

4'. $S \rightarrow E(A) : B, \{K_B\}_{K_E}$

2. $S \rightarrow E(B) : A$

3. $E(B) \rightarrow S : A, \{K_B\}_{K_S}$

4. $S \rightarrow A : B, \{K_B\}_{K_A}$

Pre účastníkov A a B protokol prebehol ako mal, avšak E pozná kľúč K_B.

Predpokladajme, že autentifikácia je v konkrétnom prostredí nasadenia zabezpečená inými prostriedkami, prípadne nás vôbec nezaujíma. Nech je šifrovanie v poslednom kroku realizované ako súčet modulo 2 (xor). Keďže K_A a K_B sú volené náhodne a nezávisle a sú známe len účastníkom A, B a dôveryhodnému S, takéto šifrovanie znemožuje útočníkovi získať K_B. Nech je použitý asymetrický systém RSA s verejným exponentom $e = 3$ (kapitola 5). To znamená, že výpočtová zložitosť je čo najviac presunutá na server S. Protokol s takouto požiadavkou je dôležitý v situáciach, keď sú účastníci realizovaní výpočtovo obmedzenými zariadeniami, ako sú napríklad čipové karty. TMN protokol potom prebieha takto (n označuje modul verejného kľúča S):

1. $A \rightarrow S : B, K_A^3 \bmod n$
2. $S \rightarrow B : A$
3. $B \rightarrow S : A, K_B^3 \bmod n$
4. $S \rightarrow A : B, K_A \oplus K_B$

Účastník A následne pripočítaním K_A získa kľúč pre spojenie: $K_A \oplus (K_A \oplus K_B) = K_B$. Útočník E dokáže využiť algebraické vlastnosti použitých šifrovacích algoritmov a zistiť kľúč K_B. Protokol medzi A, B a S prebehne tak, ako je uvedený, pričom útočník odpočuje správu prenášanú v treťom kroku. Potom E použije S ako orákulum pre pomocný výpočet:

1'. $E \rightarrow S : D, K_E^3 \cdot K_B^3 \bmod n$

2'. $S \rightarrow E(D) : E$

3'. $E(D) \rightarrow S : E, K_D^3 \bmod n$

4'. $S \rightarrow E : D, K_E \cdot K_B \oplus K_D$

Útočník z poslednej správy vypočíta K_B, keďže K_E a K_D pozná. Pokiaľ sa rozhodnú útočiť dvaja útočníci (E a D) spoločne, tak pri útoku nie je potrebné predstierať cudziu identitu.

TMN protokol ilustruje aj posledný princíp:

> *12. Nič nepredpokladať*
> V protokole nepredpokladajme nič, o čom sa nemôžeme sami presvedčiť a nie je v predpokladoch protokolu (napr. účastník niečo pozná, správa je v určitom tvare, atď.).

V TMN protokole takýto predpoklad možno vystopovať v tom, že dôveryhodný server S po prijatí správ v prvom a treťom kroku predpokladá, že účastníci poznajú príslušné otvorené texty. To však nemusí byť pravda a S sa o tom nedokáže presvedčiť. Práve takýto zamlčaný predpoklad útočník dokáže využiť na útok.

V úvode tejto časti sme zdôraznili, že uvedené princípy a doporučenia nie sú nutnou ani postačujúcou podmienkou návrhu bezpečných protokolov. Dokonca sú situácie, v ktorých je nutné princípy porušiť, napríklad pri realizácii slepých podpisov (keď účastník podpisuje údaje, o ktorých nič nevie). V každom prípade sú doporučenia vhodnou pomôckou pri návrhu kryptografických protokolov.

16 BAN logika

BAN logika je formálny systém, umožňujúci analyzovať autentifikačné protokoly. Názov je odvodený z mien autorov – M. Burrows, M. Abadi, R. Needham (1989). BAN logika sa stala prvým úspešným formalizmom, ktorý dokázal odhaliť viaceré bezpečnostné slabiny v kryptografických protokoloch.

Cieľom BAN logiky je analyzovať protokol a dať odpovede na nasledujúce otázky:

1. Čo dosiahneme protokolom? Robí to, čo chceme?
2. Aké predpoklady vyžaduje?
3. Robí niečo redundantné (nadbytočné kroky alebo správy)?

Vzhľadom na zameranie a použitý formálny aparát, analýza pomocou BAN logiky nedokáže identifikovať:

- chyby zanesené konkrétnymi implementáciami protokolu,
- problémy spôsobené autentifikáciou nečestného účastníka,
- slabiny použitých kryptografických primitív v protokole, ...

BAN logika je založená na „vierach“ čestných účastníkov protokolu. Analýza protokolu znamená odvodenie tých faktov, ktorým môžu čestní účastníci protokolu veriť. Príkladom takej viery je: „kľúč, ktorý účastník A obdržal od servera je dobrý kľúč na šifrovanie komunikácie s B“. Na začiatku protokolu majú účastníci prvotné viery – zhodné s predpokladmi protokolu. Postupne, prijímaním správ od ostatných účastníkov, vznikajú nové viery. Jednotlivé správy protokolu sú prevedené do formálneho jazyka – sú idealizované.

V tejto časti prezentujeme BAN logiku v rozsahu, ktorý je potrebný na pochopenie jej princípov. Ukážeme použitie BAN logiky na analýze Needhamovho-Schroederovho protokolu.

16.1 Jazyk logiky

Viery a idealizovanú komunikáciu v BAN logike budeme zapisovať pomocou nasledujúcich formúl. Označme P, Q účastníkov protokolu, nech K je šifrovací kľúč a nech X je formula, prípadne správa alebo jej časť.

$\{X\}_K$ – X šifrované s použitím kľúča K. Predpokladá sa, že účastníci dokážu rozpoznať vlastné správy.

$P \models X$ – P verí, že X je pravda a správa sa podľa toho.

P **sees** X – P obdržal X (dostal ako správu), pričom k nej možno musel dospieť dešifrovaním.

P **said** X – P niekedy poslal X a veril (vtedy), že X je pravda.

P **controls** X – P riadi a je dôveryhodný ohľadne X (napríklad generovanie kľúčov).

$\#(X)$ – X je „čerstvé“, nebolo poslané v žiadnej správe pred aktuálnym behom protokolu.

$P \overset{K}{\longleftrightarrow} Q$ – K je dobrý kľúč pre P a Q, teda K poznajú len účastníci P, Q a účastníci, ktorým títo dôverujú.

$\overset{K}{\mapsto} P$ – K je verejným kľúčom P.

Poznamenajme, že v zápise BAN logiky nerozlišujeme medzi symetrickým a asymetrickým šifrovaním. Zápis je rovnaký $\{X\}_K$. Ak je K verejný kľúč účastníka P, tak zodpovedajúci súkromný kľúč sa označuje K^{-1}. Potom zápis $\{X\}_{K^{-1}}$ označuje účastníkom P digitálne podpísané X.

16.2 Pravidlá a analýza protokolu

Odvodzovanie nových vier účastníkov vyžaduje manipuláciu s formulami BAN logiky. Základné inferenčné pravidlá BAN logiky odrážajú obvyklú interpretáciu situácie po prijatí správy, príležitostného slova a podobne.

Význam správy (message-meaning):

$$\frac{P \models P \overset{K}{\longleftrightarrow} Q,\ P \textbf{ sees } \{X\}_K}{P \models Q \textbf{ said } X} \qquad \text{(R1)}$$

a verzia pre asymetrické systémy:

$$\frac{P \models (\overset{K}{\mapsto} Q),\ P \textbf{ sees } \{X\}_{K^{-1}}}{P \models Q \textbf{ said } X}$$

Pravidlo významu správy znamená, že ak P obdrží X šifrované kľúčom K a tento kľúč je dobrý kľúč na šifrovanie komunikácie medzi P a Q, tak bude veriť, že Q (niekedy) poslal X.

Overenie príležitostného slova (nonce-verification):

$$\frac{P \models \#(X),\ P \models Q\ \textbf{said}\ X}{P \models (Q \models X)} \tag{R2}$$

Overenie príležitostného slova umožňuje účastníkovi P usudzovať o čerstvosti vier ostatných účastníkov. Keďže v analýze predpokladáme čestných účastníkov, je prirodzené usudzovať, že to, čo účastník nedávno tvrdil, je zároveň to, čomu sám verí.

Jurisdikcia:

$$\frac{P \models (Q\ \textbf{controls}\ X),\ P \models (Q \models X)}{P \models X} \tag{R3}$$

Jurisdikcia sa týka odvodzovania s konštrukciou **controls** a umožňuje „prenos“ viery od dôveryhodného účastníka.

Ďalšie pravidlá:

$$\frac{P \models P \overset{K}{\longleftrightarrow} Q,\ P\ \textbf{sees}\ \{X\}_K}{P\ \textbf{sees}\ X} \tag{R4}$$

Pravidlo (R4) hovorí o schopnosti účastníka P dešifrovať prijatú správu, ak má k dispozícii kľúč.

$$\frac{P \models \#(X)}{P \models \#(X, Y)} \tag{R5}$$

Čerstvosť X zaručuje čerstvosť ľubovoľnej správy, ktorej je súčasťou. Obrátené pravidlo neplatí. Ak je čerstvá dvojica (X, Y), neznamená to ešte čerstvosť samostatného X (lebo Y môže byť čerstvé) ani čerstvosť samostatného Y (lebo X môže byť čerstvé).

$$\frac{P\ \textbf{sees}\ (X, Y)}{P\ \textbf{sees}\ X} \tag{R6}$$

Pravidlo (R6) popisuje rozkladanie zložených správ. Predpokladáme, že správy v protokole sú jednoznačne rozložiteľné na zložky.

Význam niektorých ďalších (prirodzených) pravidiel je zrejmý:

$$\frac{P \models X,\ P \models Y}{P \models (X, Y)}$$

$$\frac{P \models (Q \models (X, Y))}{P \models (Q \models X)}$$

$$\frac{P \models (Q\ \textbf{said}\ (X, Y))}{P \models (Q\ \textbf{said}\ X)}$$

Postup pri analýze

Analýza protokolu má v BAN logike štyri fázy:

1. Idealizácia protokolu – prepis do formalizmu BAN logiky
2. Sformulovanie predpokladov – definovanie počiatočných vier účastníkov
3. Analýza nových vier po každom kroku protokolu – z obdržanej správy môžu (ale nemusia) účastníkovi vzniknúť nové viery
4. Výsledok protokolu – viery účastníkov po poslednom kroku protokolu

Pre BAN logiku je charakterická monotónna inferencia, teda viery účastníkov môžu len pribúdať. Obvyklé ciele autentifikačných protokolov môžeme sformalizovať takto:

$$A \models A \overset{K}{\longleftrightarrow} B \qquad \& \qquad B \models A \overset{K}{\longleftrightarrow} B,$$

alebo ešte navyše

$$A \models (B \models A \overset{K}{\longleftrightarrow} B) \qquad \& \qquad B \models (A \models A \overset{K}{\longleftrightarrow} B).$$

16.3 Needhamov-Schroederov protokol – analýza

Pripomeňme Needhamov-Schreoderov protokol s účastníkmi A, B a dôveryhodnou treťou stranou S:

1. $A \to S$: A, B, N_A
2. $S \to A$: $\{N_A, B, K, \{K, A\}_{K_B}\}_{K_A}$
3. $A \to B$: $\{K, A\}_{K_B}$
4. $B \to A$: $\{N_B\}_K$
5. $A \to B$: $\{N_B - 1\}_K$

Kľúče K_A a K_B zdieľajú A a B s treťou stanou S a slúžia na šifrovanie vzájomnej komunikácie. Kľúč pre spojenie K generuje S a je určený na šifrovanie následnej komunikácie medzi A a B. Príležitostné slová N_A a N_B sú vygenerované samotnými účastníkmi ako dostatočne dlhé náhodné reťazce bitov.

Idealizácia

Postupne prejdeme jednotlivé fázy analýzy protokolu pomocou BAN logiky a ukážeme, ako spomenutú slabinu dokáže BAN logika zachytiť. Idealizovaný protokol:

2. $S \to A$: $\{N_A, A \overset{K}{\longleftrightarrow} B, \{A \overset{K}{\longleftrightarrow} B\}_{K_B}\}_{K_A}$
3. $A \to B$: $\{A \overset{K}{\longleftrightarrow} B\}_{K_B}$
4. $B \to A$: $\{N_B\}_K$
5. $A \to B$: $\{N_B\}_K$

Správy alebo ich časti, posielané ako otvorený text, môžu byť útočníkom ľahko zamenené. V idealizovanom protokole preto takéto správy nevystupujú. Prítomnosť kľúča K v druhej správe protokolu bola nahradená tvrdením o jeho vhodnosti pre

použitie účastníkmi A a B, lebo práve to je význam správy, ktorú posiela S. Podobne bol nahradený výskyt kľúča K aj v treťom kroku.

BAN logika nemá prostriedky na vyjadrenie aritmetických operácií, preto je výraz $N_B - 1$ idealizovaný ako N_B. Pôvodný význam odčítania jednotky je odlíšiť štvrtú a piatu správu v protokole, s cieľom zamedziť útoku opakovaním. Idealizácia toto odlíšenie nepotrebuje – je zabezpečené prostredníctvom toho, kto je adresátom konkrétnej správy a tým, že BAN logika uvažuje len o čestných účastníkoch protokolu.

Predpoklady

Predpoklady protokolu sú zároveň úvodnými vierami účastníkov. Protokol predpokladá existenciu kľúčov K_A a K_B:

(P1) $A \models A \overset{K_A}{\longleftrightarrow} S$ (P2) $S \models A \overset{K_A}{\longleftrightarrow} S$

(P3) $B \models B \overset{K_B}{\longleftrightarrow} S$ (P4) $S \models B \overset{K_B}{\longleftrightarrow} S$

Predpoklady (P2) a (P3) analýza nevyužije, uvádzame ich pre úplnosť. Účastníci veria, že nimi vygenerované príležitostné slová sú čerstvé:

(P5) $A \models \#(N_A)$ (P6) $B \models \#(N_B)$

Dôležitým predpokladom protokolu je viera účastníkov, že S vygeneruje dobrý kľúč K a prezradí ho len im:

(P7) $A \models S$ **controls** $A \overset{K}{\longleftrightarrow} B$

(P8) $B \models S$ **controls** $A \overset{K}{\longleftrightarrow} B$

Analýza

Spracujme najskôr správu z druhého kroku protokolu:

$$
\begin{array}{l}
A \textbf{ sees } \{N_A, A \overset{K}{\longleftrightarrow} B, \{A \overset{K}{\longleftrightarrow} B\}_{K_B}\}_{K_A} \\
\qquad \Downarrow \quad \text{(P1), (R1)} \\
A \models S \textbf{ said } (N_A, A \overset{K}{\longleftrightarrow} B, \{A \overset{K}{\longleftrightarrow} B\}_{K_B}) \\
\qquad \Downarrow \\
A \models S \textbf{ said } (N_A, A \overset{K}{\longleftrightarrow} B)
\end{array}
\tag{16.1}
$$

Aplikáciou (R5) a predpokladu (P5) o čerstvosti N_A dostaneme:

$$
\begin{array}{l}
A \models \#(N_A, A \overset{K}{\longleftrightarrow} B) \\
\quad \Big\Downarrow \quad \text{(16.1), (R2)} \\
A \models S \models (N_A, A \overset{K}{\longleftrightarrow} B) \\
\quad \Big\Downarrow \\
A \models S \models A \overset{K}{\longleftrightarrow} B \qquad (16.2) \\
\quad \Big\Downarrow \quad \text{(P7), (R3)} \\
A \models A \overset{K}{\longleftrightarrow} B \qquad (16.3)
\end{array}
$$

Teda účastník A je presvedčený, že kľúč K je dobrý (vhodný) kľúč na šifrovanie komunikácie medzi ním a účastníkom B.

Tretia správa vedie k nasledujúcim záverom:

$$
\begin{array}{l}
B \textbf{ sees } \{A \overset{K}{\longleftrightarrow} B\}_{K_B} \\
\quad \Big\Downarrow \quad \text{(P3), (R1)} \\
B \models S \textbf{ said } A \overset{K}{\longleftrightarrow} B
\end{array}
$$

Účastník B však nemá možnosť overiť platnosť formuly $S \models A \overset{K}{\longleftrightarrow} B$, pretože ku kľúču K sa neviaže nič čerstvé. Na tom už nič nezmenia ani ďalšie dva kroky protokolu.

Poznámky k analýze Pridajme k predpokladom (prirodzenú) vieru účastníkov v čerstvosť kľúčov generovaných S:

$$
\begin{array}{ll}
\text{(P9)} & A \models S \textbf{ controls } \#(A \overset{K}{\longleftrightarrow} B) \\
\text{(P10)} & B \models S \textbf{ controls } \#(A \overset{K}{\longleftrightarrow} B),
\end{array}
$$

Potom je možné pre účastníka A po druhej správe z (P9), (16.2) a pravidla jurisdikcie (R3) odvodiť

$$A \models \#(A \overset{K}{\longleftrightarrow} B). \qquad (16.4)$$

Tieto dodatočné predpoklady nemenia problém, ktorý má účastník B s čerstvosťou kľúča K.

Idealizácia posledných dvoch správ v protokole môže byť aj takáto:

4. $B \to A$: $\{N_B, A \overset{K}{\longleftrightarrow} B\}_K$
5. $A \to B$: $\{N_B, A \overset{K}{\longleftrightarrow} B\}_K$

Pridanie formúl o vhodnosti kľúča K je odôvodnené tým, že účastník použil kľúč K na šifrovanie vzájomnej komunikácie a je (mal by byť) teda presvedčený o jeho

vhodnosti. Pri takejto idealizácii dokážeme v štvrtom kroku pre účastníka A odvodiť navyše:

$$
\begin{array}{l}
A \textbf{ sees } \{N_B, A \overset{K}{\longleftrightarrow} B\}_K \\
\qquad \Downarrow \quad \text{(16.3), (R1)} \\
A \models B \textbf{ said } (N_B, A \overset{K}{\longleftrightarrow} B\}_K) \\
\qquad \Downarrow \\
A \models B \textbf{ said } A \overset{K}{\longleftrightarrow} B \\
\qquad \Downarrow \quad \text{(16.4), (R2)} \\
A \models B \models A \overset{K}{\longleftrightarrow} B
\end{array}
$$

Uvedená idealizácia (opäť) nepomôže prekonať účastníkovi B bezpečnostný problém s čerstvosťou K.

16.4 Opravený Needhamov-Schroederov protokol

Odstráňme problém s čerstvosťou správy v treťom kroku protokolu pridaním príležitostného slova. Toto príležitostné slovo bude generovať B a poskytne ho S ešte pred vytvorením kľúča K. Opravený Needhamov-Schroederov protokol:

1. $A \to B$: A, B, N_A
2. $B \to S$: A, B, N_A, N_B
3. $S \to A$: $\{N_A, B, K, \{N_B, A, K\}_{K_B}\}_{K_A}$
4. $A \to B$: $\{N_B, A, K\}_{K_B}$

Idealizácia protokolu je analogická s idealizáciou pôvodného protokolu:

3. $S \to A$: $\{N_A, A \overset{K}{\longleftrightarrow} B, \{N_B, A \overset{K}{\longleftrightarrow} B\}_{K_B}\}_{K_A}$
4. $A \to B$: $\{N_B, A \overset{K}{\longleftrightarrow} B\}_{K_B}$

Predpoklady protokolu sú nezmenené. Správa pre účastníka A od S ostala rovnaká, len v časti šifrovanej pre B pribudlo N_B. Preto odvodenie vier účastníka A je totožné s odvodením v predchádzajúcej časti. Analyzujme viery účastníka B po prijatí správy v štvrtom kroku opraveného protokolu:

$$
\begin{array}{l}
B \textbf{ sees } \{N_B, A \overset{K}{\longleftrightarrow} B\}_{K_B} \\
\qquad \Downarrow \quad \text{(P3), (R1)} \\
B \models S \textbf{ said } (N_B, A \overset{K}{\longleftrightarrow} B)
\end{array}
\tag{16.5}
$$

Z predpokladu (P6) použitím pravidla (R5) dostaneme:

$$
\begin{array}{l}
B \models \#(N_B, A \overset{K}{\longleftrightarrow} B) \\
\quad \Big\Downarrow \quad \text{(16.5), (R1)} \\
B \models S \models (N_B, A \overset{K}{\longleftrightarrow} B) \\
\quad \Big\Downarrow \\
B \models S \models A \overset{K}{\longleftrightarrow} B \\
\quad \Big\Downarrow \quad \text{(P8), (R3)} \\
B \models A \overset{K}{\longleftrightarrow} B
\end{array}
$$

Teda výsledkom opraveného protokolu sú viery oboch účastníkov vo vhodnosť kľúča K na šifrovanie vzájomnej komunikácie:

$$A \models A \overset{K}{\longleftrightarrow} B \qquad \& \qquad B \models A \overset{K}{\longleftrightarrow} B.$$

Podobne ako v predchádzajúcej časti, môžeme aj teraz uvažovať o predpokladoch (P9) a (P10). Tie umožnia odvodiť nasledujúce viery účastníkov:

$$A \models \#(A \overset{K}{\longleftrightarrow} B) \qquad \& \qquad B \models \#(A \overset{K}{\longleftrightarrow} B).$$

Ak si následne A a B vymenia správy šifrované kľúčom K (v zátvorke je uvedená zodpovedajúca idealizácia):

5. $B \to A$: $\{m_1\}_K$ $\qquad (\{m_1, A \overset{K}{\longleftrightarrow} B\}_K)$
6. $A \to B$: $\{m_2\}_K$ $\qquad (\{m_2, A \overset{K}{\longleftrightarrow} B\}_K)$,

tak v analýze možno dospieť k vieram

$$A \models B \models A \overset{K}{\longleftrightarrow} B \qquad \& \qquad B \models A \models A \overset{K}{\longleftrightarrow} B.$$

Záverečné poznámky k BAN logike

BAN logika bola prvým významným formálnym systémom pre analýzu autentifikačných protokolov. Jej výhodou je jednoduchý aparát – neveľký počet pravidiel a formúl. Na druhej strane nie je BAN logika schopná analyzovať niektoré typy protokolov (napr. Diffieho-Hellmanov protokol). Ďalšími nevýhodami sú neformálna idealizácia protokolu (nie je presne definovaný prechod od protokolu k jeho idealizácii) a analýza len s čestnými účastníkmi protokolu.

Nasledovníci BAN logiky sa snažia spomenuté (aj ďalšie) nedostatky odstrániť a tým rozšíriť možnosti použitia takýchto formálnych systémov. Jednou z pokročilejších logík je napríklad SVO [SO96].

Okrem systémov založených na logikách rôzneho typu existujú aj iné prístupy k formálnemu „dokazovaniu“ bezpečnosti kryptografických protokolov. Významným je napríklad NRL Protocol Analyzer [Mea96], založený na automatickom dokazovaní, že protokol sa nemôže ocitnúť v neželanom stave.

Návrh bezpečného kryptografického protokolu je ťažká úloha. V kapitole venovanej útokom na kryptografické protokoly sú popísané viaceré problémy, na ktoré je potrebné pamätať. Formálne metódy analýzy môžu pri návrhu pomôcť. Úspešná analýza ukazuje bezpečnosť protokolu v rámci modelu danom príslušným formálnym systémom. Na druhej strane analýza nevylučuje existenciu útokov mimo modelu (teda ak sa útočník nebude správať tak, ako systém predpokladá).

Dodatky

A Teória čísel

V tejto časti popíšeme a vysvetlíme vybrané pojmy z teórie čísel, ktoré sú potrebné na pochopenie niektorých kryptografických konštrukcií. Budeme sa v potrebnom rozsahu zaoberať rozšíreným Euklidovým algoritmom, Eulerovou vetou, Čínskou zvyškovou vetou a Jacobiho symbolom.

A.1 Rozšírený Euklidov algoritmus

Rozšírený Euklidov algoritmus pre zadanú dvojicu prirodzených čísel a, b vypočíta ich najväčší spoločný deliteľ (označíme $\mathrm{nsd}(a,b)$) a celé čísla u, v také, že $ub + va = \mathrm{nsd}(a,b)$. Bez ujmy na všeobecnosti budeme predpokladať, že $a \geq b$ (v prípade potreby môžeme argumenty vymeniť).

Rozšírený Euklidov algoritmus

$s_0 \leftarrow a$; $s_1 \leftarrow b$;
$u_0 \leftarrow 0$; $u_1 \leftarrow 1$;
$v_0 \leftarrow 1$; $v_1 \leftarrow 0$;
$n = 1$;
while $s_n > 0$
  $n \leftarrow n + 1$;
  $q_n \leftarrow s_{n-2}/s_{n-1}$; // celočíselné delenie
  $s_n \leftarrow s_{n-2} - q_n \cdot s_{n-1}$;
  $u_n \leftarrow q_n \cdot u_{n-1} + u_{n-2}$;
  $v_n \leftarrow q_n \cdot v_{n-1} + v_{n-2}$;
end while
$u \leftarrow (-1)^n \cdot u_{n-1}$;
$v \leftarrow (-1)^{n-1} \cdot v_{n-1}$;
$\mathrm{nsd}(a,b) \leftarrow s_{n-1}$;

V nasledujúcej vete dokážeme korektnosť tohto algoritmu.

Veta 8. *Nech a, b sú prirodzené čísla, pričom $a \geq b$. Potom*

$$\mathrm{nsd}(a,b) = s_{n-1}; \tag{A.1}$$
$$ub + va = \mathrm{nsd}(a,b). \tag{A.2}$$

Dôkaz. Vlastnosť (A.1) (zodpovedá klasickému Euklidovmu algoritmu) vyplýva z toho, že

$$\begin{aligned}\mathrm{nsd}(a,b) &= \mathrm{nsd}(b, a \bmod b) = \mathrm{nsd}(b, s_2)\\ &= \mathrm{nsd}(s_2, b \bmod s_2) = \mathrm{nsd}(s_2, s_3)\\ &= \quad \cdots\\ &= \mathrm{nsd}(s_{n-2}, s_{n-1}) = s_{n-1}.\end{aligned}$$

V ďalšom sa zameriame na dôkaz vlastnosti (A.2). Najskôr matematickou indukciou ukážeme, že pre všetky $k \in \{0, \ldots, n\}$ platí:

$$(-1)^{k+1} u_k b + (-1)^k v_k a = s_k.$$

1. $k = 0$: $(-1)^1 u_0 b + (-1)^0 v_0 a = a = s_0$
   $k = 1$: $(-1)^2 u_1 b + (-1)^1 v_1 a = b = s_1$
2. Predpokladajme, že identita platí pre $k-1$. Ukážeme platnosť pre k.

$$\begin{aligned}
&(-1)^{k+1} u_k b + (-1)^k v_k a = \\
&\quad = (-1)^{k+1}(q_k u_{k-1} + u_{k-2})b + \\
&\qquad + (-1)^k (q_k v_{k-1} + v_{k-2})a = \\
&\quad = -q_k((-1)^k u_{k-1} b + (-1)^{k-1} v_{k-1} a) \\
&\qquad + (-1)^{k+1} u_{k-2} b + (-1)^k v_{k-2} a = \\
&\quad = -q_k s_{k-1} + (-1)^{k-1} u_{k-2} b + \\
&\qquad + (-1)^{k-2} v_{k-2} a = \\
&\quad = -q_k s_{k-1} + s_{k-2} = s_k
\end{aligned}$$

Po aplikácii práve dokázanej identity a využitím vlastnosti (A.1) dostávame:

$$\underbrace{(-1)^n u_{n-1}}_{u} b + \underbrace{(-1)^{n-1} v_{n-1}}_{v} a = s_{n-1} = \mathrm{nsd}(a,b). \quad \square$$

Poznámka. Uvažujme o celočíselných argumentoch, t.j. $a, b \in \mathbb{Z}$ ($|a| \geq |b|$). Keďže pre ľubovoľnú dvojicu celých čísel x, y platí $\mathrm{nsd}(x,y) = \mathrm{nsd}(-x,y) = \mathrm{nsd}(x,-y) = \mathrm{nsd}(-x,-y)$, stačí brať pre rozšírený Euklidov algoritmus absolútne hodnoty vstupov a, b. Výstupy u a v sa vynásobia -1, ak príslušné b (resp. a) boli záporné.

Príklad. 1. Nech $a = 27$, $b = 21$.

i	s_i	q_i	u_i	v_i
0	27	–	0	1
1	21	–	1	0
2	6	1	1	1
3	3	3	4	3
4	0	2	9	7

Teda $\mathrm{nsd}(a,b) = 3$, $u = (-1)^4 \cdot 4 = 4$, $v = (-1)^3 \cdot 3 = -3$. Skúška: $ub + va = 4 \cdot 21 - 3 \cdot 27 = 3$.

2. Nech $a = -27$, $b = 21$. Počítame s $a' = |a| = 27$ a $b' = |b| = 21$. Potom $u' = 4$, $v' = -3$. Keďže $a < 0$ a $b > 0$, položíme $v = -v' = 3$ a $u = u' = 4$. Potom $ub + va = 4 \cdot 21 + 3 \cdot (-27) = 3 = \mathrm{nsd}(-27, 21)$.
3. Nech $a = 40$, $b = 3$.

i	s_i	q_i	u_i	v_i
0	40	–	0	1
1	3	–	1	0
2	1	13	13	1
3	0	3	40	3

Teda $\mathrm{nsd}(a,b) = 1$, $u = (-1)^3 \cdot 13 = -13$, $v = (-1)^2 \cdot 1 = 1$. Skúška: $ub + va = -13 \cdot 3 + 1 \cdot 40 = 1$.

Zložitosť rozšíreného Euklidovho algoritmu je asymptoticky rovnaká ako zložitosť klasického („deliaceho“) Euklidovho algoritmu. Všimnime si, že ak sa nebudeme

zaoberať číslami u_i a v_i, dostaneme práve klasický algoritmus. Nie je ťažké ukázať, že zložitosť algoritmu je $O(\log^3 a)$, teda kubická vzhľadom na dĺžku zápisu väčšieho parametra. Prirodzene, pri skutočnej implementácii nie je potrebné uchovávať všetky hodnoty s_i, q_i, u_i, v_i. Stačí si pamätať poslednú, resp. posledné dve aktuálne hodnoty. Nasledovný dôsledok je jednoduchou aplikáciou vety 8.

Dôsledok 1. *Nech $a \neq b$ sú navzájom nesúdeliteľné celé čísla, t.j.* $\mathrm{nsd}(a,b) = 1$. *Potom* $\exists u, v \in \mathbb{Z} :\ va + ub = 1$.

Nech $a \neq b$ sú dve nesúdeliteľné prirodzené čísla. Potom podľa dôsledku 1 existujú celé čísla u, v také, že $va + ub = 1$. Teda $va = 1 + b(-v)$, odkiaľ vyplýva:

$$va \equiv 1 \pmod{b}.$$

Rozšírený Euklidov algoritmus teda dokazuje existenciu inverzného prvku k a vzhľadom na násobenie modulo b. Navyše algoritmus poskytuje aj návod, ako tento inverzný prvok (v našom prípade v) nájsť. Poznamenajme, že rovnakými inverznými prvkami sú aj všetky čísla v tvare $v + bt$, kde $t \in \mathbb{Z}$.

A.2 Eulerova veta

Najskôr uvedieme pomocné lemy a definície.

Lema 11. *Nech $n \in \mathbb{N}$ a nech $a, b, k \in \mathbb{Z}$. Ak $ka \equiv kb \pmod{n}$ a $\mathrm{nsd}(k, n) = 1$, tak $a \equiv b \pmod{n}$.*

Dôkaz. Ak $a = b$, tak lema triviálne platí. Bez ujmy na všeobecnosti teda môžeme predpokladať, že $a > b$. Takže existuje $l \in \mathbb{N}$:

$$k(a-b) = ln. \tag{A.3}$$

Keďže $\mathrm{nsd}(k, n) = 1$, tak podľa dôsledku 1 $\exists u, v \in \mathbb{Z} :\ ku + nv = 1$. Z tohto vzťahu vyjadríme k a dosadíme do (A.3):

$$\begin{aligned} k(a-b) = \frac{1-nv}{u} \cdot (a-b) &= ln \\ a - b &= lnu + nv(a-b) \\ a - b &= n(lu + v(a-b)) \end{aligned}$$

A teda $a \equiv b \pmod{n}$. □

Definícia 14. Pre ľubovoľné prirodzené číslo n označíme symbolom $\mathbb{Z}_n^*$ množinu všetkých čísel nesúdeliteľných s n, ktoré sú menšie ako n a väčšie ako 0:

$$\mathbb{Z}_n^* = \{a \mid a \in \{1, \ldots, n-1\} \ \&\ \mathrm{nsd}(a, n) = 1\}.$$

Zároveň symbolom $\phi(n)$ budeme označovať mohutnosť množiny $\mathbb{Z}_n^*$: $\phi(n) = |\mathbb{Z}_n^*|$. Funkcia $\phi(n)$ sa nazýva Eulerova funkcia.

Príklad. 1. $\mathbb{Z}_8^* = \{1, 3, 5, 7\}$, $\phi(8) = 4$;

2. Ak p je prvočíslo, tak $\mathbb{Z}_p^* = \{1, \ldots, p-1\}$;
3. Ak $n = p \cdot q$ je súčin dvoch prvočísel, tak $\phi(n) = (p-1)(q-1)$.

Lema 12. *Nech $\mathbb{Z}_n^* = \{r_1, \ldots, r_{\phi(n)}\}$ sú všetky prirodzené čísla menšie ako n a nesúdeliteľné s n. Nech a je celé číslo a $\mathrm{nsd}(a, n) = 1$. Potom $\{ar_1 \bmod n, \ldots, ar_{\phi(n)} \bmod n\} = \mathbb{Z}_n^*$.*

Dôkaz. Potrebujeme ukázať, že čísla $ar_1 \bmod n, \ldots, ar_{\phi(n)} \bmod n$ sú navzájom rôzne a nesúdeliteľné s n. Ľahko vidieť, že $0 < ar_i \bmod n < n$ pre všetky $i = 1, \ldots, \phi(n)$.

1. Nech $i, j \in \{1, \ldots, \phi(n)\}$ sú také indexy, že $ar_i \bmod n = ar_j \bmod n$. Keďže $\mathrm{nsd}(a, n) = 1$, tak podľa lemy 11 platí $r_i \equiv r_j \pmod{n}$. Podľa predpokladu $r_i, r_j < n$, a preto $r_i = r_j \Rightarrow i = j$. Teda v postupnosti $ar_1 \bmod n, \ldots, ar_{\phi(n)} \bmod n$ nie sú rovnaké prvky.
2. Pre všetky $i \in \{1, \ldots, \phi(n)\}$: $\mathrm{nsd}(r_i, n) = 1$. Rovnako $\mathrm{nsd}(a, n) = 1$. Preto aj $\mathrm{nsd}(ar_i \bmod n, n) = 1$.

□

Veta 9 (Eulerova veta). *Nech $n \in \mathbb{N}$, $a \in \mathbb{Z}$ a nech $\mathrm{nsd}(a, n) = 1$. Potom $a^{\phi(n)} \equiv 1 \pmod{n}$.*

Dôkaz. Vezmime $\mathbb{Z}_n^* = \{r_1, \ldots, r_{\phi(n)}\}$. Potom platí:

$$\prod_{i=1}^{\phi(n)} r_i \equiv \prod_{i=1}^{\phi(n)} ar_i \equiv a^{\phi(n)} \prod_{i=1}^{\phi(n)} r_i \pmod{n}.$$

Pripomeňme, že prvá kongruencia platí podľa lemy 12. Keďže $\mathrm{nsd}(\prod_{i=1}^{\phi(n)} r_i, n) = 1$, podľa lemy 11 dostávame:

$$a^{\phi(n)} \equiv 1 \pmod{n}.$$

□

Dôsledok 2 (Malá Fermatova veta). *Nech p je prvočíslo a nech $a \in \mathbb{Z}$ je také, že $p \nmid a$ (p nedelí a). Potom $a^{p-1} \equiv 1 \pmod{p}$.*

Dôkaz. Stačí si uvedomiť, že $\phi(p) = p - 1$. □

A.3 Čínska zvyšková veta

Veta 10 (Čínska zvyšková veta). *Nech $m_1, \ldots, m_k$ sú po dvoch nesúdeliteľné prirodzené čísla, t.j.*

$$\forall\, i, j\ (1 \leq i < j \leq k) : \mathrm{nsd}(m_i, m_j) = 1.$$

Nech $a_1, \ldots, a_k \in \mathbb{Z}$. Potom sústava kongruencií

$$\begin{aligned} x &\equiv a_1 \pmod{m_1} \\ x &\equiv a_2 \pmod{m_2} \\ &\ldots \\ x &\equiv a_k \pmod{m_k} \end{aligned}$$

má riešenie. Navyše, všetky riešenia tejto sústavy kongruencií sú navzájom kongruentné modulo $M = m_1 \cdot m_2 \cdot \ldots \cdot m_k$.

Dôkaz. Hľadané riešenie je

$$x = \sum_{i=1}^{k} a_i M_i N_i,$$

kde $M_i = \frac{M}{m_i}$ je súčin všetkých modulov okrem m_i a N_i je inverzný prvok k M_i vzhľadom na násobenie modulo m_i, t.j. $M_i N_i \equiv 1 \pmod{m_i}$. Existencia N_i je zaručená, lebo $\text{nsd}(M_i, m_i) = 1$ a možno ho vypočítať pomocou rozšíreného Euklidovho algoritmu (pozri časť A.1). Overme, že x je naozaj riešením sústavy kongruencií. Pre ľubovoľné r $(1 \leq r \leq k)$:

$$\begin{aligned} x \bmod m_r &= \sum_{i=1}^{k} a_i M_i N_i \bmod m_r \\ &\overset{(*)}{=} a_r M_r N_r \bmod m_r = a_r \bmod m_r, \end{aligned}$$

teda $x \equiv a_r \pmod{m_r}$. Rovnosť $(*)$ vyplýva z toho, že $m_r \mid M_i$, pre všetky $i \neq r$.

Ukážme ešte druhú časť tvrdenia. Nech x' a x'' sú dve riešenia sústavy. Nech $z = x' - x''$. Potom $m_i \mid z$, pre všetky $i = 1, \ldots, k$. Odtiaľ $M \mid z$ a teda $x' \equiv x'' \pmod{M}$. □

Príklad. Uvažujme sústavu kongruencií:

$$\begin{aligned} x &\equiv 3 \pmod{6}, \\ x &\equiv 2 \pmod{7}. \end{aligned}$$

Riešením je $x = 3 \cdot 7 \cdot 1 + 2 \cdot 6 \cdot 6 = 93$. Najmenšie prirodzené číslo, ktoré je riešením sústavy je $93 \bmod (6 \cdot 7) = 9$.

Čínska zvyšková veta hovorí, že ľubovoľné číslo z množiny $\{0, 1, \ldots, M\}$ (t.j. zvyškovú triedu modulo M) možno jednoznačným spôsobom reprezentovať pomocou zvyškových tried modulo $m_1, \ldots, m_k$. Odtiaľ vyplýva aj nasledovný dôsledok.

Dôsledok 3. *Nech p a q sú navzájom nesúdeliteľné prirodzené čísla. Potom pre ľubovoľné $x \in \mathbb{Z}$ platí:*

$$x \equiv a \pmod{p\,q} \iff \begin{aligned} x &\equiv a \pmod{p} \\ x &\equiv a \pmod{q}. \end{aligned}$$

Poznámka. Dôsledok môžeme ľahko zovšeobecniť aj pre väčší počet navzájom po dvoch nesúdeliteľných čísel (modulov).

A.4 Jacobiho symbol***

B Teória pravdepodobnosti***

V tejto časti uvádzame niektoré tvrdenia z teórie pravdepodobnosti, ktoré sú potrebné pre výklad v predchádzajúcich kapitolách. Podrobnejší výklad možno nájsť v špecializovaných knihách a učebniciach. V diskusii používame tieto označenia (poznamenajme, že sa zaoberáme výlučne diskrétnymi náhodnými premennými, ktoré nadobúdajú konečný počet hodnôt):

$E(X)$ stredná hodnota náhodnej premennej X, teda $E(X) = \sum_x \Pr[X = x] \cdot x$;
$D(X)$ disperzia náhodnej premennej X, teda $D(X) = E((X - E(X))^2)$ alebo inak: $D(X) = E(X^2) - E(X)^2$;

B.1 Narodeninový paradox

B.2 Markovova a Čebyševova nerovnosť

Markovova a Čebyševova nerovnosť platia pre všetky nezáporné náhodné premenné, bez ohľadu na ich distribúciu. Poskytujú základné prostriedky pre konštrukciu odhadov náhodných premenných.

Veta 11 (Markovova nerovnosť). *Nech X je náhodná premenná nadobúdajúca nezáporné hodnoty. Nech $v > 0$ je reálne číslo. Potom*

$$\Pr[X \geq v] \leq \frac{E(x)}{v}.$$

Dôkaz. Počítajme:

$$\begin{aligned} E(X) &= \sum_x \Pr[X = x] \cdot x \\ &\geq \sum_{x<v} \Pr[X = x] \cdot 0 + \sum_{x \geq v} \Pr[X = x] \cdot v \\ &= \Pr[X \geq v] \cdot v. \end{aligned}$$

Požadovanú nerovnosť dostaneme po vydelení nerovnosti hodnotou v. □

Nerovnosť ľahko upravíme na odhad strednej hodnoty, stačí za v dosadiť výraz $v \cdot E(X)$. Dostaneme nasledujúci dôsledok:

Dôsledok 4. *Nech X je náhodná premenná nadobúdajúca nezáporné hodnoty. Nech $v > 0$ je reálne číslo. Potom*

$$\Pr[X \geq v \cdot E(X)] \leq \frac{1}{v}.$$

Čebyševova nerovnosť vylepšuje odhad strednej hodnoty tým, že berie do úvahy disperziu náhodnej premennej.

Veta 12 (Čebyševova nerovnosť). *Nech X je náhodná premenná nadobúdajúca nezáporné hodnoty. Nech $v > 0$ je reálne číslo. Potom*

$$\Pr[|X - E(X)| \leq v] \geq \frac{D(X)}{v^2}.$$

Dôkaz. Vieme, že $D(X) = E((X - E(X))^2)$. Položme $Y = (X - E(X))^2$ a aplikujme na túto náhodnú premennú Markovovu nerovnosť:

$$\Pr[Y \geq t] \leq \frac{E(Y)}{t} = \frac{D(X)}{t},$$

pre ľubovoľné $t > 0$. Položme $v = \sqrt{t}$ a počítajme:

$$\Pr[Y \geq t] = \Pr[(X - E(X))^2 \geq v^2] = \Pr[|X - E(X)| \geq v],$$

Teda dostaneme:

$$\Pr[|X - E(X)| \geq v] \leq \frac{D(X)}{v^2}.$$

□

Príklad. Háďžme ideálnou mincou, teda pravdepodobnosť, že padne „hlava“ je pri každom hode $1/2$. Nech náhodná premenná X vyjadruje počet hláv v n po sebe nasledujúcich hodoch. To znamená, že X nadobúda hodnotu $k \in \{0, \ldots, n\}$ s pravdepodobnosťou

$$\Pr[X = k] = \binom{n}{k} \left(\frac{1}{2}\right)^n,$$

teda X má binomické rozdelenie pravdepodobnosti. Preto je stredná hodnota $E(X) = n/2$ a disperzia $D(X) = n/4$. Pomocou Čebyševovej nerovnosti odhadnime zhora pravdepodobnosť toho, že počet hláv bude najviac $n/3$ alebo aspoň $2n/3$. Pre hľadanú pravdepodobnosť dostaneme:

$$\Pr\left[\left|X - \frac{n}{2}\right| \geq \frac{n}{6}\right] \leq \frac{D(X)}{(n/6)^2} = \frac{n/4}{n^2/36} = \frac{9}{n}.$$

B.3 Chernoffova nerovnosť

Chernoffove nerovnosti umožňujú získať presnejšie odhady ako Čebyševova nerovnosť pre niektoré typy náhodných premenných. Môžeme ich uplatniť vtedy, ak sa náhodná premenná X dá vyjadriť ako súčet nezávislých náhodných premenných. Pre naše potreby sa obmedzíme na rovnako distribuované náhodné premenné nadobúdajúce hodnoty z množiny $\{0, 1\}$.

Veta 13 (Chernoffova nerovnosť). *Nech $X_1, \ldots, X_n$ sú nezávislé náhodné premenné nadobúdajúce hodnotu 0 s pravdepodobnosťou p a hodnotu 1 s pravdepodobnosťou $1-p$. Nech $X = \sum_{i=1}^{n} X_i$. Potom pre ľubovoľné $\varepsilon > 0$ platí:*

$$\Pr[X \geq (1+\varepsilon)E(X)] \leq e^{np(\varepsilon-(1+\varepsilon)\ln(1+\varepsilon))}.$$

Dôkaz. Náhodná premenná X má binomické rozdelenie, preto $E(X) = np$. Počítajme hľadanú pravdepodobnosť, pričom zavedieme dodatočný parameter $t > 0$:

$$\Pr[X \geq (1+\varepsilon)np)] = \Pr\left[e^{tX} \geq e^{t(1+\varepsilon)np}\right] \leq \frac{E(e^{tX})}{e^{t(1+\varepsilon)np}}.$$

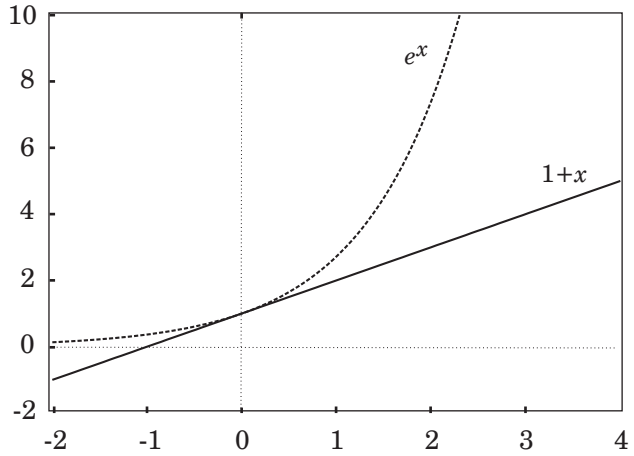
Prvá rovnosť platí pre ľubovoľné $t > 0$ a nerovnosť je aplikáciou Markovovej nerovnosti. Strednú hodnotu v čitateli vieme počítať vďaka nezávislosti náhodných premenných $X_1, \ldots, X_n$:

$$\begin{aligned} E(e^{tX}) &= E(e^{tX_1} \cdot e^{tX_2} \cdot \ldots \cdot e^{tX_n}) \\ &= E(e^{tX_1}) \cdot E(e^{tX_2}) \cdot \ldots \cdot E(e^{tX_n}) \\ &= E(e^{tX_1})^n. \end{aligned}$$

Posledná rovnosť platí preto, lebo $X_1, \ldots, X_n$ sú rovnako distribuované. Pokračujme ďalej:

$$E(e^{tX_1}) = (p \cdot e^t + (1-p) \cdot e^0)^n = (1 + p \cdot (e^t - 1)) \leq e^{p(e^t-1)}.$$

Pri úprave sme využili nerovnosť $1 + x \leq e^x$, pre všetky $x \in \mathbb{R}$. Platnosť tejto nerovnosti ilustruje aj obrázok B.1.



Obrázok B.1: Nerovnosť $1 + x \leq e^x$, pre $x \in \mathbb{R}$

Dosadením dostaneme:

$$E(e^{tX}) \leq e^{np(e^t-1)}.$$

Preto:

$$\Pr[X \geq (1+\varepsilon)np] = \Pr\left[e^{tX} \geq e^{t(1+\varepsilon)np}\right] \leq \frac{e^{np(e^t-1)}}{e^{t(1+\varepsilon)np}} = e^{np(e^t-1-t(1+\varepsilon))}.$$

Hľadáme pokiaľ možno najlepší odhad, preto minimalizujme výraz v exponente. Položme $f(t) = e^t - 1 - t(1+\varepsilon)$. Riešením rovnice $f'(t) = 0$ získame hodnotu $t = \ln(1+\varepsilon)$ a tým aj požadovaný odhad:

$$\Pr[X \geq (1+\varepsilon)np] \leq e^{np(\varepsilon-(1+\varepsilon)\ln(1+\varepsilon))}.$$

□

Príklad. Pomocou Chernoffovej nerovnosti odhadnime pravdepodobnosť toho, že pri n hodoch ideálnou mincou padne hlava aspoň $2n/3$ krát (podobný príklad sme riešili v časti B.2). Nech náhodná premenná X_i, pre $i = 1, \ldots, n$, nadobúda hodnoty 1 a 0, podľa toho, či v i-tom hode padne hlava alebo nie. Náhodné premenné $X_1, \ldots, X_n$ sú nezávislé. Označme $X = \sum_{i=1}^{n} X_i$ počet hláv v n hodoch. Podmienky Chernoffovej nerovnosti sú splnené, teda:

$$\Pr\left[X \geq \left(1+\frac{1}{3}\right)\cdot\frac{n}{2}\right] = \Pr\left[X \geq \frac{2n}{3}\right] \leq e^{n/2\cdot(1/3-(4/3)\ln(4/3))} \leq e^{-0{,}025\cdot n}.$$

Pokiaľ chceme získať odhad pravdepodobnosti pre rovnakú udalosť ako v príklade z časti B.2, môžeme využiť symetrickosť binomického rozdelenia náhodnej premennej X:

$$\Pr\left[\left|X - \frac{n}{2}\right| \geq \frac{n}{6}\right] \leq 2\cdot e^{-0{,}025\cdot n}.$$

Literatúra

[AES01] *Advanced Encryption Standard (AES)*, FIPS PUB 197, NIST, 2001.

[AKS02] Agrawal, M. – Kayal, N. – Saxena, N.: *PRIMES is in P*, Technical report, 2002.

[AN94] Abadi, M. – Needham, R.: *Prudent engineering practice for cryptographic protocols*, SRC Research Report 125, Digital System Research Centre, 1994.

[AN95] Anderson, R. – Needham, R.: Robustness priciples for public key protocols, *Advances in Cryptology – CRYPTO'95*, LNCS 963, 236–247, Springer-Verlag, 1995.

[BCK96] Bellare, M. – Canetti, R. – Krawczyk, H.: Keying hash functions for message authentication, *Advances in Cryptology – CRYPTO'96*, LNCS ???, ???–???, Springer-Verlag, 1996.

[BD99] Boneh, D. – Durfee, G.: Cryptanalysis of RSA with private key d less than $N^{0.292}$, *Advances in Cryptology – Eurocrypt'99*, LNCS 1592, ???–???, Springer-Verlag, 1999.

[Ble96] Bleichenbacher, D.: Generating ElGamal signatures without knowing the secret key, *Advances in Cryptology – Eurocrypt'96*, LNCS 1070, 11–18, Springer-Verlag, 1996.

[BV98] Boneh, D. – Venkatesan, R.: Breaking RSA may not be equivalent to factoring, *Advances in Cryptology – Eurocrypt'98*, LNCS 1403, ???–???, Springer-Verlag, 1998.

[CFPR96] Coppersmith, D. – Franklin, M. – Patarin, J. – Reiter, M.: Low-Exponent RSA with Related Messages, *Advances in Cryptology – Eurocrypt'96*, LNCS 1070, 1–9, Springer-Verlag, 1996.

[Cop96] Coppersmith, D.: Finding a Small Root of a Univariate Modular Equation, *Advances in Cryptology – Eurocrypt'96*, LNCS 1070, 155–165, Springer-Verlag, 1996.

[DH76] Diffie, W. – Hellman, M.: New directions in cryptography, *IEEE Transactions on Information Theory*, IT-22, 1976.

[DSS01] *Digital Signature Standard (DSS)*, FIPS PUB 186-2 (Change Notice 1), NIST, 2001.

[FS97] Fischlin, R. – Schnorr, C.: Stronger security proofs RSA and Rabin Bits, *Advances in Cryptology – Eurocrypt'97*, LNCS 1233, ???–???, Springer-Verlag, 1997.

[GKP94] Graham, R. – Knuth, D. – Patashnik, O.: *Concrete Mathematics: A Foundation for Computer Science*, Addison-Wesley, 2nd edition, 1994.

[Gor92] Gordon, D.: Designing and detecting trapdoors for discrete log cryptosystems, *Advances in Cryptology – CRYPTO'92*, LNCS 740, 66–75, Springer-Verlag, 1992.

[Has88] Hastad, J.: Solving simultaneous modular equations of low degree, *SIAM Journal on Computing*, 17(2), 336–341, 1988.

[Kob87] Koblitz, N.: *A Course in Number Theory and Cryptography*, Springer-Verlag, 1987.

[KSW98] Kelsey, J. – Schneier, B. – Wagner, D.: Protocol interactions and the chosen protocol attack, *Security Protocols, 5th International Workshop April 1997 Proceedings*, 91–104, Springer-Verlag, 1998.

[LKBS92] Loxton, J. – Khoo, D. – Bird, G. – Seberry, J.: A cubic RSA code equivalent to factorization, *Journal of Cryptology*, 5, 1992.

[Low95] Lowe, G.: An attack on the needham-schroeder public key authentication protocol, *Information Processing Letters*, 3(56), 131–136, 1995.

[Mau94] Maurer: aaaaa, 1994.

[Mea96] Meadows, C.: The NRL Protocol Analyzer: An overview, *Journal of Logic Programming*, 2(26), 1996.

[Riv98] Rivest, R. L.: Chaffing and winnowing: Confidentiality without encryption, 1998.

[SHS02] *Secure Hash Standard (SHA)*, FIPS PUB 180-2, NIST, 2002.

[SO96] Syverson, P. F. – van Oorschot, P. C.: *A Unified Cryptographic Protocol Logic*, NRL Publication 5540-227, Naval Research Lab, 1996.

[Sti95] Stinson, R.: *Cryptography: Theory and Practice*, CRC Press, 1995.

[Syv94] Syverson, P.: A taxonomy of replay attacks, 131–136, IEEE Computer Society Press, 1994.

[Syv96] Syverson, P.: Limitations on design principles for public key protocols, *IEEE Symposium on Security and Privacy*, 62–73, IEEE Computer Society Press, Oakland, CA, 1996.

[Wil80] Williams, H.: A modification of the RSA public-key procedure, *IEEE Transactions on Information Theory*, IT-26, 726–729, 1980.