Криптографија

Миодраг Живковић

6 априља 2017 г.

Садржај

Основа за овај курс је текст лекција E. Schaefera (универзитет Санта Клара у Калифорнији) An introduction to cryptography, http://math.scu.edu/~eschaefe/crylec.pdf. Извршена су само минимална прилагођавања, пре свега језичка. Потребно предзнање је минимално.

Алиса жели да пошаље поруку Бобану, али тако да избегне да радознала Цица (противник, непријатељ, која може да снима туђе поруке) може да разуме ту поруку. Због тога Алиса шифрује поруку пре слања. Бобан дешифрује примљену поруку. У току првог дела курса (*криптографија*) анализираћемо ове две активности, *шифровање* и *дешифровање*. Ако Цица пресретне поруку, онда ће она покушати да *разбије шифру* и декриптира (прочита) поруку; у другом делу курса бавићемо се алгоритмима повезаним са разбијањем шифри, односно *криптоанализом*.

Литература:

Menezes, Oorshot, Vanstone, Handbook of Applied Cryptography
B. Schneier, Applied Cryptography, II izdanje, 1996.
D. Stinson, Cryptography - Theory And Practice, III izd, CRC 2006.
N. Koblitz, A course in number theory and cryptography, Springer, 1987.
A. G. Konheim, Computer Security and Cryptography, Wiley, 2007.
D. Kahn, The CodeBreakers, 1973.
А. Трифони: Шифре и прислушкивање 1-3, Плато 2002.

1 Увод

После прегледа основних појмова из криптологије, неколико примера из историје криптографије и основних појмова из теорије бројева, у овом курсу разматраћемо неколико једноставних шифарских система, примере криптоанализе и неке савремене шифарске системе.

2 Основни појмови

Отворени текст, OT је порука коју треба послати, нпр. ZDRAVO.
Шифрат, ST је шифрована порука, нпр. XQABER.
Шифровање је трансформација отвореног текста у шифрат.
Дешифровање је инверзна трансформација шифрата у отворени текст.
Кодирање трансформише отворени текст у низ цифара или бита. На пример, ако велика слова (енглеске) абецеде кодирамо са A$\rightarrow$ 0, ... Z$\rightarrow$ 25, онда се реч ZDRAVO кодира са 25 3 17 0 21 14. У пракси се често користи ASCII код, који сваку симбол представља са 8 бита, нпр. A $\rightarrow$ 01000001, B $\rightarrow$ 01000010, a $\rightarrow$ 01100001, 0 $\rightarrow$ 00110000, ? $\rightarrow$ 00111111, и сл.
Декодирање трансформише низ цифара или бита у полазни текст. У кодирању и декодирању нема ничега тајног.
Проточна шифра трансформише отворени текст симбол по симбол, односно најчешће бит по бит.

Блоковска шифра примењује се на симболе отвореног текста груписане у блокове. То могу да буду нпр. **биграми** — парови слова (као што је TI), односно **триграми** — тројке слова. AES (скраћеница од Advanced Encryption Standard) ради са блоковима од по 128 бита (односно 16 знакова). Ако алгоритам шифровања ради са биграмима, онда он може (не мора) да увек TI замењује нпр. са AG, односно TE са LK.
Шифра премештања премешта (пермутује) слова (знакове, бите).
Шифра замене замењује слова (знакове, бите) другим, не мењајући им редослед.
Комбинована шифра примењује наизменично премештања и замене.

Подела на проточне и блоковске шифре је условна.

Шифарски систем (или само систем) је пар чији су елементи алгоритам шифровања и алгоритам дешифровања. Ови алгоритми скоро увек зависе од посебног параметра, који се зове **кључ**. Кључ је параметар којим се бира конкретна шифарска трансформација у оквиру изабраног система.
Симетрични систем подразумева употребу истог тајног кључа за шифровање и дешифровање. Алиса и Бобан морају унапред да се договоре који кључ ће користити.
Асиметрични систем (систем са јавним кључем) подразумева да Алиса и Бобан објаве (публикују) своје кључеве за шифровање, а да у највећој тајности чувају своје кључеве за дешифровање.
Криптоанализа је процес помоћу кога Цица покушава да шифрат трансформише у одговарајући отворени текст, не знајући кључ.
Декриптирање је (макар и делимично) успешна криптоанализа.

Алиса је пошиљалац шифроване поруке. **Бобан** је њен прималац. **Цица** прислушкује канал везе и покушава да прочита шифровану поруку.

Уобичајене претпоставке:

1. Шифровање и дешифровање треба да буду једноставни за регуларне учеснике, Алису и Бобана. Декриптирање треба да буде тежак проблем.

2. Сигурност и практичност шифарског система су скоро увек противречни захтеви.

3. Претпоставља се да Цица зна детаље примењеног шифарског система, а да не зна само кључ.

3 Историја

Спартанска шифра Скитале (400 година п.н.е.) Ово је пример шифре премештања. Слова поруке исписују се на дугачкој папирној траци, која се омотава око штапа. Дијаметар штапа је кључ за шифровање.

```
----------------------------
  /T/P/A/K/A/ /            / \
 /C/E/ /O/M/O/            |   |
/T/A/B/A/ / /              \ /
----------------------------
```

Цезарова шифра. Шифра замене, која свако слово замењује трећим словом удесно (дуж абецеде) од њега. Ако се ради о енглеској абецеди, онда су замене $A \to D$, $B \to E$, $\ldots Z \to C$. Шифрат HAL одговара поруци IBM ("Одисеја у свемиру 2001") (овде се врши померање за једно слово удесно).

Плејферова шифра. Примењивана је око 1910-20. године у Бурском рату, првом светском рату. То је једна од првих шифара која је обрађивала биграме. То је истовремено и шифра замене. На пример, за кључ PALMERSTON формира се табела

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

Да би се шифровао пар SF посматра се правоугаоник коме су темена ова два слова; остала два темена правоугаоника су шифрат OC. Редослед је одређен чињеницом да су S и O у истој врсти као F и C. Ако су два слова отвореног текста у истој врсти, онда се свако слово замењује словом са његове десне стране. Тако SO постаје TN, а BG постаје CB. Ако су два слова отвореног текста у истој колони, онда се свако слово замењује словом испод себе. Тако IS постаје WC, а SJ постаје CW. Двострука слова раздвајају се словом X, па се тако отворени текст BALLOON трансформише у BA LX LO ON пре шифровања. Свако слово J у тексту замењује се словом I (тако се број различитих слова смањује на 25).

ADFGVX. Ову шифру користили су Немци у првом светском рату. Заснива се на примени фиксне табеле

	A	D	F	G	V	X
A	K	Z	W	R	1	F
D	9	B	6	C	L	5
F	Q	7	J	P	G	X
G	E	V	Y	3	A	N
V	8	O	D	H	0	2
X	U	4	I	S	T	M

Наредно слово које треба шифровати се проналази у табели, па се замењује паром ознака (врста, колона). Тако отворени текст PRODUCTCIPHERS

постаје FG AG VD VF XA DG XV DG XF FG VG GA AG XG. Ово је фаза замене. Затим следи фаза премештања, која зависи од кључа без поновљених слова. Нека је то на пример DEUTCH. Слова се нумеришу према месту у абецеди. Резултат прве фазе напише се испод кључа у више врста.

```
D  E  U  T  S  C  H
2  3  7  6  5  1  4
F  G  A  G  V  D  V
F  X  A  D  G  X  V
D  G  X  F  F  G  V
G  G  A  A  G  X  G
```

Слова се затим исписују редом по колонама, при чему бројеви изнад колона треба да чине растући низ. У овом примеру шифрат је DXGX FFDG GXGG VVVG VGFG GDFA AAXA (размаци се игноришу).

За време другог светског рата Шенон (Shannon) показао је да наизменично коришћење замена и премештања даје добре шифарске системе. Систем ADFGVX је лош, јер се користи само по једна замена и премештање, при чему је замена фиксна (не зависи од кључа). За време другог светског рата коришћене су компликоване комбиноване шифре, као што су немачка ENIGMA и јапанска PURPLE, а направљени су и рачунари (Colossus) за разбијање тих шифри. Још увек је тајна које шифре су користили Американци.

После 1970. године угрожавање безбедности рачунара постало је озбиљан проблем. Појавила се потреба за сигурнијим шифрама за комерцијалну употребу. Постало је могуће реализовати сложене алгоритме у једном чипу, што је омогућавало брзо шифровање. Порасле су и могућности криптоанализе.

Проблем је интензивно анализиран између 1968. и 1976. године. Године 1974. појавила се шифра LUCIFER (IBM), а 1975. DES (скраћеница од Data Encryption Standard). Оба ова система су комбиноване шифре. DES користи кључ од 56 бита. У њему се наизменично користе 16 замена и 15 премештања. AES користи кључ од 128 бита, а састоји се од 10 замена и 10 премештања. Године 1975. појављују се системи са јавним кључем.

4 Преглед основа теорије бројева

Нека $\mathbf{Z}$ означава скуп целих бројева, $\mathbf{Z} = \{0, \pm 1, \pm 2, \ldots\}$. За целе бројеве $a, b \in \mathbf{Z}$ кажемо да a дели b (ознака $a|b$) ако је $b = na$ за неко $n \in \mathbf{Z}$. Број a дели b ако и само ако је b умножак a. Тако $3|12$ јер је $12 = 4 \cdot 3$, $3|3$ јер је $3 = 1 \cdot 3$, $5|-5$ јер је $-5 = -1 \cdot 5$, $6|0$ јер је $0 = 0 \cdot 6$. Ако $x|1$, колико је x? (Одговор: ± 1).

Особине оператора $|$:
Ако $a, b, c \in \mathbf{Z}$ и $a|b$, онда $a|bc$. Тако, из $3|12$ следи $3|60$.
Ако $a|b$ и $a|c$, онда $a|b \pm c$.
Ако $a|b$ и $a \not| c$, онда $a \not| b \pm c$.

Прости бројеви су $2, 3, 5, 7, 11, 13, \ldots$.

Основна теорема аритметике: Сваки број $n \in \mathbf{Z}$, $n > 1$, може се на јединствени начин представити у облику производа простих бројева $n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_r^{\alpha_r}$, где су α_i позитивни цели бројеви. Сви позитивни делиоци броја n су облика $p_1^{\beta_1} p_2^{\beta_2} \cdots p_r^{\beta_r}$, где је $0 \leq \beta_i \leq \alpha_i$ за свако $i = 1, 2, \ldots, r$, па n има $(\alpha_1 + 1)(\alpha_2 + 1) \cdots (\alpha_r + 1)$ различитих делилаца.

На пример, $90 = 2^1 \cdot 3^2 \cdot 5^1$. Да бисмо пронашли све позитивне делиоце броја 90,

$$\begin{array}{ccc} 1 & 1 & 1 \\ 2 & 3 & 5 \\ & 9 & \end{array}$$

треба да прођемо све могуће путеве слева удесно: $1 \cdot 1 \cdot 1 = 1$, $1 \cdot 1 \cdot 5 = 5$, $1 \cdot 3 \cdot 1 = 3$, $1 \cdot 3 \cdot 5 = 15$, $1 \cdot 9 \cdot 1 = 9$, $1 \cdot 9 \cdot 5 = 45$, $2 \cdot 1 \cdot 1 = 2$, $2 \cdot 1 \cdot 5 = 10$, $2 \cdot 3 \cdot 1 = 6$, $2 \cdot 3 \cdot 5 = 30$, $2 \cdot 9 \cdot 1 = 18$, $2 \cdot 9 \cdot 5 = 90$.

Последица основне теореме аритметике: ако је p прост број и $p|ab$, онда $p|a$ или $p|b$. На пример, пошто $3|180 = 4 \cdot 45$, онда $3|4$ или $3|45$ (у овом случају тачно је друго тврђење). Импликација није увек тачна за сложене бројеве: $4|6 \cdot 2$, иако $4 \not| 6$ и $4 \not| 2$.

Нека су $a, b \in \mathbf{Z}_{\geq 0}$ позитивни цели бројеви, од којих један може бити 0. *Највећи заједнички делилац* (НЗД) a и b (ознака $\mathrm{nzd}(a, b)$) је највећи цели број d који дели и a и b. Приметимо да ако $d|a$ и $d|b$, онда $d|\mathrm{nzd}(a, b)$. На пример, $\mathrm{nzd}(12, 18) = 6$, $\mathrm{nzd}(12, 19) = 1$. Највећи заједнички делилац се користи за скраћивање (свођење) разломака; разломак $12/19$ је сведен.

Ако знамо факторизације бројева $a = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_r^{\alpha_r}$ и $b = p_1^{\beta_1} p_2^{\beta_2} \cdots p_r^{\beta_r}$ (неки од експонената могу бити 0), онда је $\mathrm{nzd}(a, b) = p_1^{\gamma_1} p_2^{\gamma_2} \cdots p_r^{\gamma_r}$, где је $\gamma_i = \min\{\alpha_i, \beta_i\}$. На пример, $2520 = 2^3 \cdot 3^2 \cdot 5^1 \cdot 7^1$ и и $2700 = 2^2 \cdot 3^3 \cdot 5^2 \cdot 7^0$, па је $\mathrm{nzd}(2520, 2700) = 2^2 \cdot 3^2 \cdot 5^1 \cdot 7^0 = 180$. Приметимо да је $2520/180 = 14$, $2700/180 = 15$ и $\mathrm{nzd}(14, 15) = 1$. За бројеве којима је НЗД једнак 1 кажемо да су *узајмно прости*.

Нека $a \mod b$ означава остатак при дељењу a са b; ако је $a = qb + r$, $0 \leq r < b$, онда је $a \mod b = r$. На пример, $12 \mod 5 = 2$, $7 \mod 5 = 2$. Растављање великих бројева на просте чиниоце је тежак проблем. Уместо помоћу растављања на чиниоце, једноставан и ефикасан начин за израчунавање НЗД је примена Еуклидовог алгоритма. Користи се чињеница да је $\mathrm{nzd}(a, b) = \mathrm{nzd}(a \mod b, b)$. Одредимо нпр. $\mathrm{nzd}(329, 119)$. Најпре делимо 329 са 119; добијамо количник 2 и остатак 91. У сваком наредном кораку делилац и остатак постају наредни дељеник и делилац:

$$\begin{aligned} 329 &= 2 \cdot \underline{119} + \underline{91} \\ 119 &= 1 \cdot \underline{91} + \underline{28} \\ 91 &= 3 \cdot \underline{28} + \underline{7} \\ 28 &= 4 \cdot \underline{7} + \underline{0} \end{aligned}$$

Последњи остатак различит од 0 је тражени НЗД. Овде је $\mathrm{nzd}(329, 119) = 7$.

Низ дељења који чини Еуклидова алгоритам може се искористити да се $\mathrm{nzd}(a,b)$ представи у облику целобројне линеарне комбинације $\mathrm{nzd}(a,b) = na + mb$, за неке $m, n \in \mathbf{Z}$. У сваком кораку се замењује мањи подвучени број (члан низа остатака):

$$\begin{array}{rcll} 7 & = & \underline{91} - 3 \cdot \underline{28} & \text{заменити мањи} \\ & = & \underline{91} - 3(\underline{119} - 1 \cdot \underline{91}) & \text{упростити} \\ & = & 4 \cdot \underline{91} - 3 \cdot \underline{119} & \text{заменити мањи} \\ & = & 4(\underline{329} - 2 \cdot \underline{119}) - 3 \cdot \underline{119} & \text{упростити} \\ 7 & = & 4 \cdot \underline{329} - 11 \cdot \underline{119} & \end{array}$$

Према томе, $7 = 4 \cdot 329 - 11 \cdot 119$, односно $n = 4$ и $m = -11$.

Релација **mod**. Поред тога што mod означава бинарну операцију (остатак при целобројном дељењу), mod је и ознака релације у $\mathbf{Z}$: пишемо $a \equiv b \pmod{m}$ ако $m|b-a$. Другим речима, разлика a и b је умножак m. Тако је $7 \equiv 2 \pmod 5$, јер $5|5$, $2 \equiv 7 \pmod 5$, јер $5|-5$, $12 \equiv 7 \pmod 5$, јер $5|5$, $12 \equiv 2 \pmod 5$, јер $5|10$, $7 \equiv 7 \pmod 5$, јер $5|0$, $-3 \equiv 7 \pmod 5$, јер $5|-10$. За $k = 0$ ($k = 1, 2, 3, 4$) бројеви $k, k \pm 5, k \pm 10, \ldots$ су сви међусобно конгруентни по модулу 5.

У неким ситуацијама уобичајено је коришћења релације mod . Часовник користи аритметику mod 12. На пример, 3 сата после 11 је 2 сата, јер је $11 + 3 = 14 \equiv 2 \pmod{12}$. Парни бројеви су сви $\equiv 0 \pmod 2$, а непарни су $\equiv 1 \pmod 2$.

Конгруентност по модулу m је **релација еквиваленције**:

- $a \equiv a \pmod m$
- ако је $a \equiv b \pmod m$, онда је $b \equiv a \pmod m$
- ако је $a \equiv b \pmod m$ и $b \equiv c \pmod m$, онда је $a \equiv c \pmod m$.

Због тога релација mod m разбија све целе бројеве на m дисјунктних подскупова (класа еквиваленције за ову релацију). Сваки подскуп садржи тачно једног представника у интервалу $[0, m-1]$. Скуп ових подскупова означава се са $\mathbf{Z}/m\mathbf{Z}$ или $\mathbf{Z}_m$. Видимо да $\mathbf{Z}_m$ има m елемената; бројеви $0, 1, \ldots m-1$ су представници m елемената скупа $\mathbf{Z}_m$.

Особине конгруенције:

1. [сабирање и множење у $\mathbf{Z}_m$] ако је $a \equiv b \pmod m$ и $c \equiv d \pmod m$, онда је $a \pm c \equiv b \pm d \pmod m$ и $a \cdot c \equiv b \cdot d \pmod m$; на пример,

$$\begin{array}{ccc} 12, 14 & \overset{\text{mod } 5}{\longrightarrow} & 2, 4 \\ +\downarrow & & \downarrow + \\ 26 & \overset{\text{mod } 5}{\longrightarrow} & 1 \end{array}$$

   Другим речима, релација mod је **сагласна** са операцијама $+$, $-$ и $\cdot$. Ова чињеница омогућује извршавање ових операција у $\mathbf{Z}_m$. Нека је

$m = 5$. Тада је $\mathbf{Z}/5Z = \{\overline{0}, \overline{1}, \overline{2}, \overline{3}, \overline{4}\} = \{0, 1, 2, 3, 4\}$ (због једноставности поистовећујемо елементе $\mathbf{Z}/5Z$ са њиховим представницима — бројевима). У $\mathbf{Z}/5Z$ је $2 \cdot 3 = 1$, јер је $2 \cdot 3 = 6 \equiv 1 \pmod 5$; $3 + 4 = 2$, јер је $3 + 4 = 7 \equiv 2 \pmod 5$; $0 - 1 = 4$, јер је $0 - 1 \equiv 4 \pmod 5$. Таблица сабирања у $\mathbf{Z}_5$:

	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

2. [Промена модула конгруенције] Ако је $a \equiv b \pmod m$ и $d|m$, онда $a \equiv b \pmod d$. Тако из $12 \equiv 2 \pmod{10}$ следи $12 \equiv 2 \pmod 5$.

3. [Инверз] Неки елемент $x \in \mathbf{Z}_m$ има *мултипликативни инверз* $(1/x) = x^{-1}$ у $\mathbf{Z}_m$ ако је $\mathrm{nzd}(x, m) = 1$: пошто се $\mathrm{nzd}(x, m) = 1$ изрази у облику целобројне линеарне комбинације $1 = ax + bm$, види се да је $ax \equiv 1 \pmod m$. С друге стране, ако је $\mathrm{nzd}(x, m) = d > 1$, онда x нема инверз по модулу m: производ xa за произвољни цели број a дељив је са d па не може бити једнак 1. Скуп елемената $\mathbf{Z}_m$ који имају инверзе (односно скуп бројева узајамно простих са m) означава се са $\mathbf{Z}_m^*$. На пример $1/2 = 2^{-1} \equiv 3 \pmod 5$, јер је $2 \cdot 3 \equiv 1 \pmod 5$.

   У скупу $\mathbf{Z}_9 = \{0, 1, \ldots, 8\}$ могу да се користе операције $+$, $-$, $\cdot$. У скупу $\mathbf{Z}_9^* = \{1, 2, 4, 5, 7, 8\}$ могу да се користе операције $\cdot$ и $/$.

   Да бисмо пронашли инверз 7 по модулу 9, односно елеменат $7^{-1} \in \mathbf{Z}_9^*$, најпре примењујемо Еуклидов алгоритам за одређивање $\mathrm{nzd}(9, 7)$ (иако знамо да је тај НЗД једнак 1):

$$\begin{aligned} 9 &= 1 \cdot 7 + 2 \\ 7 &= 3 \cdot 2 + 1 \\ 2 &= 2 \cdot 1 + 0 \end{aligned}$$

   Затим се инверз одређује прелазећи овај низ једнакости уназад:

$$\begin{aligned} 1 &= 7 - 3 \cdot 2 \\ 1 &= 7 - 3(9 - 7) \\ 1 &= 4 \cdot 7 - 3 \cdot 9 \end{aligned}$$

   Посматрајмо ову једнакост по модулу 9 (то можемо јер је $a \equiv a \pmod m$). Добијамо $1 = 4 \cdot 7 - 3 \cdot 9 \equiv 4 \cdot 7 - 3 \cdot 0 \equiv 4 \cdot 7 \pmod 9$. Према томе, $1 \equiv 4 \cdot 7 \pmod 9$ и $7^{-1} = 1/7 = 4$ у $\mathbf{Z}_9$.

   Чему је једнако $2/7$ у $\mathbf{Z}_9$? $2/7 = 2 \cdot 1/7 = 2 \cdot 4 = 8 \in \mathbf{Z}_9$. Према томе, $2/7 \equiv 8 \pmod 9$. Приметимо да је $2 \equiv 8 \cdot 7 \pmod 9$, јер $9|2 - 56 = -54$.

Број 6 нема инверз по модулу 9, јер из $6x \equiv 1 \pmod 9$, следи $9|6x-1$, односно $3|6x-1$, односно $3|1$ (због $3|6x$), што није тачно. Дакле, 6 нема инверз по модулу 9.

4. [Дељење у $\mathbf{Z}_m^*$] Ако је $a \equiv b \pmod m$, $c \equiv d \pmod m$ и $\text{nzd}(c, m) = 1$ (из чега следи и $\text{nzd}(d, m) = 1$), онда је $ac^{-1} \equiv bd^{-1} \pmod m$, односно $a/c \equiv b/d \pmod m$. Према томе, дељење је дозвољено уколико је дељеник узајамно прост са модулом m, односно инвертибилан је по модулу m.

5. [Решавање конгруенције] Потребно је за дате a, b, m по x решити конгруенцију $ax \equiv b \pmod m$.

   - Ако је $\text{nzd}(a, m) = 1$, онда су решења сви бројеви $x \equiv a^{-1}b \pmod m$.
   - Ако је $\text{nzd}(a, m) = g > 1$, онда конгруенција има решења ако $g|b$. Тада је конгруенција еквивалентна са $ax/g \equiv b/g \pmod{m/g}$. Пошто је сада $\text{nzd}(a/g, m/g) = 1$, решења су $x \equiv (a/g)^{-1}(b/g) \pmod{m/g}$.
   - Ако је $\text{nzd}(a, m) = g > 1$ и $g \not| b$, онда конгруенција нема решења.

Следећи примери илуструју ова три случаја:

- Решити конгруенцију $7x \equiv 3 \pmod{10}$. Пошто је $\text{nzd}(7, 10) = 1$, решење је $\equiv 7^{-1} \cdot 3 \pmod{10}$. Одређивање $7^{-1} \pmod{10}$: $10 = 7+3, 7 = 2 \cdot 3+1$, па је $1 = 7 - 2(10 - 7) = 3 \cdot 7 - 2 \cdot 7$. Према томе, $1 \equiv 3 \cdot 7 \pmod{10}$ и $1/7 \equiv 3 \equiv 7^{-1} \pmod{10}$. Дакле, $x \equiv 3 \cdot 3 \equiv 9 \pmod{10}$. Решење је скуп позитивних бројева са цифром јединица 9 и скуп негативних бројева са цифром јединица 1.

- Решити конгруенцију $6x \equiv 8 \pmod{10}$. Пошто је $\text{nzd}(6, 10) = 2$, и $2|8$, решења постоје. Конгруенција је еквивалентна са $3x \equiv 4 \pmod 5$, па је $x \equiv 4 \cdot 3^{-1} \pmod 5$. Пошто је $3^{-1} \equiv 2 \pmod 5$, добија се $x \equiv 4 \cdot 2 \equiv 3 \pmod 5$. Другим речиме, $x = 3 + 5n$, где је $n \in \mathbf{Z}$, односно $x \equiv 3 \pmod{10}$ или $x \equiv 8 \pmod{10}$ (конгруенција има два решења по модулу 10).

- Решити $6x \equiv 7 \pmod{10}$. Ова конгруенција нема решења, јер је $\text{nzd}(6, 10) = 2$ и $2 \not| 7$.

Наравно, инверзија се не мора вршити помоћу Еуклидовог алгоритма. Нека је потребно одредити инверзе свих елемената $\mathbf{Z}_{17}^*$. Цели бројеви који су конгруентни са 1 по модулу 17 су облика $17n + 1$. Можемо да раставимо на чиниоце неколико таквих бројева. Првих неколико природних бројева облика $17n+1$ су 18, 35, 52. Пошто је $18 = 2 \cdot 9$, биће $2 \cdot 9 \equiv 1 \pmod{17}$ и $2^{-1} \equiv 9 \pmod{17}$, $9^{-1} \equiv 2 \pmod{17}$. Слично, из $18 = 3 \cdot 6$, следи да су бројеви 3 и 6 су узајамно инверзни по модулу 17. Због једнакости $35 = 5 \cdot 7$ су бројеви 5 и 7 међусобно инверзни. Слично, $52 = 4 \cdot 13$. Даље, $18 = 2 \cdot 9 \equiv (-2)(-9) \equiv 15 \cdot 8$

и $18 = 3 \cdot 6 = (-3)(-6) \equiv 14 \cdot 11$. Слично, $35 = 5 \cdot 7 = (-5)(-7) \equiv 12 \cdot 10$. Приметимо да је $16 \equiv -1$ и $1 = (-1)(-1) \equiv 16 \cdot 16$. Дакле, одредили смо инверзе свих елемената $\mathbf{Z}_{17}^*$.

Још једно вежбање: треба доказати да $x^3 - x - 1$, $x \in \mathbf{Z}$, никад није једнако квадрату неког целог броја. Заиста, квадрати целих бројева су $\equiv 0^2, 1^2, 2^2 \pmod 3 \equiv 0, 1, 1 \pmod 3$. С друге стране, конгруенција $x^3 - x - 1 \equiv 2 \pmod 3$ важи за свако x: $0^3 - 0 - 1 \equiv 2$, $1^3 - 1 - 1 \equiv 2$, $2^3 - 2 - 1 \equiv 2$.

Кинеска теорема о остацима. Нека су m_1, m_2,..., m_r, у паровима узајамно прости природни бројеви. Систем конгруенција $x \equiv a_1 \pmod{m_1}$, $x \equiv a_2 \pmod{m_2}$,..., $x \equiv a_r \pmod{m_r}$, има јединствено решење по модулу $m = m_1 m_2 \dots m_r$.

Пример: ако је $x \equiv 1 \pmod 7$ и $x \equiv 2 \pmod 4$, онда је $x \equiv 22 \pmod{28}$.

Један занимљив пример. Последње три цифре квадрата броја 6378000 су 000. Уствари, то је тачно за било који цели број коме су три последње цифре 000. Слично тврђење тачно је и за бројеве којима су три последње цифре 625; квадрат тих бројева завршава се исто са 625. Да ли постоји још неки троцифрени број са истом особином? Потребно је да решимо конгруенцију $x^2 \equiv x \pmod{1000}$. Уместо ње можемо да решимо $x^2 \equiv x \pmod 8$ и $x^2 \equiv x \pmod{125}$. Непосредно се проверава да су решења ових конгруенција $x \equiv 0, 1 \pmod 8$ $x \equiv 0, 1 \pmod{125}$. Тако добијамо четири решења:

- $x \equiv 0 \pmod 8$ и $x \equiv 0 \pmod{125}$ дају $x \equiv 0 \pmod{1000}$;
- $x \equiv 1 \pmod 8$ и $x \equiv 1 \pmod{125}$ дају $x \equiv 1 \pmod{1000}$;
- $x \equiv 1 \pmod 8$ и $x \equiv 0 \pmod{125}$ дају
- $x \equiv 625 \pmod{1000}$; $x \equiv 0 \pmod 8$ и $x \equiv 1 \pmod{125}$ дају $x \equiv 376 \pmod{1000}$. Заиста, $376^2 = 141376$.

Сада ћемо видети алгоритам за одређивање броја x. Нека је $m = m_1 m_2 \dots m_r$, и нека је $b_i \equiv (m/m_i)^{-1} \pmod{m_i}$, $i = 1, 2, \dots, r$ (ови инверзи постоје јер је $\mathrm{nzd}(m/m_i, m_i) = 1$).

- Потребан нам је израз који је конгруентан са a_1 по модулу m_1, и конгруентан са нула по осталим модулима m_i, $i \neq 1$. Ту особину има израз $a_1(m/m_1)b_1$,
- Слично, потребан нам је израз који је конгруентан са $a_2 \pmod{m_2}$, односно конгруентан са нула по осталим модулима m_i. Ту особину има израз $a_2(m/m_2)b_2$.
- ...
- На крају, потребан нам је израз који је конгруентан са $a_r \pmod{m_r}$, односно конгруентан са нула по осталим модулима m_i. Ту особину има израз $a_r(m/m_r)b_r$.

Дакле, тражени остатак x је

$$x = a_1(m/m_1)b_1 + a_2(m/m_2)b_2 + \ldots + a_r(m/m_r)b_r \pmod{m}.$$

Пример. Решити $x \equiv 2 \pmod 3$, $x \equiv 3 \pmod 5$, $x \equiv 9 \pmod{11}$. Пошто се израчунају бројеви

$$b_1 = (5 \cdot 11)^{-1} \pmod 3 = 1^{-1} \pmod 3 = 1,$$

$$b_2 = (3 \cdot 11)^{-1} \pmod 5 = 3^{-1} \pmod 5 = 2,$$

$$b_3 = (3 \cdot 5)^{-1} \pmod{11} = 4^{-1} \pmod{11} = 3,$$

добија се да је

$$x = 2(5 \cdot 11)1 + 3(3 \cdot 11)2 + 9(3 \cdot 5)3 = 713 \equiv 53 \pmod{165}.$$

Ојлерова функција φ. Нека $n \in \mathbf{Z}_+$. Нека је $\mathbf{Z}_n^* = \{a \mid 1 \leq a \leq n, \mathrm{nzd}(a,n) = 1\}$. Овај скуп је **група** за множење (скуп је затворен за множење; множење је асоцијативно, 1 је неутрални елемент, и сваки елемент има инверз). На пример, $\mathbf{Z}_{12}^* = \{1,5,7,11\}$. Нека је $\varphi(n) = |\mathbf{Z}_n^*|$. На пример, $\varphi(12) = 4$, $\varphi(5) = 4$ и $\varphi(6) = 2$. Ако је p је прост број, онда је $\varphi(p) = p - 1$. Чему је једнако $\varphi(5^3)$? Скуп $\mathbf{Z}_{125}^*$ добија се од $\mathbf{Z}_{125}$ избацивањем умножака 5. Умножака 5 има $125/5$, па је $\varphi(125) = 125 - 25$. Ако је $r \geq 1$, а p је прост број, онда је $\varphi(p^r) = p^r(1 - \frac{1}{p}) = p^{r-1}(p-1)$; специјално, $\varphi(p) = p - 1$.

За остале природне бројеве се вредност Ојлерове функције може израчунати коришћењем чињенице да ако је $\mathrm{nzd}(m,n) = 1$, онда $\varphi(mn) = \varphi(m)\varphi(n)$. Заиста, за произвољни цели број z важи да је $\mathrm{nzd}(z,mn) = 1$ ако и само ако је $\mathrm{nzd}(z,m) = 1$ и $\mathrm{nzd}(z,n) = 1$. На основу кинеске теореме о остацима сваком пару x, y остатака по модулима m, n, узајамно простих са тим модулима, једнозначно одговара остатак z по модулу mn, такав да је $z \equiv x \pmod m$ и $z \equiv y \pmod n$. Одатле следи да је $\mathrm{nzd}(z,mn) = 1$ тачно ако и само ако је истовремено тачно $\mathrm{nzd}(x,m) = 1$ и $\mathrm{nzd}(y,n) = 1$). Дакле, да би се израчунало $\varphi(n)$, потребно је n раставити на просте чиниоце. На пример, $\varphi(720) = \varphi(2^4)\varphi(3^2)\varphi(5) = 2^3(2-1)3^1(3-1)(5-1) = 192$. Уопште, ако је $n = \prod p_i^{\alpha_i}$, онда је

$$\varphi(n) = p_1^{\alpha_1 - 1}(p_1 - 1) \ldots p_r^{\alpha_r - 1}(p_r - 1).$$

Мала Фермаова теорема. Ако је p прост и $a \in \mathbf{Z}$, онда је $a^p \equiv a \pmod p$. Ако p не дели a, онда је $a^{p-1} \equiv 1 \pmod p$. Општије, ако је $\mathrm{nzd}(a,m) = 1$, онда је $a^{\varphi(m)} \equiv 1 \pmod m$.

Заиста, скуп остатака $\{ax \pmod m \mid x \in \mathbf{Z}_m^*\}$ једнак је скупу $\mathbf{Z}_m^*$ елементи првог скупа су испремештани елементи другог скупа: (ако x пролази све остатке из $\mathbf{Z}_m^*$ онда и $ax \pmod m$ пролази све остатке из $\mathbf{Z}_m^*$). Због тога је

$$\prod_{x \in \mathbf{Z}_m^*} x = \prod_{x \in \mathbf{Z}_m^*} ax = a^{\varphi(m)} \prod_{x \in \mathbf{Z}_m^*} x.$$

Скраћивањем са бројем $\prod_{x \in \mathbf{Z}_m^*} x$ узајамно простим са m добија се $a^{\varphi}(m) \equiv 1 \pmod{m}$.

Због ове теореме смо сигурни да је, пошто је 5 прост број, $2^5 \equiv 2 \pmod 5$, $4^5 \equiv 4 \pmod 5$, $2^4 \equiv 1 \pmod 5$ (ове три конгруенције могу се и непосредно проверити).

Други пример: пошто је $\varphi(10) = \varphi(5)\varphi(2) = 4 \cdot 1 = 4$, и $\mathbf{Z}_{10}^* = \{1, 3, 7, 9\}$, биће $1^4 \equiv 1 \pmod{10}$, $3^4 \equiv 1 \pmod{10}$, $7^4 \equiv 1 \pmod{10}$ и $9^4 \equiv 1 \pmod{10}$.

Ако је $\mathrm{nzd}(c, m) = 1$ и $a \equiv b \pmod{\varphi(m)}$, при чему је $a, b \in \mathbf{Z}_{\geq 0}$, онда је $c^a \equiv c^b \pmod m$.

На пример, ако треба израчунти остатак $2^{1004} \mod 15$, приметимо да је $\varphi(15) = \varphi(5)\varphi(3) = 4 \cdot 2 = 8$ и $1004 \equiv 4 \pmod 8$. Према томе, $2^{1004} \equiv 2^4 \equiv 16 \equiv 1 \pmod{15}$. Другим речима, експонент се може заменити својим остатком по модулу $\varphi(m)$ уколико је основа узајамно проста са модулом.

Одређивање остатка $a \mod m$ помоћу калкулатора. Да бисмо одредили $1000 \mod 23$ уносимо у калкулатор $1000/23 =$ (видимо 43.478...) $-43 =$ (видимо 0.478...) $\times 23 =$ (видимо 11). Према томе, $1000 \equiv 11 \pmod{23}$. Зашто је овај поступак коректан? Како је $1000 = 43 \cdot 23 + 11$, дељењем десне стране са 23, добијамо $43 + \frac{11}{23}$; одузимањем 43 добија се $\frac{11}{23}$; множењем са 23, добија се 11.

5 Једноставни шифарски системи

Нека је $\mathcal{P}$ скуп могућих отворених текстова. То може да буде, на пример, скуп {A,B,...,Z} величине 26, или скуп {AA,AB,...,ZZ} величине 26^2. Нека је $\mathcal{C}$ скуп могућих шифрата.

Шифарска трансформација f је једнозначна функција која пресликава $\mathcal{P}$ у $\mathcal{C}$ (f не сме да пресликава два различита отворена текста у исти шифрат). Пар трансформација $f : \mathcal{P} \to \mathcal{C}$ и $f^{-1} : \mathcal{C} \to \mathcal{P}$ је шифарски систем. Размотрићемо сада неколико једноставних шифарских система.

Размотрићемо најпренеколико једноставних шифарским системом који трансформишу појединачна слова. Слова се могу замењивати другим словима, при чему се може користити произвољна пермутација, на пример A $\to$ F, B $\to$ Q, C $\to$ N, ... Уместо да се пермутација задаје таблицом, може се задати изразом који описује процес шифровања, односно дешифровања.

Најједноставнија трансформација је *транслација*. Нека P означава слово отвореног текста (односно одговарајући број) A = 0, B = 1, ..., Z = 25. Шифровање се описује једнакошћу $C \equiv P + 3 \pmod{26}$; због тога дешифровање описује једнакост $P \equiv C - 3 \pmod{26}$. Ово је Цезарова шифра. Ако је величина азбуке N, онда је шифарска операција транслације $C \equiv P + b \pmod N$, где је b кључ за шифровање, а $-b$ је кључ за дешифровање.

Да би могла да изврши криптоанализу, Цица која зна да се користи транслација, треба да одреди само кључ b. У општем случају мора се претпоставити да Цица зна тип примењеног шифарског система (овде — транслација).

Претпоставимо да нам је на располагању велика количина шифрата и да желимо да одредимо b, да бисмо могли да читамо наредне поруке. Један могући метод је испробати свих могућих $N = 26$ вредности за b. Очекујемо да ће само једна од њих давати смислене резултате дешифровања. Други начин заснива се коришћењу учестаности појављивања слова. Знамо да је E = 4 најчешће слово (у енглеском језику). Ако се испостави да је у прикупљеним шифратима најчешће слово J = 9, онда је највероватније $b = 5$.

Афина шифарска трансформација је облика $C \equiv aP + b \pmod N$, где је кључ пар (a, b). При томе је неопходно да буде $\text{nzd}(a, N) = 1$, јер би у противном различите отворене текстове трансформација пресликавала у исте шифрате. Заиста, ако је $d = \text{nzd}(a, N)$, $g = N/d$, онда за свако i, $0 \leq i < d$, важи

$$a(P + ig) + b \equiv aP + i(a/d) \cdot gd + b \equiv aP + i(a/d) \cdot N + b \equiv aP + b \pmod N.$$

Пример: $C \equiv 4P + 5 \pmod{26}$ (овде је $\text{nzd}(4, 26) = 2 \neq 1$). Приметимо да се и $B = 1$ и $O = 14$ пресликавају у исто слово $9 = J$.

Пример: шифровање $C \equiv 3P+4 \pmod{26}$ је регуларно, јер је $\text{nzd}(3, 26) = 1$. Алиса шаље поруку U Бобану; $U = 20$ пресликава се у $3 \cdot 20 + 4 = 64 \equiv 12 \pmod{26}$. Према томе, $U = 20 \to 12 = M$, тј. Алиса шаље Бобану слово M. Бобан може да дешифрује поруку решавајући једначину по P: $P \equiv 3^{-1}(C - 4) \pmod{26}$. Како је $3^{-1} \equiv 9 \pmod{26}$ (јер је $3 \cdot 9 = 27 \equiv 1 \pmod{26}$), биће $P \equiv 9(C-4) \equiv 9C-36 \equiv 9C+16 \pmod{26}$. Пошто је Бобан примио $M = 12$, он израчунава $9 \cdot 12 + 16 = 124 \equiv 20 \pmod{26}$.

Уопште, шифровању $C \equiv aP + b \pmod N$ одговара дешифровање $P \equiv a^{-1}(C - b) \pmod N$. Овде је $(a^{-1}, -a^{-1}b)$ кључ за дешифровање.

Како Цица може да изврши криптоанализу? У случају $N = 26$ може да испроба свих $\varphi(26) \cdot 26 = 312$ могућих кључева (a, b), или да искористи учестаности слова у шифрату. Пошто кључ има две компоненте, потребно је имати две једначине. Претпоставимо да Цица има на располагању велику количину шифрата. Тада она може да установи да је $Y = 24$ најчешће, а $H = 7$ следеће најчешће слово у шифрату. Претпоставимо да се дешифровање врши на основу $P \equiv a'C + b' \pmod{26}$, где је $a' = a^{-1}$ и $b' = -a^{-1}b$. Потребно је дешифровати поруку $VNLGDO$.

Најпре се одређује (a', b'). Претпоставимо да су слова Y, H шифрати редом слова E, T, односно $E = 4 \equiv a'24 + b' \pmod{26}$ и $T = 19 \equiv a'7 + b' \pmod{26}$. Одузимањем се добија

$$17a' \equiv 4 - 19 \equiv 4 + 7 \equiv 11 \pmod{26} \qquad (*)$$

Према томе, $a' \equiv 17^{-1}11 \pmod{26}$. Користећи Еуклидов алгоритам добија се $17^{-1} \equiv 23 \pmod{26}$, па је $a' \equiv 23 \cdot 11 \equiv 19 \pmod{26}$. Замењујући ово у претходну једнакост, добија се $19 \equiv 19 \cdot 7 + b' \pmod{26}$, па је $b' \equiv 16 \pmod{26}$. Дакле, $P \equiv 19C + 16 \pmod{26}$.

Сада може да се дешифрује $VNLGDO$, односно низ 21 13 11 6 3 14. Добија се $19 \cdot 21 + 16 \equiv 25 = Z$, $19 \cdot 13 + 16 \equiv 3 = D$, ..., па је резултат

дешифровања реч $ZDRAVO$. Погледајмо још једном једнакост (*); могуће је да се добије конгруенција као што је $2a' \equiv 8 \pmod{26}$. Њена решења су $a' \equiv 4 \pmod{13}$, па је $a' \equiv 4$ или $a' \equiv 17 \pmod{26}$. Тада је потребно проверити оба решења, односно установити које од њих даје смислени отворени текст.

Цица сада може да се лажно представи као Алиса, и да пошаље нпр. поруку $NEMOJ$, односно 13 4 12 14 9. Пошто се ради о шифровању, користи се израз $C \equiv aP+b \pmod{26}$. Полазећи од $P \equiv 19C+16 \pmod{26}$, добија се $C \equiv 19^{-1}(P-16) \equiv 11P+6 \pmod{26}$. Шифрат поруке $NEMOJ$ је $TYIEB$.

Исти тип система могао би да се користи за шифровање биграма (парова слова). Ако се користи алфабет A-Z, у коме су слова нумерисана бројевима $0-25$, онда се биграм xy кодира са $26x+y$. Добијени број је између 0 и $675 = 26^2 - 1$. На пример, TO постаје $26 \cdot 19 + 14 = 508$. Да би се ово декодирало, дели се 508 са 26 са остатком, $508 = 26 \cdot 19 + 14$. Ипак, шифровање применом једнакости $C \equiv aP+b \pmod{675}$ је лоше, јер ако се два биграма завршавају истим словом, онда ће се резултујући шифрати такође завршавати истим словом.

Криптоанализа је разбијање шифри или проучавање разбијања шифри. Шифарски системи могу се поделити у три категорије:

1. Они који су разбијени (већина).
2. Они који нису до сада били анализирани (зато што су нови и нису широко коришћени).
3. Они који су анализирани, али нису разбијени. (RSA , системи који користе дискретни логаритам, троструки DES , AES).

Најчешћа три начина да Цица добије отворени текст који одговара неком шифрату су

1. Крађом, куповином кључа, односно подмићивањем.
2. Коришћењем слабости у реализацији, односно проблема са протоколом. На пример, неко користи име супружника као кључ, или неко пошаље кључ заједно са поруком.
3. Криптоанализом.

6 Савремене проточне шифре

Савремене проточне шифре су симетрични шифарски системи. Алиса и Бобан морају унапред да се договоре који ће кључ користити. Отворени текст се најпре трансформише применом ASCII кода у низ нула и јединица. На пример, отворени текст Go кодира се са 0100011101101111. Алиса и Бобан одабирају неки *генератор псеудослучајних бројева* и договарају се које ће почетно стање (seed), односно кључ користити. Изабраним генератором обоје изгенеришу исти псеудослучајни низ бита (*низ кључа*), на

пример 0111110110001101. Алиса израчунава шифрат сабирањем по модулу два, бит по бит ($0 \oplus 0 = 0$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$, $1 \oplus 1 = 0$).

Пример:

		шифровање			дешифровање
	ОТ	0100011101101111		СТ	0011101011100010
низ кључа	$\oplus$	0111110110001101	низ кључа	$\oplus$	0111110110001101
	СТ	0011101011100010			0100011101101111
				ОТ	Go

Нека је p_i i-ти бит отвореног текста, k_i i-ти бит низа кључа, а c_i i-ти бит шифрата. Тада се шифровање врши на основу израза $c_i = p_i \oplus k_i$, а дешифорвање на основу израза $p_i = c_i \oplus k_i$.

Наводимо пример несигурне проточне шифре, која се користи на персоналним рачунарима за шифровање фајлова. Одабере се нека реч, нпр. Sue 01010011 01110101 01100101 (кључ). Низ кључа се добија периодичним понављањем кључа. Добра страна овог система је променљива дужина кључа. За одређивање дужине кључа и самог кључа може се користити чињеница да се обично само око 1/4 могућих ASCII знакова ефективно користи у текстовима. Због тога, ако се покуша са дешифровањем погрешним кључем, велика је вероватноћа да ће се у резултату дешифровања појавити неки непринтабилни ASCII знак.

7 Коначна поља

Ако је p прост број, означимо са $\mathbf{F}_p = \mathbf{Z}_p$ поље са p елемената $\{0, 1, \ldots, p-1\}$ са операцијама $+$, $-$, $\times$. Приметимо да за елементе $\alpha \neq 0$ важи $\mathrm{nzd}(\alpha, p) = 1$, па се може одредити α^{-1}. Због тога се може делити било којим елементом различитим од нуле. Ово поље слично је пољима рационалних, реалних или комплексних бројева.

У скупу $\mathbf{F}_p^* = \{1, 2, \ldots, p-1\}$ се могу користити операције $\times$, $/$. Група $\mathbf{F}_p^*$ је цикличка, тј. садржи бар један елеменат g такав да је $\{1, g, g^2, g^3, \ldots\} = \mathbf{F}_p^*$ (овде то нећемо доказивати). За елеменат g се такође каже да је *примитиван корен* по модулу p. Скупови $\{g, g^2, g^3, \ldots, g^{p-1}\}$ и $\{1, 2, \ldots, p-1\}$ су једнаки (иако су њихови елементи можда написани различитим редоследом).

На пример,

- у групи $\mathbf{F}_5^*$, $g = 2$: $2^1 = 2$, $2^2 = 4$, $2^3 = 3$, $2^4 = 1$.
- У овој групи и 3 је генератор: $3^1 = 3$, $3^2 = 4$, $3^3 = 2$, $3^4 = 1$.
- У групи $\mathbf{F}_7^*$ је $2^1 = 2$, $2^2 = 4$, $2^3 = 1$, $2^4 = 2$, $2^5 = 4$, $2^6 = 1$, па 2 није генератор.
- За $g = 3$ добија се $3^1 = 3$, $3^2 = 2$, $3^3 = 6$, $3^4 = 4$, $3^5 = 5$, $3^6 = 1$, тј. 3 јесте генератор.

Неки елеменат $x \in \mathbf{F}_p^*$ јесте генератор ако и само ако је његов *ред* (број различитих елемената у скупу $\{x, x^2, x^3, \ldots\}$) једнак $p-1$.

Теорема: Нека је g генератор групе $\mathbf{F}_p^*$. Тада је g^k генератор ове групе ако и само ако је $\mathrm{nzd}(k, p-1) = 1$.

Последица теореме је да група $\mathbf{F}_p^*$ има $\varphi(p-1)$ генератора.

Доказ: Претпоставимо да је $\mathrm{nzd}(k, p-1) = 1$. Нека је n ред елемента g^k, $1 \leq n \leq p-1$. Тада је $1 = (g^k)^n = g^{kn}$. Према томе, $p-1 | kn$. Пошто је $\mathrm{nzd}(k, p-1) = 1$, биће $p-1 | n$, па дакле $p-1 = n$. Према томе, g^k је генератор.

Доказ у супротном смеру: ако је $\mathrm{nzd}(k, p-1) = d > 1$, онда g^k није генератор, јер је $(g^k)^{(p-1)/d} = 1$.

Претпоставимо да је p велики прост број. Нека је изабран неки генератор g групе $\mathbf{F}_p^*$ и неки елеменат $h \in \mathbf{F}_p^*$. Тада је јако тешко одредити x такво да је $g^x = h$, иако знамо да оно постоји. На пример, у $\mathbf{F}_7$, уз $g = 5$, решавање $5^x = 2$ еквивалентно је са конгруенцијом $5^x \equiv 2 \pmod 7$. Ово је *проблем дискретног логаритма*.

Наводимо пример генератора псеудослучајног низа бита, који је спор, па се практично не користи. Нека је p велики прост број за који је 2 генератор $\mathbf{F}_p^*$; претпоставимо да је поред тога и $q = 2p+1$ такође прост број. Нека је g генератор $\mathbf{F}_q^*$. Нека је број k кључ, при чему је $\mathrm{nzd}(k, 2p) = 1$. Нека је $s_1 = g^k \in \mathbf{F}_q$ (па је $1 \leq s_1 < q$) и $k_1 \equiv s_1 \pmod 2$, $k_1 \in \{0, 1\}$. За $i \geq 1$ нека је $s_{i+1} = s_i^2 \in \mathbf{F}_q$, $1 \leq s_i < q$ и $k_i \equiv s_i \pmod 2$, $k_i \in \{0, 1\}$. Због тога што је 2 генератор у $\mathbf{F}_p^*$, сви остаци $2^i \mod p$, $0 \leq i \leq p-1$ су различити. Како је $s_i = g^{k 2^{i-1}}$ и $p = (q-1)/2 \mid \phi(q)$, види се да су за $i = 1, \ldots, p$ сви бројеви s_i различити.

Пример. Број 2 је генератор $\mathbf{F}_{29}^*$ (ово следи из чињенице да је $2^{28/2} \neq 1$ и $2^{28/7} \neq 1$; зашто?). Број 2 је такође и генератор $\mathbf{F}_{59}^*$. Нека је $k = 11$. Тада је $s_1 = 2^{11} = 42$, $s_2 = 42^2 = 53$, $s_3 = 53^2 = 36$, $s_4 = 36^2 = 57$, ... па је $k_1 = 0$, $k_2 = 1$, $k_3 = 0$, $k_4 = 1$, ...

8 RC4

RC4 је једна од најпопуларнијих проточних шифара. Њен аутор је Ronald Rivest (слово R у RSA, 1987). Структура овог псеудослучајног генератора била је држана у тајности, све док његов изворни код није (проваљен и) објављен 1994. (Cypherpunks mailing list).

Најпре се бира природни број n; обично се обично користи $n = 8$. Основна компонента генератора је променљива табела S дужине 2^n, чији је садржај у сваком тренутку нека пермутација бројева $i = 0, 1, \ldots, 2^n - 1$.

У фази припреме генератора се на основу кључа израчунава почетни садржај низа S — пермутација скупа $\{0, 1, \ldots, 2^n - 1\}$. Израчунавање почиње тако што се стави $S_i = i$ за $i = 0, 1, \ldots, 2^n - 1$. Затим се од кључа (поновљањем довољно пута) формира други низ K_0, K_1, ..., K_{2^n-1} од 2^n n-торки бита (које се такође сматрају бројевима из опсега од 0 до $2^n - 1$). Рад генератора

при генерисању l n-битних блокова низа кључа KS_r, $r = 0, 1, \ldots, l-1$, описује се следећим кодом.

$j \leftarrow 0$ {иницијализација бројача}
for $i \leftarrow 0$ **to** $2^n - 1$ **do**
  $j \leftarrow j + S_i + K_i \pmod{2^n}$
  заменити S_i и S_j
$i \leftarrow 0; j \leftarrow 0$ {иницијализација бројача}
{ почетак генерисања l случајних n-торки бита:}
for $r \leftarrow 0$ **to** $l - 1$ **do**
  $i \leftarrow i + 1 \pmod{2^n}$
  $j \leftarrow j + S_i \pmod{2^n}$
  заменити S_i и S_j
  $t \leftarrow S_i + S_j \pmod{2^n}$
  $KS_r \leftarrow S_t$ {наредна n-торка бита излазног низа кључа}

Индекс i обезбеђује да се сваки елеменат низа (табеле) S промени бар једном, а индекс j обезбеђује да се елементи мењају на компликовани, привидно случајан начин.

Демонстрираћемо рад овог алгоритма за $n = 3$.

Нека је кључ 011001100001101, односно 011 001 100 001 101, односно $[3, 1, 4, 1, 5]$. Периодичним проширивањем добијамо низ дужине $2^n =$

$$[3, 1, 4, 1, 5, 3, 1, 4] = [K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7].$$

i	j	t	KS_t	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
	0			0	1	2	3	4	5	6	7
0	3			3	1	2	0	4	5	6	7
1	5			3	5	2	0	4	1	6	7
2	3			3	5	0	2	4	1	6	7
3	6			3	5	0	6	4	1	2	7
4	7			3	5	0	6	7	1	2	4
5	3			3	5	0	1	7	6	2	4
6	6			3	5	0	1	7	6	2	4
7	6			3	5	0	1	7	6	4	2
0	0										
1	5	3	1	3	6	0	1	7	5	4	2
2	5	5	0	3	6	5	1	7	0	4	2
3	6	5	0	3	6	5	4	7	0	1	2
4	5	7	2	3	6	5	4	0	7	1	2
5	4	7	2	3	6	5	4	7	0	1	2
6	5	1	6	3	6	5	4	7	1	0	2
7	7	4	7	3	6	5	4	7	1	0	2
0	2	0	5	5	6	3	4	7	1	0	2
1	0	3	4	6	5	3	4	7	1	0	2
2	3	7	2	6	5	4	3	7	1	0	2
3	6	3	0	6	5	4	0	7	1	3	2
4	5	0	6	6	5	4	0	1	7	3	2

Низ кључа добија се од тробитних репрезентација бројева $1, 0, 0, 2, 2, 6, 7, 5, 4, 2, 0, 6$, односно 001 000 000 010 010 110 111 101 100 010 000 110 (без размака).

9 Самосинхронишућа проточна шифра

Када се отворени текст бит по бит сабира по модулу два са низом кључа да би се добио шифрат, онда је то *синхрона проточна шифра*. Ако се приликом преноса (случајно или намерно) промени један бит, биће покварен одговарајући бит у дешифрованој поруци. Ако се у преносу уметне или избрише један бит, губи се синхронизам и остатак дешифроване поруке постаје нечитљив. Цица може да дође у посед пара одговарајућих OT/CT, па да на основу тога одреди део низа кључа, и да на крају некако реконструише кључ. У процес шифровања могу се укључити претходни бити отвореног текста:

$$c_i = p_i + k_i + f(p_{i-1}, p_{i-2}, \ldots, p_{i-k+1})$$

таква шифра зове се *самосинхронишућа*. Ни ова шифра није отпорна на брисање или уметање бита у шифрат, али ако дође до грешке у преносу, онда то изазива највише k узастопних грешака у дешифрованој поруци (после грешке у преносу долази до самосинхронизације). За исправно дешифровање неопходно је да пошиљалац и прималац користе исти почетни

део низа p_{-k+1}, p_{-k+2}, ..., p_0. Дешифровање се врши на основу израза

$$p_i = c_i + k_i + f(p_{i-1}, p_{i-2}, \ldots, p_{i-k+1}).$$

Наводимо пример овакве шифре. Нека је

$$c_i = p_i + k_i + \begin{cases} p_{i-2} & ako\ je\ p_{i-1} = 0 \\ p_{i-3} & ako\ je\ p_{i-1} = 1 \end{cases}$$

Отворени текст се мозе додефинисати тако што се стави $p_{-1} = p_0 = 0$. Прималац поруке користи за дешифровање израз

$$p_i = c_i + k_i + \begin{cases} p_{i-2} & ako\ je\ p_{i-1} = 0 \\ p_{i-3} & ako\ je\ p_{i-1} = 1 \end{cases}$$

На пример, за отворени текст (Go) и низ кључа из претходног примера, добија се:

	шифровање		дешифровање	
ОТ	000100011101101111	СТ	0010101000001111	
низ кључа	0111110110001101	низ кључа	0111110110001101	
СТ	0010101000001111	ОТ	000100011101101111	Go

10 Случајна шифра

Ако је кључ (не низ кључа) за проточну шифру случајан и дужине веће или једнаке од дужине отвореног текста, онда се систем зове *случајна шифра* (енглески one-time-pad). Кључ се никада не сме употребити за шифровање неке друге поруке (у противном је понекад могуће на основу два шифрата одредити низ кључа, тзв. напад Казиског). Криптоанализа овакве шифре је немогућа, јер су сва могућа дешифровања (која одговарају свим потенцијалним кључевима) једнако вероватна, односно из шифрата се не добија никаква информација о отвореном тексту. Овај систем коришћен је нпр. за време хладног рата за везу црвеним телефоном између Москве и Вашингтона. Ипак, због своје непрактичности се ова шифра не користи универзално.

11 Коначна поља II

Размотрићемо сада другачију врсту коначних поља. Нека је $\mathbf{F}_2[x]$ скуп полинома са коефицијентима из $\mathbf{F}_2 = \mathbf{Z}_2 = \{0, 1\}$. Приметимо да је $-1 = 1$, па је одузимање исто што и сабирање. Полиноми из овог скупа су

$$0, 1, x, x+1, x^2, x^2+1, x^2+x, x^2+x+1, \ldots$$

Постоје два полинома степена 0 (0, 1), четири полинома степена ≤ 1, осам полинома степена ≤ 2; уопште, број полинома степена $\leq n$ је 2^{n+1}. То су

полиноми $a_n x^n + \cdots + a_1 x + a_0$, $a_i \in \{0, 1\}$. Полиноми се множе на уобичајени начин, при чему се са коефицијентима рачуна у $\mathbf{F}_2$:

$$\begin{array}{cccccccc} & & (x^2 & + & x & + & 1)\times \\ & & & & (x^2 & + & x) = \\ \hline & & x^3 & + & x^2 & + & x \\ x^4 & + & x^3 & + & x^2 & & \\ \hline = x^4 & & & & & + & x \end{array}$$

Неки полином је *несводљив* над пољем ако се не може раставити у производ полинома (нижег степена) са коефицијентима из тог поља. Над пољем рационалних бројева полиноми $x^2 + 2$ и $x^2 - 2$ су несводљиви. Над пољем реалних бројева полином $x^2 + 2$ је несводљив, а полином $x^2 - 2 = (x - \sqrt{2})(x + \sqrt{2})$ није несводљив. Над пољем комплексних бројева ни полином $x^2 + 2 = (x + i\sqrt{2})(x - i\sqrt{2})$ није несводљив.

Над пољем $\mathbf{F}_2$ полином x^2+x+1 није дељив ни једним полиномом првог степена, па је несводљив (то је једини несводљиви квадратни полином); полином $x^2+1 = (x+1)^2$ није несводљив. Једини несводљиви кубни полиноми над $\mathbf{F}_2$ су $x^3 + x + 1$ и $x^3 + x^2 + 1$. Уопште, несводљиви полиноми добијају се полазећи од списка свих полинома прецртавањем умножака несводљивих полинома, слично као што се прости бројеви добијају применом Ератостеновог сита.

Када се елементи $\mathbf{Z}$ сведу по модулу простог броја p (дакле нерастављивог броја), добијају се остаци $0, 1, \ldots, p - 1$, тј. бројеви мањи од p. Тај скуп означавамо са $\mathbf{Z}/p\mathbf{Z}$ или $\mathbf{Z}/(p)$ (овде (p) означава скуп свих умножака полинома p).

Посматрајмо сада полиноме $\mathbf{F}_2[x]$ и њихове остатке по модулу несводљивог полинома x^3+x+1. Остаци су сви полиноми степена мањег од 3; при томе важи $x^3 + x + 1 \equiv 0 \pmod{x^3 + x + 1}$, тј. $x^3 \equiv x + 1 \pmod{x^3 + x + 1}$ (конгруенција по модулу полинома је природно уопштење конгруенције по модулу целог броја; ову конгруенцију по модулу x^3+x+1 писаћемо убудуће као једнакост). Према томе, на скупу $\mathbf{F}_2[x]/(x^3+x+1) = \{0, 1, x, x+1, x^2, x^2+1, x^2+x, x^2+x+1\}$ дефинисане су уобичајене операције $+$, $(-)$, $\times$, при чему је $x^3 = x + 1$.

Пример множења у $\mathbf{F}_2[x]/(x^3 + x + 1)$:

$$\begin{array}{cccccccc} & & (x^2 & + & x & + & 1)\times \\ & & & & (x & + & 1) = \\ \hline & & x^2 & + & x & + & 1 \\ x^3 & + & x^2 & + & x & & \\ \hline = x^3 & & & & & + & 1 \end{array}$$

Пошто је $x^3 = x+1$, биће $x^3+1 \equiv (x+1)+1 \pmod{x^3+x+1}$, односно $x^3+1 \equiv x \pmod{x^3 + x + 1}$. Према томе, $(x^2 + x + 1)(x + 1) = x$ у $\mathbf{F}_2[x]/(x^3 + x + 1)$. Ово поље означава се са $\mathbf{F}_8$ јер има 8 елемената. Приметимо да је у $\mathbf{F}_8$ $x^4 = x^3 \cdot x = (x + 1)x = x^2 + x$.

Уопште, ако је $p(x)$ несводљиви полином степена d, онда је скуп $\mathbf{F}_2[x]/(p(x))$ поље са 2^d елемената, које се означава са $\mathbf{F}_{2^d}$. Ово поље састоји се од свих полинома степена $\leq d-1$. $\mathbf{F}_{2^d}^*$ је скуп елемената поља различитих од нуле. Мултипликативна група тог поља је такође циклична (тј. у њој је бар један елеменат генератор — без доказа). Број елемената у тој групи је 2^d-1, па је број генератора једнак $\varphi(2^d-1)$. На пример, у пољу $\mathbf{F}_8$ је полином x генератор $\mathbf{F}_8^*$: $g=x, x^2, x^3=x+1, x^4=x^2+x, x^5=x^4\cdot x=x^3+x^2=x^2+x+1$, $x^6=x^3+x^2+x=x^2+1$, $x^7=x^3+x=1$. Сви генератори у овом пољу су g^a, где је $\mathrm{nzd}(a,2^3-1)=1$, односно $a=1,2,\ldots,6$.

Елементи поља лако се могу представити у рачунару. На пример, полином $1\cdot x^2+0\cdot x+1$ представља се тројком 101. Због тога се често користе шифарски системи засновани на проблему дискретног логаритма у пољима типа $\mathbf{F}_{2^d}$ уместо у пољима типа $\mathbf{F}_p$. Типичне вредности су $p\approx 2^d\approx 10^{300}$. Сматра се да је систем над $\mathbf{F}_p$ сигурнији, а да је на рачунару једноставнија реализација система над $\mathbf{F}_{2^d}$.

Инверзију елемента над пољем $\mathbf{F}_{2^d}$ илустроваћемо на примеру инвертовања x^4+x^3+1 у пољу $\mathbf{F}_2[x]/(x^6+x+1)$. Потребно је применити Еуклидов алгоритам на полиноме x^6+x+1 и x^4+x^3+1; полином x^6+x+1 је несводљив, па знамо да је резултат $\mathrm{nzd}(x^6+x+1,x^4+x^3+1)=1$; на основу тога треба НЗД изразити као линеарну комбинацију полинома x^6+x+1 и x^4+x^3+1, и тако добити инверз полинома x^6+x+1 по модулу x^4+x^3+1. Прво дељење даје $x^6+x+1=q(x^4+x^3+1)+r$, где је r полином степена мањег од 4 (степен дељеника).

										x^2	$+$	x	$+$	1	$=q$
x^4+x^3+1	$\mid$	x^6									$+$	x	$+$	1	
		x^6	$+$	x^5				$+$	x^2						
				x^5				$+$	x^2	$+$	x				
				x^5	$+$	x^4				$+$	x				
						x^4		$+$	x^2			$+$	1		
						x^4	$+$	x^3				$+$	1		
								x^3	$+$	x^2					$=r$

Према томе, $x^6+x+1=(x^2+x+1)(\underline{x^4+x^3+1})+(\underline{x^3+x^2})$. Наредним дељењем добија се $x^4+x^3+1=x(\underline{x^3+x^2})+\underline{1}$. Као и код Еуклидовог алгоритма са бројевима, ово омогућује да се $1=\mathrm{nzd}(x^6+x+1,x^4+x^3+1)$ изрази као линеарна комбинација ова два полинома:

$$\begin{aligned}
1 &= (\underline{x^4+x^3+1})+x(\underline{x^3+x^2}) \\
1 &= (\underline{x^4+x^3+1})+x(\underline{x^6+x+1}+(x^2+x+1)(\underline{x^4+x^3+1})) \\
1 &= 1(\underline{x^4+x^3+1})+x(\underline{x^6+x+1})+(x^3+x^2+x)(\underline{x^4+x^3+1}) \\
1 &\equiv (x^3+x^2+x+1)(x^4+x^3+1) \pmod{x^6+x+1}
\end{aligned}$$

Дакле, у пољу $\mathbf{F}_2[x]/(x^6+x+1)=\mathbf{F}_{64}$ је $(x^4+x^3+1)^{-1}=x^3+x^2+x+1$. Крај примера.

У описаном пољу $\mathbf{F}_8$ са полиномима из $\mathbf{Z}[x]$ ради се по два модула: коефицијенти се рачунају по модулу два, а полиноми по модулу $x^3 + x + 1$. Приметимо да ако је $d > 1$ онда је $\mathbf{F}_{2^d} \neq \mathbf{Z}_{2^d}$ (у $\mathbf{F}_8$ је $1 + 1 = 0$, а у $\mathbf{Z}_8$ је $1 + 1 = 2$).

12 AES

12.1 Увод

Влада САД је око 1970. године покренула процес развоја алгоритма за шифровање који би се могао реализовати на чипу, који би могао бити широко коришћен и који би био сигуран. Тако је 1975. године прихваћен алгоритам DES (Data Encryption Standard) фирме IBM. DES је симетрични шифарски систем са 56-битним кључем који 64-битни отворени текст трансформише у 64-битни шифрат. Кључ дужине 56 бита је већ око 1995. године омогућавао разбијање ове шифре, без обзира на њен квалитет. Због тога се данас користи тзв. троструки DES, систем који подразумева примену алгоритма DES три пута (са два различита кључа — први, други, први, тј. укупно 112 бита кључа) за шифровање 64-битних отворених текстова. Занимљиво је да двоструки DES са два различита кључа није много сигурнији од основног алгоритма са једним кључем, у шта ћемо се уверити у делу курса о криптоанализи (напад "сусрет на пола пута, meet-in-the-middle"). Medjutim, DES није био пројектован са намером да се користи његова трострука верзија; сигурно је да постоји ефикаснији алгоритам по нивоу сигурности еквивалентан троструком DES. Због тога је 1997. године расписан конкурс за нови алгоритам. Нови стандард (AES, Advanced encryption standard) је 2001. године постао алгоритам Рајндол (Rijndael) са блоком величине 128 бита и кључем дужине 128, 192 или 256 бита. Рајндол је симетрична блоковска шифра коју су конструисали белгијанци Joan Daemen и Vincent Rijmen.

12.2 Упрошћени AES

Упознаћемо се најпре са поједностављеном верзијом алгоритма AES (у даљем тексту SAES) коју је конструисао Е. Шафер са своја два бивша студента 2002. године и објавио га у часопису Cryptologia 2003. године. Са линеарном и диференцијалном анализом — нападима на AES упознаћемо се у делу курса о криптоанализи.

12.2.1 Коначно поље

Алгоритми проширивања кључа и шифровања у SAES користе табелу S (S-box), чија структура се описује коришћењем коначног поља $\mathbf{F}_{16} = \mathbf{F}_2[x]/(x^4 + x + 1)$ од 16 елемената. Реч *нибл* означава четворку бита, нпр. 1011. Ниблу $b_0b_1b_2b_3$ може се придружити елеменат $b_0x^3 + b_1x^2 + b_2x + b_3$ поља $\mathbf{F}_{16}$.

12.2.2 Табела S

Табела S је бијективно пресликавање $S : \{0,1\}^4 \to \{0,1\}^4$ ниблова у ниблове. Ова функција је композиција два пресликавања.

- Прво од њих је инверзија нибла у $\mathbf{F}_{16}$. На пример, инверз полинома $x+1$ је полином x^3+x^2+x, па прва компонента пресликавања S пресликава 0011 у 1110. Нибл 0000 је изузетак — није инвертибилан, па се пресликава у себе самог.

- Ниблу $N = b_0b_1b_2b_3$ добијеном инверзијом придружује елеменат $N(y) = b_0y^3 + b_1y^2 + b_2y + b_3$ **прстена** $\mathbf{F}_2[y]/(y^4+1)$ (приметимо да $y^4+1 = (y+1)^4$ није несводљив полином, па је $\mathbf{F}_2[y]/(y^4+1)$ прстен, а не поље — у њему 0 није једини елеменат без инверза). Нека је $a(y) = y^3+y^2+1$ и $b(y) = y^3+1$ у $\mathbf{F}_2[y]/(y^4+1)$. Множење у овом прстену је слично као у F_{16}, изузев што се ради по модулу y^4+1, па је $y^4 = 1$, $y^5 = y$ и $y^6 = y^2$.

- Друга компонента пресликавања S је трансформација нибла $N(y)$ у нибл $a(y)N(y) + b(y)$. Тако се на пример нибл $1110 = y^3+y^2+y$ пресликава у нибл 1011, јер је

$$\begin{aligned} & (y^3+y^2+1)(y^3+y^2+y) + (y^3+1) = \\ = \; & (y^6+y^5+y^4) + (y^5+y^4+y^3) + (y^4+y^3+y^2) + (y^3+1) \\ = \; & y^2+y+1+y^1+1+y^3+1+y^3+y^2+y^3+1 \\ = \; & 3y^3+2y^2+3y+3 = y^3+y+1 \end{aligned}$$

Према томе, $S(0011) = 1011$.

Приметимо да, пошто $\mathbf{F}_2[y]/(y^4+1)$ није поље, нису сви његови елементи инвертибилни. Међутим, полином $a(y)$ јесте инвертибилан. У литератури се друга компонента пресликавања S обично зове афино матрично пресликавање.

Пресликавање S може се приказати табелом,

nib	$S(nib)$	nib	$S(nib)$
0000	1001	1000	0110
0001	0100	1001	0010
0010	1010	1010	0000
0011	1011	1011	0011
0100	1101	1100	1100
0101	0001	1101	1110
0110	1000	1110	1111
0111	0101	1111	0111

или краће (замењујући ниблове одговарајућим декадним бројевима)

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

Елементи ове матрице су редом по врстама $S(0)$, $S(1), \ldots,$ $S(15)$. На пример, нибл $0000 = 0$ пресликава се у нибл $9 = 1001$, $0001 = 1 \to 4 = 0100$, $\ldots$, $0100 = 4 \to 13 = 1101$, итд. Бинарна верзија табеле је кориснија за ручно извршавање алгоритма SAES.

12.2.3 Проширивање кључа

Алгоритам SAES има 16-битни кључ $k_0k_1 \ldots k_{15}$. Од њега се формира низ од 48 бита (три 16-то битна *поткључа*), (од којих су првих 16 једнаки оригиналном кључу) процесом *проширивања кључа*.

Ако су N_0 и N_1 ниблови, нека N_0N_1 означава њихову конкатенацију. Нека су константе $RC[i]$ дефинисане изразом $RC[i] = x^{i+2} \in \mathbf{F}_{16}$. У алгоритму SAES се користе константе $RC[1] = x^3 = 1000$ и $RC[2] = x^4 = x{+}1 = 0011$, односно константни бајтови $RCON[i] = RC[i]0000$: $RCON[1] = 10000000$ и $RCON[2] = 00110000$. Нека су функције $RotNib$ и $SubNib$ дефинисане изразима

$$RotNib(N_0N_1) = N_1N_0,$$

односно

$$SubNib(N_0N_1) = S(N_0)S(N_1);$$

ове две функције пресликавају бајтове у бајтове. Имена ових функција одговарају њиховој природи (ротација ниблова, односно супституција ниблова применом S). Дефинишимо сада низ бајтова W. Бити бајтова $W[0]$, односно $W[1]$, су првих, односно других осам бита кључа. Остали чланови низа $W[i]$, $2 \leq i \leq 5$, дефинишу се рекурентном релацијом

$$W[i] = \begin{cases} W[i-2] \oplus RCON(i/2) \oplus SubNib(RotNib(W[i-1])), & i \equiv 0 \pmod 2 \\ W[i-2] \oplus W[i-1]), & i \not\equiv 0 \pmod 2 \end{cases}.$$

Нека су бити садржани у члановима низа W означени са $k_0, \ldots, k_{47}$. За $0 \leq i \leq 2$ нека је $K_i = W[2i]W[2i+1]$. Према томе, $K_0 = k_0 \ldots k_{15}$, $K_1 = k_{16} \ldots k_{31}$ и $K_2 = k_{32} \ldots k_{47}$. За $i \geq 1$ K_i је поткључ који се користи на крају i-те рунде; K_0 се користи пре прве рунде. Као и раније, $\oplus$ означава сабирање по модулу два, бит по бит.

Пример проширивања кључа

Нека је кључ 0101 1001 0111 1010. Према томе, $W[0] = 0101\ 1001$ и $W[1] = 0111\ 1010$. Приликом израчунавања $W[2]$, пошто је индекс 2 паран, примењује се $RotNib(W[1]) = 1010\ 0111$, па $SubNib(1010\ 0111) = 0000\ 0101$. Добијени резултат се сабира са $W[0] \oplus RCON(1)$, и добија се $W[2]$.

	0000	0101
	0101	1001
$\oplus$	1000	0000
	1101	1100

Преме томе, $W[2] = 11011100$. Наредни члан са непарним индексом израчунава се на једноставнији начин, $W[3] = W[1] \oplus W[2] = 0111\ 1010 \oplus 1101\ 1100 = 1010\ 0110$. При израчунавању $W[4]$ примењује се $RotNib(W[3]) = 0110\ 1010$, па $SubNib(0110\ 1010) = 1000\ 0000$. Добијени резултат се сабира са $W[2] \oplus RCON(2)$, и добија се $W[4]$.

	1000	0000
	1101	1100
$\oplus$	0011	0000
	0110	1100

Преме томе, $W[4] = 01101100$. На крају, $W[5] = W[3] \oplus W[4] = 1010\ 0110 \oplus 0110\ 1100 = 1100\ 1010$.

12.2.4 Алгоритам SAES

Алгоритам SAES трансформише 16-битне отворене текстове у 16-битне шифрате, користећи проширени кључ $k_0 \dots k_{47}$. Алгоритам шифровања је композиција осам функција које се редом примењују на отворени текст:

$$A_{K_2} \circ SR \circ NS \circ A_{K_1} \circ MC \circ SR \circ NS \circ A_{K_0}$$

(најпре се примењује функција A_{K_0} која је на десној страни овог израза). Свака од ових функција примењује се на стање; стање је четворка ниблова, видети слику 1. Почетно стање је отворени текст, слика 2, а завршно стање је шифрат, слика 3.

$b_0b_1b_2b_3$	$b_8b_9b_{10}b_{11}$
$b_4b_5b_6b_7$	$b_{12}b_{13}b_{14}b_{15}$

Слика 1.

$p_0p_1p_2p_3$	$p_8p_9p_{10}p_{11}$
$p_4p_5p_6p_7$	$p_{12}p_{13}p_{14}p_{15}$

Слика 2.

$c_0c_1c_2c_3$	$c_8c_9c_{10}c_{11}$
$c_4c_5c_6c_7$	$c_{12}c_{13}c_{14}c_{15}$

Слика 3.

Дефиниције примењених функција:

- Функција A_{K_i} (*add key*) представља сабирање стања са K_i по модулу два, бит по бит, тако да се индекси бита стања и бита кључа слажу по модулу 16.
- Функција NS (*nibble substitution*) замењује сваки нибл N_i из стања ниблом $S(N_i)$, не мењајући редослед ниблова. Другим речима, стање $\begin{array}{|c|c|}\hline N_0 & N_2 \\ \hline N_1 & N_3 \\ \hline \end{array}$ трансформише се у стање $\begin{array}{|c|c|}\hline S(N_0) & S(N_2) \\ \hline S(N_1) & S(N_3) \\ \hline \end{array}$
- Функција SR (*shift row*) стање $\begin{array}{|c|c|}\hline N_0 & N_2 \\ \hline N_1 & N_3 \\ \hline \end{array}$ трансформише се у стање $\begin{array}{|c|c|}\hline N_0 & N_2 \\ \hline N_3 & N_1 \\ \hline \end{array}$.
- Функција MC (*mix column*): Колони $\begin{array}{|c|}\hline N_i \\ \hline N_j \\ \hline \end{array}$ у стању одговара елеменат

$N_i z+N_j$ прстена $\mathbf{F}_{16}[z]/(z^2+1)$. Тако колони $\begin{array}{|c|}\hline 1010 \\ \hline 1001 \\ \hline \end{array}$ одговара елеменат $(x^3+x)z+(x^3+1)$. Овде је $\mathbf{F}_{16}[z]$ скуп полинома по z са коефицијентима из $\mathbf{F}_{16}$. Према томе $\mathbf{F}_{16}[z]/(z^2+1)$ подразумева да се полиноми посматрају по модулу z^2+1; због тога је $z^2=1$. Скуп представника класа еквиваленције састоји се од 16^2 полинома по z степена мањег од 2. Функција MC множи сваку колону полиномом $c(z)=x^2z+1$. У овом примеру

$$\begin{aligned} & [((x^3+x)z+(x^3+1))](x^2z+1) = \\ = \;& (x^5+x^3)z^2+(x^3+x+x^5+x^2)z+(x^3+1) \\ = \;& (x^5+x^3+x^2+x)z+(x^5+x^3+x^3+1) \\ = \;& (x^2+x+x^3+x^2+x)z+(x^2+x+1) \\ = \;& (x^3)z+(x^2+x+1), \end{aligned}$$

што одговара колони $\begin{array}{|c|}\hline 1000 \\ \hline 0111 \\ \hline \end{array}$. Приметимо да полином $z^2+1=(z+1)^2$ није несводљив над $\mathbf{F}_{16}$, па $\mathbf{F}_{16}[z]/(z^2+1)$ није поље (тј. нису инвертибилни сви његови елементи); међутим, полином $c(z)$ јесте инвертибилан. Уопште, после множења по модулима z^2+1, x^4+x+1 и 2 добија се

$$\begin{aligned} & \left((b_0x^3+b_1x^2+b_2x+b_3)z+(b_4x^3+b_5x^2+b_6x+b_7)\right)(x^2z+1) \equiv \\ \equiv \;& (b_1+b_7)+(b_0+b_1+b_6)x+(b_0+b_3+b_5)x^2+(b_2+b_4)x^3+ \\ + \;& ((b_3+b_5)+(b_2+b_4+b_5)x+(b_1+b_4+b_7)x^2+(b_0+b_6)x^3)z \end{aligned}$$

што значи да функција MC трансформише колону

$\begin{array}{|c|}\hline b_0b_1b_2b_3 \\ \hline b_4b_5b_6b_7 \\ \hline \end{array}$ у колону $\begin{array}{|cccc|}\hline b_0\oplus b_6 & b_1\oplus b_4\oplus b_7 & b_2\oplus b_4\oplus b_5 & b_3\oplus b_5 \\ \hline b_2\oplus b_4 & b_0\oplus b_3\oplus b_5 & b_0\oplus b_1\oplus b_6 & b_1\oplus b_7 \\ \hline \end{array}$

Mathematica:

$$\begin{aligned} A &= ((b0x^3+b1x^2+b2x+b3)z+(b4x^3+b5x^2+b6x+b7))(x^2z+1) \\ B &= Collect[PolynomialMod[A,\{z^2+1,x^4+x+1,2\}],\{z,x\}] \\ &= b1+b7+(b0+b1+b6)x+(b0+b3+b5)x^2+(b2+b4)x^3+ \\ &+ (b3+b5+(b2+b4+b5)x+(b1+b4+b7)x^2+(b0+b6)x^3)z \end{aligned}$$

Композиција функција $A_{K_i}\circ MC\circ SR\circ NS$ је i-та *рунда* алгоритма шифровања; алгоритам SAES има две рунде. Поред тога, примењује се допунска трансформација A_{K_0} пре прве рунде, а последња рунда нема трансформацију MC; ова чињеница биће објашњена у следећем одељку.

12.2.5 Дешифровање

Приметимо да за произвољне функције (за које је дефинисана композиција и инверзне функције) важи $(f \circ g)^{-1} = g^{-1} \circ f^{-1}$. Поред тога, ако је композиција функције са самом собом идентично пресликавање, онда је функција сама себи инверзна; тада се каже да је она инволуција. Све функције A_{K_i} су инволуције. Иако је и SR инволуција код SAES , због тога што она није инволуција код AES израз SR^{-1} неће бити упрошћаван. Дешифровање је према томе дефинисано композицијом

$A_{K_0} \circ NS^{-1} \circ SR^{-1} \circ MC^{-1} \circ A_{K_1} \circ NS^{-1} \circ SR^{-1} \circ A_{K_2}$

Да се изврши пресликавање NS^{-1} нибл се множи са $a(y)^{-1} = y^2 + y + 1$, па се резултату додаје $a(y)^{-1}b(y) = y^3 + y^2$ у прстену $\mathbf{F}_2[y]/(y^4 + 1)$. После тога нибл се инвертује у $\mathbf{F}_{16}$. Уместо свега тога, може се користити унапред израчуната табела функције S^{-1}.

Пошто је MC множење са $c(z) = x^2z + 1$, функција MC^{-1} је множење са $c(z)^{-1} = xz + (x^3 + 1)$ у $\mathbf{F}_{16}[z]/(z^2 + 1)$.

Дешифровање се може обавити на горе описани начин. Сада ћемо видети разлог зашто у последњој рунди нема функције MC . Приметимо најпре да је $NS^{-1} \circ SR^{-1} = SR^{-1} \circ NS^{-1}$. Нека St означава неко стање. Тада је $MC^{-1}(A_{K_i}(St)) = MC^{-1}(K_i \oplus St) = c(z)^{-1}(K_i \oplus St) = c(z)^{-1}(K_i) \oplus c(z)^{-1}(St) = c(z)^{-1}(K_i) \oplus MC^{-1}(St) = A_{c(z)^{-1}(K_i)}(MC^{-1}(St))$. Према томе, $MC^{-1}(A_{K_i}(St)) = A_{c(z)^{-1}(K_i)} \circ MC^{-1}(St)$.

Шта означава $c(z)^{-1}(K_i)$? Нека су $b_0b_1 \dots b_7$, $b_8b_9 \dots b_{15}$ два бајта од којих се састоји K_i. Први бајт

$\begin{array}{|c|}\hline b_0b_1b_2b_3 \\ \hline b_4b_5b_6b_7 \\ \hline \end{array}$ можемо сматрати елементом $\mathbf{F}_{16}[z]/(z^2 + 1)$. Њега множимо са $c(z)^{-1}$, па га поново претварамо у бајт. Исто се ради и са $b_8 \dots b_{15}$. Према томе, $c(z)^{-1}(K_i)$ има 16 бита. Израз $A_{c(z)^{-1}(K_i)}$ означава сабирање по модулу два $c(z)^{-1}K_i$ са текућим стањем. Приметимо да се приликом извршавања MC^{-1} стање множи са $c(z)^{-1}$ (или још једноставније, користи се еквивалентна табела, чији садржај треба израчунати за домаћи задатак). Да се изврши $A_{c(z)^{-1}(K_i)}$ најпре се K_i множи са $c(z)^{-1}$ (или још једноставније, користи се еквивалентна табела, чији садржај треба израчунати за домаћи задатак), а онда се резултат сабира по модулу два са текућим стањем.

Дешифровање се може вршити такође применом композиције

$$A_{K_0} \circ SR^{-1} \circ NS^{-1} \circ A_{c(z)^{-1}K_1} \circ MC^{-1} \circ SR^{-1} \circ NS^{-1} \circ A_{K_2}.$$

Подсетимо се да се шифровање врши на основу

$$A_{K_2} \circ SR \circ NS \circ A_{K_1} \circ MC \circ SR \circ NS \circ A_{K_0}$$

Другим речима, функције се у току дешифровања појављују истим редом као и при шифровању, изузев што се поткључеви примењују обрнутим редоследом. За оригинални AES ово може да олакша имплементацију. Ово не би било могуће да се MC појављује у последњој рунди.

12.2.6 Пример шифровања

Нека је кључ исти као у претходном примеру, 0101 1001 0111 1010. Према томе, $W[0] = 0101\ 1001$, $W[1] = 0111\ 1010$, $W[2] = 1101\ 1100$, $W[3] = 1010\ 0110$, $W[4] = 0110\ 1100$ и $W[5] = 1100\ 1010$. Нека је отворени текст "Ed" кодирано ASCII кодом, 01000101 01100100. Тада је почетно стање (водећи рачуна да ниблови иду редом у горњи леви, **затим доњи леви**, па горњи десни, па доњи десни угао)

0100	0110
0101	0100

Најпре примењујемо A_{K_0} (као што смо рекли, $K_0 = W[0]W[1]$); ново стање је:

	0100		0110				0001	0001
$\oplus$	0101	$\oplus$	0111		=		1100	1110
	0101		0100					
$\oplus$	1001	$\oplus$	1010					

Применом NS и SR добија се

0100	0100	$\rightarrow SR \rightarrow$	0100	0100
1100	1111		1111	1100

Применивши MC, добијамо

1101	0001
1100	1111

Затим се примењује A_{K_1}, при чему је $K_1 = W[2]W[3]$.

	1101		0001				0000	1011
$\oplus$	1101	$\oplus$	1010		=		0000	1001
	1100		1111					
$\oplus$	1100	$\oplus$	0110					

Применом NS и SR добија се

1001	0011	$\rightarrow SR \rightarrow$	1001	0011
1001	0010		0010	1001

Затим се примењује A_{K_2}, при чему је $K_2 = W[4]W[5]$.

	1001		0011				1111	1111
$\oplus$	0110	$\oplus$	1100		=		1110	0011
	0010		1001					
$\oplus$	1100	$\oplus$	1010					

Према томе, шифрат је 11111110 11110011.

12.3 Комплетан AES

Због једноставности описаћемо верзију AES са 128-битним кључем и 10 рунди. Као што смо рекли, AES обрађује 128-битне блокове. Описаћемо разлике у односу на упрошћену верзију. Свако стање састоји се од матрице бајтова димензије 4×4 .

Коначно поље у коме се рачуна је $\mathbf{F}_{2^8} = \mathbf{F}_2[x]/(x^8 + x^4 + x^3 + x + 1)$. Бајту $b_0b_1b_2b_3b_4b_5b_6b_7$ одговара елеменат $b_0x^7 + \cdots + b_7 \in \mathbf{F}_{2^8}$. Табела S најпре инвертује бајт у $\mathbf{F}_{2^8}$, па га онда множи са $a(y) = y^4 + y^3 + y^2 + y + 1$ и резултату додаје $b(y) = y^6 + y^5 + y + 1$ у прстену $\mathbf{F}_2[y]/(y^8 + 1)$. Инверз полинома $a(y)$ је $a(y)^{-1} = y^6 + y^3 + y$; поред тога, $a(y)^{-1}b(y) = y^2 + 1$.

Функција *ByteSub* у AES је очигледно уопштење функције *SubNib* — она сваки бајт B замењује његовом сликом $S(B)$. Функција $ShiftRow$ циклички помера врсте улево за $0, 1, 2, 3$. Према томе, она стање

B_0	B_4	B_8	B_{12}
B_1	B_5	B_9	B_{13}
B_2	B_6	B_{10}	B_{14}
B_3	B_7	B_{11}	B_{15}

преводи у стање

B_0	B_4	B_8	B_{12}
B_5	B_9	B_{13}	B_1
B_{10}	B_{14}	B_2	B_6
B_{15}	B_3	B_7	B_{11}

Функција $MixColumn$ множи колону полиномом $c(z) = (x+1)z^3 + z^2 + z + x$ у $\mathbf{F}_{2^8}[z]/(z^4+1)$. Инверзни полином је $c(z)^{-1} = (x^3+x+1)z^3 + (x^3 + x^2+1)z^2 + (x^3+1)z + (x^3+x^2+x)$. Функција $MixColumn$ појављује се у свим рундама, сем у последњој. Функција *AddRoundKey* је очигледно уопштење A_{K_i}. Допунска функција *AddRoundKey* са поткључем за рунду 0 примењује се на почетку шифровања.

Проширивање кључа ради са низом W чији чланови имају по четири бајта. Кључ попуњава $W[0], \ldots, W[3]$. Функција *RotByte* циклички ротира групу од четири бајта за један бајт улево, слично као што се ради са другом врстом у $ShiftRow$. Функција *ByteSub* примењује функцију (табелу) S на сваки бајт. Као и раније, $RC[i] = x^i$ у $\mathbf{F}_{2^8}$, а $RCON[i]$ је конкатенација $RC[i]$ и три бајта од свих нула. За $4 \leq i \leq 43$ рачуна се

$$W[i] = \begin{cases} W[i-4] \oplus RCON(i/4) \oplus ByteSub(RotByte(W[i-1])), & i \equiv 0 \pmod 4 \\ W[i-4] \oplus W[i-1]), & i \not\equiv 0 \pmod 4 \end{cases}.$$

Поткључ K_i састоји се од бита у члановима низа $W[4i] \ldots W[4i+3]$.

12.4 AES као комбинована шифра

Премештање се врши применом $ShiftRow$. Иако то не може да буде произвољна премутација, даља дисперзија постиже се премештањем колона у $MixColumn$. Замена се постиже применом *ByteSub*, а *AddRoundKey* чини алгоритам зависним од кључа.

12.5 Начини коришћења блоковских шифри

Постоје четири уобичајена начина коришћења AES (односно било које блоковске шифре). Стандардни начин је ECB (electronic code book). Ако је p_i i-ти блок отвореног текста, c_i одговарајући блок шифрата, а $E_k(\cdot)$ означава примену блоковске шифре, онда је $c_i = E_k(p_i)$, $i = 0, 1, \ldots$. Дешифровање се врши на основу израза $p_i = E_k^{-1}(p_i)$, $i = 0, 1, \ldots$. ECB је најједноставнији начин употребе; међутим, његов недостатак је да за исти кључ два иста отворена текста дају исте шифрате. Ако број бита није умножак 128, онда се на крају поруке додају знаци (који обично почињу са 1, јер обични ASCII знаци почињу нулом) тако да и последњи блок има дужину 128 бита.

Други (најчешћи) начин коришћења је CBC (cipher block chaining). Алиса и Бобан морају унапред да се договоре који ће (случајни, не-тајни)

иницијализациони вектор (IV) користити; уствари, пре сваке поруке шаље се одговарајући (нови) иницијализациони вектор. Резултат шифровања је $c_i = E_k(c_{i-1} \oplus p_i)$, $i \geq 0$. При томе се за шифровање првог блока p_0 користи иницијализациони вектор, односно узима се да је $c_{-1} = IV$. Дешифровање: $p_i = c_{i-1} \oplus E_k^{-1}(c_i)$.

Трећи начин коришћења је CFB (cipher feedback). Он такође подразумева употребу иницијализационог вектора IV, који се мора мењати за сваки наредни шифрат. Шифровање, односно дешифровање описују се изразима $c_i = E_k(c_{i-1}) \oplus p_i$, $i \geq 0$, односно $p_i = E_k(c_{i-1}) \oplus c_i$, $i \geq 0$. При томе се може користити нпр. само нпр. најнижи бајт израза $E_k(c_{i-1})$, тј. може се шифровати/дешифровати нпр. низ бајтова. За разлику од тога, у режиму CBC мора се сачекати комплетирање блока од 128 бита OT пре почетка израчунавања шифрата.

Последњи начин коришћења је OFB (output feedback). То је уствари начин да се добије низ кључа за шифровање. $Z_i = E_k(Z_{i-1}) = E_k^{i+1}(IV)$. Низ кључа добија се конкатенацијом $Z_0 Z_1 Z_2 \ldots$. Шифрат се добија сабирањем са низом кључа, $c_i = p_i \oplus Z_i$, $i \geq 0$, па је дешифровање $p_i = c_i \oplus Z_i$.

12.6 Анализа упрошћеног алгоритма AES

Размотрићемо могуће нападе на SAES у режиму ECB.

Цица је дошла до пара (OT, CT) и жели да одреди кључ. Нека је OT $p_0 p_1 \ldots p_{15}$, одговарајући CT $c_0 c_1 \ldots c_{15}$, а кључ $k_0 k_1 \ldots k_{15}$.

Може се формирати 16 једначина облика

$$f_i(p_0 p_1 \ldots p_{15}, k_0 k_1 \ldots k_{15}) = c_i$$

где је f_i полином од 32 променљиве са коефицијентима из $\mathbf{F}_2$, за који се очекује да има просечно 2^{31} чланова. Када се фиксирају (замене) бити c_j, p_j (из пара одговарајућих OT, CT), добијамо 16 нелинеарних једначина са 16 непознатих (бити k_i). Полиноми у овим једначинама у просеку имају по 2^{15} чланова.

У алгоритму SAES је све линеарно, изузев табеле S, коју ћемо сада размотрити. Означимо улазни нибл са $abcd$, а излазни нибл са $efgh$. Тада се трансформација коју врши табела S може описати следећим једначинама:

$$\begin{aligned} e &= acd + bcd + ab + ad + cd + a + d + 1 \\ f &= abd + bcd + ab + ac + bc + cd + a + b + d \\ g &= abc + abd + acd + ab + bc + a + c \\ h &= abc + abd + bcd + acd + ac + ad + bd + a + c + d + 1 \end{aligned}$$

при чему су сва сабирања по модулу два. На пример, да се добије израз за e, полази се од израза

$$e = \overline{a} f_0(b, c, d) + a f_1(b, c, d),$$

где су f_0, односно f_1 функције од три променљиве b, c, d одређене првом, односно другом половином табеле истинитости функције e. Заменом $\overline{a} = 1 + a$ добија се

$$e = f_0(b, c, d) + a(f_0(b, c, d) + f_1(b, c, d)).$$

Настављајући овакво разлагање редом за променљиве b, c, d, добија се наведени израз за функцију e, а исто тако и за функције f, g и h. Наизменично смењивање линеарних са овим нелинеарним пресликавањима као резултат даје врло сложени полиномијални израз за бите шифрата преко бита кључа за познате бите отвореног текста.

За решавање система линеарних једначина са много непознатих је лако, јер се знају алгоритми полиномијалне сложености. С друге стране, нису познати алгоритми за ефикасно решавање нелинеарних полиномијалних једначина са много непознатих.

12.7 Објашњење конструкције AES

За квалитет шифровања постоје два основна критеријума, *сигурност* и *ефикасност*. Приликом пројектовања AES , аутори су водили рачуна о њима. Они су поред тога уградили у алгоритам *једноставност* и *понављање*. Сигурност се мери отпорношћу шифровања на све познате нападе. Ефикасност је комбинација брзине шифровања/дешифровања и мере у којој алгоритам користи ресурсе (потребан простор на чипу за хардверску реализацију, односно потребна меморија за софтверску реализацију). Једноставност се односи на сложеност појединих корака, као и целине алгоритма за шифровање. Ако се алгоритам може лако разумети, вероватније је да ће реализације алгоритма бити коректне. На крају, понављање се односи на вишеструку употребу функција у оквиру алгоритма.

У наредне две тачке размотрићемо сигурност, ефикасност, једноставност и понављање у алгоритму AES .

12.8 Сигурност

Као стандард за шифровање, AES мора да буде отпоран на све познате криптоаналитичке нападе. Према томе, приликом пројектовања алгоритма водило се пре свега рачуна да он буде отпоран на нападе, посебно на диференцијалну и линеарну криптоанализу. Да би блоковска шифра била сигурна, она мора да обезбеђује *дифузију* и *нелинеарност*.

Дифузија се дефинише као ширење утицаја бита у току шифровања. Потпуна дифузија значи да сваки бит стања зависи од свих бита претходног стања. У алгоритму AES две узастопне рунде обезбеђују потпуну дифузију. Функције $ShiftRow$, $MixColumn$ и проширивање кључа обезбеђују дифузију, неопходну да систем буде отпоран на познате нападе.

Нелинеарност у алгоритму потиче од табеле S, која се користи у $ByteSub$ и проширивању кључа. Поред осталог, нелинеарност је проузрокована инверзијом

у коначном пољу. То није линеарно пресликавање бајтова у бајтове. Овде се под линеарним (афиним, прецизније) пресликавањем подразумева пресликавање бајтова (тј. осмодимензионалних вектора на пољем $\mathbf{F}_2$) у бајтове које се може представити као множење матрицом 8×8 и додавањем вектора.

Нелинеарност повећава отпорност шифре на криптоаналитичке нападе. Нелинеарност у проширивању кључа обезбеђује да познавање дела кључа или поткључа не омогућује једноставно одређивање пуно других бита кључа.

Једноставност повећава кредибилност шифре на следећи начин. Коришћење једноставних корака сугерише да је једноставно разбити шифру, па то покушава да уради више људи. Кад се много таквих покушаја оконча неуспехом, шифра улива веће поверење.

Иако понављање има много предности, оно може учинити шифру осетљивијом на неке нападе. Конструкција AES обезбеђује да понављање не изазива смањење сигурности. На пример, коришћење константи $RCON[i]$ разбија потенцијалне сличности између поткључева.

12.9 Ефикасност

Очекује се да се AES извршава на рачунарима и уређајима различите величине и снаге. Због тога је алгоритам пројектован тако да се може ефикасно извршавати на многим платформама, од стоних рачунара до малих уређаја који се могу сместити у прикључак на мрежу.

Понављање у структури AES омогућује да се паралелном реализацијом убрза шифровање/дешифровање. Сваки корак може се због понављања разбити на независна израчунавања. Функције $MixColumn$ и $ShiftRow$ независно обрађују поједине колоне, односно врсте стања. Функција $AddKey$ може се паралелизовати на више начина.

Поклапање редоследа корака за шифровање и дешифровање омогућује да се исти чип користи и за шифровање и за дешифровање. То смањује цену хардвера и повећава брзину.

Једноставност алгоритма олакшава његово објашњавање, па су због тога реализације једноставне и поуздане. Коефицијенти полинома који дефинише $\mathbf{F}_{256}$ изабрани су тако да минимизирају израчунавање.

Упоређење AES и RC4. Сматра се да су блоковске шифре флексибилније, тиме што омогућују шифровање на више различитих начина. Блоковска шифра може се лако употребити као проточна, али обрнуто није могуће. У једној од имплементација шифра RC4 је 1.77 пута бржа од AES , али се сматра да је мање сигурна.

13 Напади на блоковске шифре

13.1 Потпуна претрага и сусрет у средини

Потпуна претрага је напад са познатим отвореним текстом. Цица има на располагању пар (OT,CT). Претпоставимо да је дужина кључа b бита. Она

шифрује отворени текст са свих 2^b различитих кључева и проверава који од њих даје задати шифрат. Вероватно је да ће само неколико кључева да испуни овај услов. Ако постоји више од једног кандидата, онда њих Цица проверава на другом пару (OT,CT); тада ће вероватно преостати само један кандидат. Она очекује успех у просеку после пола прегледаних кључева. Према томе, за разбијање DES -а потребно је 2^{55} покушаја а за разбијање AES -а — 2^{127} покушаја.

Напад *сусрет на пола пута* (meet-in-the-middle) на блоковске шифре открили су Дифи (Diffie) и Хелман (Hellman). То је такође напад са познатим отвореним текстом. Нека E_{K_1} означава шифровање кључем K_1, применом одређене блоковске шифре. Тада је $CT = E_{K_2}(E_{K_1}(PT))$ (где је PT отворени текст, а CT шифрат) двострука блоковска шифра.

Цеци је потребно да зна два пара отворени текст/шифрат. Она шифрује отворени текст PT свим могућим кључевима K_1 и резултате сортиране према шифрату смешта на диск. Она затим дешифрује одговарајући шифрат CT свим кључевима K_2 и такође смешта на диск резултате, сортиране према отвореном тексту. Она затим проналази парове $(k_{1,i}, k_{2,i})$ такве да је $E_{k_{1,i}}(PT_1) = D_{k_{2,i}}(CT_1)$ (овде D означава дешифровање). За ово је довољан пролаз без враћања кроз два припремљена списка. За сваки овакав пар она израчунава $E_{k_{2,i}}(E_{k_{1,i}}(PT_2))$ и то упоређује са CT_2. Вероватно је да ће само један пар кључева испунити овај услов. Овакав напад захтева огроман меморијски простор. Међутим, број корака је око $b2^{b-1}$ (због сортирања), па двострука шифра није битно сигурнија од једноструке шифре са једним кључем.

Може се показати да због напада са сусретом на средини троструки DES са три кључа није битно сигурнији од троструког DES -а са два кључа. Пошто је два кључа лакше разменити него три, троструки DES обично се користи са два кључа.

Два друга важна напада на блоковске шифре су линеарна и диференцијална криптоанализа; они ће бити размотрени у делу о криптоанализи.

14 Степеновање поновљеним квадрирањем

Као што смо видели, ако је $\mathrm{nzd}(a, m) = 1$ и $b \equiv c \pmod{\varphi(m)}$, онда је $a^b \equiv a^c \pmod{m}$. Према томе, приликом израчунавања $a^b \pmod{m}$ кад је $\mathrm{nzd}(a, m) = 1$ и $b \geq \varphi(m)$, први корак је свођење $b \mod \varphi(m)$.

Да би се израчунало $a^b \mod m$ кад је $b < \varphi(m)$, али је b још увек велико, може се искористити калкулатор. Израчунајмо на пример $87^{43} \mod 103$. Број 43 најпре представљамо у облику збира степенова двојке: $43 = 32+8+2+1$ (тако што сваки пут одузмемо највећи могући степен двојке). Према томе, $87^{43} = 87^{32+8+2+1} = 87^{32}87^{8}87^{2}87^{1}$. Затим рачунамо $87 \equiv 87 \mod 103$, $87^2 \equiv 50$, $87^4 \equiv 50^2 \equiv 28$, $87^8 \equiv 28^2 \equiv 63$, $87^{16} \equiv 63^2 \equiv 55$, $87^{32} \equiv 55^2 \equiv 38$. Дакле, $87^{43} \equiv 38 \cdot 63 \cdot 50 \cdot 87 \equiv 85 \pmod{103}$.

На рачунару се то ради мало друкчије. Да се израчуна $b^n \pmod{m}$ најпре се n представи у систему са основом 2, $n = n_k 2^k + n_{k-1}2^{k-1} + \cdots + n_0 =$

$(n_k n_{k-1} \dots n_0)_2$, $n_i \in \{0,1\}$. Нека је a делимичан производ. На почетку је $a = 1$.

StepenovanjeKvadriranjem(b, n)
  $b_0 \leftarrow n$ $\quad \{\ b_0 = n^{2^0} = n\}$
  if $n_0 = 1$ **then** $a \leftarrow 1$ **else** $a \leftarrow n$
  for $j \leftarrow 1$ **to** k **do**
    $b_j \leftarrow b_{j-1}^2 \pmod m$ $\quad \{\ b_j = n^{2^j}\}$
    if $n_i = 1$ **then**
      $a \leftarrow a \cdot b_{j-1}$

Урадимо још једном претходни пример, $b = 87$, $n = 43$. У систему са основом 2 је $43 = 101011$, тј. $n_0 = 1$, $n_1 = 1$, $n_2 = 0$, $n_3 = 1$, $n_4 = 0$, $n_5 = 1$.

$$
\begin{array}{llllllllllll}
a = 1 & & & (n_0 = 1) & a & = & 87 & & & & & \\
87^2 & \equiv & 50 & (n_1 = 1) & a & = & 50 & \cdot & 87 & \equiv & 24 & (\equiv 87^2 \cdot 87^1) \\
50^2 & \equiv & 28 & (n_2 = 0) & a & = & 24 & & & & & \\
28^2 & \equiv & 63 & (n_3 = 1) & a & \equiv & 63 & \cdot & 24 & \equiv & 70 & (\equiv 87^8 \cdot 87^2 \cdot 87^1) \\
63^2 & \equiv & 55 & (n_4 = 0) & a & = & 70 & & & & & \\
55^2 & \equiv & 38 & (n_5 = 1) & a & = & 38 & \cdot & 70 & \equiv & 85 & (\equiv 87^{32} \cdot 87^8 \cdot 87^2 \cdot 87^1)
\end{array}
$$

15 Временска сложеност алгоритама

Логаритми значајно смањују велике бројеве. На пример, ако узмемо лист папира, па на њега ставимо други, па онда удвостручимо гомилу (четири папира), и тако даље, после 50 дуплирања гомиле имаћемо $2^{50} \approx 10^{15}$ листова папира, па би гомила порасла до Сунца. Гомила од 50 листова папира има висину око $1\,cm$.

Ако је x реалан број, са $\lfloor x \rfloor$ означавамо највећи цели број $\leq x$. Тако је $\lfloor 1.4 \rfloor = 1$ и $\lfloor 1 \rfloor = 1$. Подсетимо се како се бројеви представљају у систему са основом два. Све дотле док је могуће, одузимамо од броја највећи степен двојке. На пример, $47 \geq 32$. $47 - 32 = 15$. $15 - 8 = 7$. $7 - 4 = 3$. $3 - 2 = 1$. Дакле, $47 = 32 + \ +8 + 4 + 2 + 1 = (101111)_2$.

Други алгоритам за одређивање бинарне представе броја описује се следећим псеудокодом; претпоставља се да је број представљен са 32 бита:

BinarneCifre(n, v)
  $v \leftarrow [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]$
  $i \leftarrow 1$
  while $i < 32$
    $ostatak \leftarrow n \mod 2$
    $v[32 - i] = ostatak$
    $n \leftarrow (n - ostatak)/2$
    $i \leftarrow i + 1$

Кажемо да је 47 шестобитни број. Број цифара у бинарној представи броја N (*дужина* броја N) је број његових бита, односно $\lfloor \log_2(N) \rfloor + 1$. Сви логаритми разликују се за константни фактор (на пример, $\log_2 x = k \log_{10} x$, где је $k = \log_2 10$).

Оцена временске сложености алгоритама (односно горње границе) обично се односи на најгори случај, при чему се претпоставља да је димензија улаза велика. Размотримо сада алгоритме за неколико битских операција са бројевима. Најпре сабирање два n-битна броја $N + M$. На пример, сабирамо $219 + 242$, односно $11011011 + 11110010$, где је $n = 8$.

```
 111  1
 11011011
 11110010
---------
111001101
```

За оно што се ради у једној колони рећи ћемо да је то битска операција. То је фиксирана комбинација упоређивања и померања. Према томе, цео алгоритам садржи $n \approx \log_2 N$ битских операција. За сабирање n-битног и m-битног броја, $n \geq m$, и даље је потребно n битских операција (јер морамо да копирамо све непромењене бите из дужег броја).

Размотримо сада множење n-битног броја N са m-битним бројем M, где је $n \geq m$. На доњем дијаграму није приказано завршно сабирање.

```
   10111
    1011
   -----
   10111
  101110
10111000
```

Множење има две фазе: исписивање врста, па њихово сабирање. Прва фаза: исписује се највише m врста испод прве линије, а у свакој врсти има највише $n + m - 1$ бита. Према томе, прва фаза обухвата $m(n + m - 1)$ битских операција. Друга фаза састоји се од највише $m - 1$ сабирања, од којих се свако састоји од највише $n + m - 1$ битских операција. Дакле, сложеност друге фазе је највише $(m - 1)(n + m - 1)$. Укупан број битских операција у току множења је највише $m(n + m - 1) + (m - 1)(n + m - 1) = (2m - 1)(n + m - 1) \leq (2m)(n + m) \leq (2m)(2n) = 4mn$. Другим речима, сложеност овог алгоритма је ограничена са $4 \log_2 N \log_2 M$. (при том занемарујемо време приступа меморији и слично, што је тривијално). Време извршавања $C \cdot 4 \log_2 N \log_2 M = C' \log N \log M$ зависи од брзине рачунара.

Ако су f и g позитивне функције од природних бројева (дефинисане на $\mathbf{Z}_{>0}$, односно $\mathbf{Z}^r_{>0}$ ако зависе од више променљивих; $\mathbf{R}_{>0}$ означава скуп реалних позитивних бројева), и постоје $c > 0$, $n_0 \in \mathbf{Z}_{>0}$, такви да је $f(n) < cg(n)$ за све $n > n_0$, онда кажемо да је $f = O(g)$.

Према томе, $f = O(g)$ значи да је f ограничено константним умношком g (обично се за g бира најједноставнији израз).

Користећи ову нотацију, сложеност сабирања N и M (ако је $N \geq M$) је $O(\log N)$. Ова оцена сложености важи и за одузимање. Сложеност множења M и N је $O(\log N \log M)$. Ако су N и M истог реда величине, кажемо да је сложеност израчунавања њиховог производа $O(\log^2 N)$. Приметимо да

$$\log^2 N = (\log N)^2 \neq \log(\log N) = \log \log N.$$

Сложеност исписивања цифара броја N је $O(\log N)$.

Постоје ефикаснији алгоритми за множење, сложености

$$O(\log N \log \log N \log \log \log N).$$

На сличан начин добија се оцена сложености дељења N са M (као и израчунавања остатка $N \mod M$, односно количника): $O(\log N \log M)$.

Правила:

1. $kO(f(n)) = O(kf(n)) = O(f(n))$.
2. Нека је $p(n) = a_d n^d + a_{d-1} n^{d-1} + \cdots + a_0$ полином.
   - **а)** Тада је $p(n) = O(n^d)$. На пример, лако је показати да је $2n^2 + 5n < 3n^2$ за велике n, па је $2n^2 + 5n = O(3n^2) = O(n^2)$.
   - **б)** $O(\log(p(n))) = O(\log n)$, јер је $O(\log(p(n))) = O(\log n^d) = O(d \log n) = O(\log n)$.
3. Ако је $h(n) \leq f(n)$ онда је $O(f(n)) + O(h(n)) = O(f(n) + h(n)) = O(2f(n)) = O(f(n))$.
4. $f(n)O(h(n)) = O(f(n))O(h(n)) = O(f(n)h(n))$.

Да се изврши анализа сложености типичног алгоритма, потребно је:

а) Избројати основне кораке (оне који се извршавају највише пута).

б) Описати најспоријi основни корак.

в) Одредити горњу границу времена извршавања најспоријег основног корака.

г) Одредити горњу границу времена извршавања целог алгоритма (најчешће израчунавањем производа а) и в))

д) Одговор треба да буде облика $O(\ldots)$.

Подсетимо се, за множење $F \cdot G$, односно дељење F/G сложеност је $O(\log F \log G)$; сложеност сабирања $F + G$, односно одузимања $F - G$ је $O(\log F)$ ако је $F \geq G$.

Проблем 1. Одредити горњу границу сложености израчунавања $\mathrm{nzd}(N, M)$ Еуклидовим алгоритмом ако је $N \geq M$.

Решење: Израчунавање НЗД је најспорије ако су сви количници 1, као на пример при израчунавању $\mathrm{nzd}(21, 13)$, односно општије, при рачунању $\mathrm{nzd}(F_n, F_{n-1})$, где је F_n је n-ти Фибоначијев број ($F_1 = F_2 = 1$, $F_n = F_{n-1}+F_{n-2}$). Број дељења у Еуклидовом алгоритму је $n-3 < n$. Нека је $\alpha = (1+\sqrt{5})/2$. Тада је $F_n \approx \alpha^n$. Према томе, за Еуклидов алгоритам је најгори случај ако је $N = F_n$, $M = F_{n-1}$. Приметимо да је $n \approx \log_\alpha N$. Број дељења је $n = O(\log N)$. Сваки корак је дељење, сложености $O(\log N \log M)$. Према томе, сложеност је $O(\log N)O(\log N \log M) = O(\log^2 N \log M)$, односно после заокруживања $O(\log^3 N)$. Ако се удвостручи дужина ($= O(\log N)$) два броја, Еуклидов алгоритам ће се извршавати 8 пута дуже. Заиста, нека је време израчунавања $\mathrm{nzd}(N, M)$ једнако $k(\log N)^3$ за неку константу k. Претпоставимо да је $M_1, N_1 \approx 2^{500}$. Тада је време за израчунавање $\mathrm{nzd}(N_1, M_1)$ једнако $t_1 = k(\log 2^{500})^3 = k(500 \log 2)^3 = k \cdot 500^3 (\log 2)^3$. Ако је $M_2, N_2 \approx 2^{1000}$ (дакле два пута дужи бројеви), онда је за израчунавање $\mathrm{nzd}(N_2, M_2)$ потребно време $t_2 = k(\log 2^{1000})^3 = k(1000 \log 2)^3 = k \cdot 1000^3 (\log 2)^3 = k \cdot 2^3 \cdot 500^3 (\log 2)^3 = 8t_1$.

Ако су бројеви са којима се ради мали, нпр. запис им је краћи од 32 бита, онда је за једно дељење потребно константно време, које зависи од снаге процесора.

Проблем 2. Одредити горњу границу за сложеност израчунавања $B^{-1} \pmod M$.

Решење. Пример: израчунавање $11^{-1} \pmod{26}$.

$$\begin{aligned} 26 &= 2 \cdot 11 + 4 \\ 11 &= 2 \cdot 4 + 3 \\ 4 &= 1 \cdot 3 + 1 \end{aligned}$$

$$\begin{aligned} 1 &= 4 - 1 \cdot 3 \\ &= 4 - 1(11 - 2 \cdot 4) = 3 \cdot 4 - 1 \cdot 11 \\ &= 3(26 - 2 \cdot 11) - 1 \cdot 11 = 3 \cdot 26 - 7 \cdot 11 \end{aligned}$$

Према томе, $11^{-1} \equiv -7 + 26 = 19 \pmod{26}$. Алгоритам се састоји од две фазе, прва: израчунавање НЗД, друга: изражавање НЗД у облику линеарне комбинације. Сложеност израчунавања НЗД је $O(\log^3 M)$. Друга фаза има $O(\log M)$ корака (исто као НЗД). Најгори је корак типа $3(26 - 2 \cdot 11) - 1 \cdot 11 = 3 \cdot 26 - 7 \cdot 11$, за копирање 6 бројева је потребно време $6O(\log M) = O(\log M)$. Упрошћавање обухвата једно множење $O(\log^2 M)$ и једно сабирање бројева $\leq M$, сложености $O(\log M)$. Према томе, сложеност првог корака је $O(\log M) + O(\log^2 M) + O(\log M) = O(\log^2 M)$. Укупна сложеност изражавања НЗД у облику линеарне комбинације бројева је $O(\log M)O(\log^2 M) = O(\log^3 M)$. Укупна сложеност инверзије по модулу M је $O(\log^3 M) + O(\log^3 M) = O(\log^3 M)$.

Проблем 3: Претпоставимо да је $B, N \leq M$. Одредити горњу границу за време израчунавања $B^N \pmod M$ вишеструким квадрирањем.

Решење: Нека је n број бита у запису N у систему са основом 2. Број основних корака је $n = O(\log N)$. Пример — израчунавање $87^{43} \pmod{103}$. $43 = (101011)_2 = (n_5 n_4 n_3 n_2 n_1 n_0)_2$.

Корак 0:	Почетак $a = 1$	пошто је $n_0 = 1$,	ставимо $a = 87$.
Корак 1:	$87^2 \equiv 50$	пошто је $n_1 = 1$,	ставимо $a = 87 \cdot 50 \equiv 24 (\equiv 87^2 \cdot 87^1)$
Корак 2:	$50^2 \equiv 28 (\equiv 87^4)$	пошто је $n_2 = 0$,	$a = 24$.
Корак 3:	$28^2 \equiv 63 (\equiv 87^8)$	пошто је $n_3 = 1$,	$a = 24 \cdot 63 \equiv 70 (\equiv 87^8 \cdot 87^2 \cdot 87^1)$
Корак 4:	$63^2 \equiv 55 (\equiv 87^{16})$	пошто је $n_4 = 0$,	$a = 70$.
Корак 5:	$55^2 \equiv 38 (\equiv 87^{32})$	пошто је $n_5 = 1$,	$a = 70 \cdot 38 \equiv 85 (\equiv 87^{32} \cdot 87^8 \cdot 87^2 \cdot 87^1)$

У најгорем случају је увек $n_i = 1$. Размотримо време извршавања једног корака. Нека је S текући остатак $B^{2^i} \pmod{M}$. Приметимо да је $0 \leq a, S < M$. У првом кораку имамо множење $S \cdot S$, $O(\log^2 M)$. Пошто је $0 \leq S^2 < M^2$, сложеност свођења S^2 је $O(\log(M^2) \log M) = O(\log^2(M))$. Нека је $H = S^2 \pmod{M}$. Пошто је $0 \leq H < M$, сложеност множења $H \cdot a$ је $O(\log^2(M))$. Због $0 \leq Ha < M^2$, сложеност свођења $Ha \pmod{M}$ је $O(\log(M^2) \log M) = O(\log^2(M))$. Дакле, слжености једног корака је $O(\log^2(M) + O(\log^2(M) + O(\log^2(M) + O(\log^2(M)) = O(\log^2(M))$. Укупна сложеност израчунавања $B^N \pmod{M}$ поновљеним квадрирањем је $O(\log N) O(\log^2(M)) = O(\log N \log^2(M))$. Ако је $N \leq M$, онда је то просто $O(\log^3(M))$.

Проблем 4. Оценити сложеност израчунавања $N!$ применом алгоритма $(((1 \cdot 2) \cdot 3) \cdot 4) \cdots$. Упутство: $\log(A!) = O(A \log A)$.

Пример: нека је $N = 5$; израчунавање 5!:

$$\begin{aligned} 1 \cdot 2 &= 2 \\ 2 \cdot 3 &= 6 \\ 6 \cdot 4 &= 24 \\ 24 \cdot 5 &= 120 \end{aligned}$$

Укупан број множења је $N - 1$, приближно N. Најгори случај је последњи, $(N-1)! \cdot N$, сложености $O(\log((N-1)!) \log N)$. Пошто је $\log((N-1)!) \approx \log(N!) = O(N \log N)$, претходни израз заокружујемо на $O(N \log N)$. Према томе, сложеност најгорег корака је $O(N \log^2 N)$. Пошто је број корака N, укупно време извршавања је $O(N^2 \log^2 N)$, што је врло споро.

Зашто је $\log(A!) = O(A \log A)$? Према Стирлинговој формули је $A! \approx (A/e)^A \sqrt{2A\pi}$. На пример, $20! = 2.43 \cdot 10^{18}$, а $(20/e)^{20} \sqrt{2 \cdot 20 \cdot \pi} = 2.42 \cdot 10^{18}$. Према томе, $\log(A!) = A(\log A - \log e) + \frac{1}{2}(\log 2 + \log A + \log \pi) = O(A \log A)$ (остали чланови су мањи).

Крај Проблема 4.

Временска сложеност израчунавања B^n је $O(N^i \log^j B)$ за неке $i, j \geq 1$ (вредности i, j одредити за домаћи задатак). Ово је врло споро.

Временска сложеност проналажења најмањег простог чиниоца N узастопним дељењима ($N/2$, $N/3$, $N/4$,...) је $O(\sqrt{N} \log^j N)$ за неко $j \geq 1$ (одредити га за домаћи). Ово је врло споро.

Претпоставимо да имамо r целих бројева као улаз за алгоритам (нпр. r променљивих $N_1, N_2, \ldots N_r$). На пример, за множење је $r = 2$, за факторизацију

$r = 1$, свођење $b^N \pmod M$: $r = 3$. За алгоритам се каже да је полиномијалне сложености ако је време извршавања полином од дужина бројева (односно бројева бита у њима), $O(\log^{d_1} N_1 \log^{d_2} N_2 \ldots \log^{d_r} N_r)$. Тако, алгоритми полиномијалне сложености постоје за налажење НЗД, сабирање, множење, дељење, степеновање поновљеним квадрирањем, одређивање инверза по модулу m.

Ако је $n = O(\log N)$ и $p(n)$ је полином, онда се за алгоритам чије је време извршавања $c^{p(n)}$ $(c > 1)$ каже да има експоненцијалну временску сложеност (у односу на дужину N).

Факторизација поновљеним дељењем је пример таквог алгоритма. Члан $\log^j N$ је толико занемарљив, да се обично каже да је сложеност алгоритма $O(\sqrt{N}) = O(N^{1/2}) = O((c^{\log N})^{1/2}) = O(c^{1/2 \log N}) = O(c^{0.5n})$. Пошто је $0.5n$ полином по n, ово је експоненцијална сложеност. Сложености израчунавања b^N и $N!$ су такође експоненцијалне.

Временска сложеност тренутно најбољих познатих алгоритам за налажење фактора N је $c\sqrt[3]{(\log N(\log\log N)^2)}$, што је много спорије од полиномијалног, али много брже од експоненцијалног. Овај израз је субекспоненцијални, јер је $\sqrt[3]{x}$ за велике x мање од сваке (неконстантне) полиномијалне функције од x. Факторизација 20-цифреног броја поновљеним дељењем трајала би дуже од старости свемира. Године 1999. број од 155 цифара растављен је за 8000 MIPS година. Године 2009. факторисан је број од 232 цифре.

Класа проблема за чије се решавање зна полиномијални алгоритам означава се са P. Постоје фамилије проблема за које се не зна полиномијални алгоритам (иако се за полиномијално време може проверити да ли је дато решење тачно); класа тих проблема означава се са NP. Поткласа ових проблема, на које се могу свести сви проблеми из класе NP је класа NP-комплетних проблема. Ако се пронађе полиномијални алгоритам за неки од NP-комплетних проблема, онда за све њих постоји полиномијални алгоритам. У погледу времена извршавања, зна се да је $\mathrm{P} \leq \mathrm{NP} \leq$ ескпоненцијални.

Важан пример NP-комплетног проблема: одредити решење система нелинеарних полиномијалних једначина по модулу два, као што је нпр. систем $x_1x_2x_5 + x_4x_3 + x_7 \equiv 0 \pmod 2$, $x_1x_9 + x_2 + x_4 \equiv 1 \pmod 2$, ... Ефикасно решење овог проблема би омогућило ефикасно разбијање алгоритма AES.

16 Системи са јавним кључем

У симетричном шифарском систему, ако знамо алгоритам шифровања и кључ за шифровање, онда лако можемо да одредимо алгоритам дешифровања, односно кључ за дешифровање (за полиномијално време). Ова важи на пример за $C \equiv aP + b \pmod{26}$, за проточне шифре, DES и AES .

Шифарски систем са јавним кључем је систем у коме свако зна шифарску трансформацију (алгоритам шифровања), али се не зна полиномијални алгоритам за одређивање кључа за дешифровање полазећи од кључа за шифровање.

Једносмерна функција (one-way function). Нека су X, Y скупови, и нека је $f : X \to Y$ функција. За дато $x \in X$ лако је израчунати $f(x)$ (*лако/брзо* шифровање). За дато $y \in Y$ тешко је одредити x такво да је

$y = f(x)$ (*тешко/споро* декриптирање). Једносмерне функције не морају бити инвертибилне.

Да се у рачунару региструје лозинка (password) *lozinka* фактички се записује $f(lozinka)$, где је f једносмерна функција. Приликом пријављивања ви уносите лозинку. Рачунар израчунава f од укуцане лозинке и то упоређује са записаном вредношћу.

Привидно једносмерна функција (trapdoor one-way function): инвертибилна једносмерна функција f, за коју је одређивање вредности f^{-1} лако, за оне који то знају (*лако/брзо* дешифровање).

Најважније примене криптографије са јавним кључем:

1. Размена кључа за симетрични шифарски систем;
2. Дигитални потпис.

Системи са јавним кључем ретко се користе за шифровање порука, јер су спорији од симетричних система.

16.1 RSA

Систем RSA добио је име по тројици својих проналазача (Rivest, Shamir, Adleman). Подсетимо се да ако је $\text{nzd}(m,n) = 1$ и $a \equiv 1 \pmod{\varphi(n)}$, онда је $m^a \equiv m \pmod{n}$.

Бобан најпре бира два проста броја са око 150 декадних цифара. Затим израчунава $n = pq \approx 10^{300}$ и $\varphi(n) = (p-1)(q-1)$. Затим он одређује неки број e такав да је $\text{nzd}(e, \varphi(n)) = 1$ и израчунава $d \equiv e^{-1} \pmod{\varphi(n)}$. Приметимо да је $ed \equiv 1 \pmod{\varphi(n)}$ и $1 < e, d < \varphi(n)$. Бобан објављује (нпр. на свом сајту) (n, e), а чува у тајности d, p, q. Он може да избрише p и q. Цео овај поступак Бобан обавља једном годишње.

Алиса жели да пошаље Бобану поруку M (то може да буде кључ за AES кодиран бројем $0 \leq M < n$). Ако је порука већа од n, онда она разбија поруку на блокове који се могу представити бројевима мањим од n. Алиса проналази Бобанов пар (n, e) на његовом сајту. Она израчунава $C \equiv M^e \pmod{n}$ (то је привидно једносмерна функција), при чему је $0 \leq C < n$; она шаље број C Бобану.

Бобан израчунава $C^d \pmod{n}$ и добија M. Зашто? $C^d \equiv (M^e)^d \equiv M^{ed} \equiv M^1 = M \pmod{n}$ (ово је тачно и ако није $\text{nzd}(M, n) = 1$, доказати). Пресретнуту поруку C Цица по свему судећи не може да искористи ако не зна Бобанов параметар d.

Пример: Бобан бира $p = 17$, $q = 41$. Он затим израчунава $n = pq = 17 \cdot 41 = 697$ и $\varphi(n) = (17-1)(41-1) = 640$. Он бира $e = 33$, што је узајамно просто са 640. Он затим израчунава $d \equiv 33^{-1} \pmod{640} = 97$. Бобан на свој сајт ставља пар $n = 697$, $e = 33$.

Алиса жели да користи афину шифру $C = aP + b \pmod{26}$ са кључем, $C \equiv 7P + 25 \pmod{26}$ да би Бобану могла да пошаље дугачку поруку. Она кодира кључ бројем $7 \cdot 26 + 25 = 207$, па израчунава шифрат $207^e \pmod{n} = 207^{33} \pmod{697}$. За то она користи свој рачунар и алгоритам

степеновање квадрирањем: $33 = 32 + 1$, $207^2 \equiv 332$, $207^4 \equiv 332^2 \equiv 98$, $207^8 \equiv 98^2 \equiv 543$, $207^{16} \equiv 543^2 \equiv 18$, $207^{32} \equiv 18^2 \equiv 324$. Према томе, $207^{33} \equiv 207^{32}207^1 \equiv 324 \cdot 207 \equiv 156 \pmod{697}$.

Алиса шаље Бобану број 156. Полазећи од броја 156 Цици је теже да израчуна 207.

Бобан добија поруку 156, па израчунава $156^d \pmod{n} = 156^{97} \pmod{697} = 207$. Затим он декодира поруку (то није део алгоритма RSA) $207 = 7 \cdot 26 + 25$. Затим (то такође није део RSA) Алиса шаље Бобану дугачку поруку користећи $C \equiv 7P + 25 \pmod{26}$. Крај примера.

Сваки корисник има свој пар бројева: Алиса има пар n_A, e_A, Бобан пар n_B, e_B, ... на свом сајту, или у "именику" на неком познатом сајту. Пар n_A, e_A је Алисин јавни кључ, а d_A је њен тајни кључ.

Када Алиса шаље поруку M Бобану, она израчунава $M^{e_B} \pmod{n_B}$. Бобан за дешифровање, односно израчунавање M, користи свој кључ d_B.

Зашто је тешко одредити d на основу e и n? Као што је речено, $d \equiv e^{-1} \pmod{\varphi(n)}$. Одређивање инверза је ефикасно (полиномијална сложеност). Одређивање $\varphi(n)$ је тешко ако знамо само n, јер је потребно раставити n на чиниоце; за овај посао знају се субекспоненцијални, али не и полиномијални алгоритми.

Претпоставимо да знамо n. Тада је познавање $\varphi(n)$ полиномијално еквивалентно са познавањем p и q.

Доказ. Ако знамо n, p, q, онда је $\varphi(n) = (p-1)(q-1)$, што се може израчунати за $O(\log^2 n)$.

Претпоставимо сада да знамо n и $\varphi(n)$. Тада је $x^2 + (\varphi(n) - n - 1)x + n = x^2 - (p+q)x + pq = (x-p)(x-q)$. Дакле, бројеви p, q се могу лако одредити решавањем квадратне једначине $x^2 + (\varphi(n) - n - 1)x + n = 0$ на обичан начин. Израчунавање квадратног корена (броја који није обавезно квадрат целог броја) и обављање осталих аритметичких операција захтева време $O(\log^3 n)$. Крај доказа.

Према томе, одређивање $\varphi(n)$ тешко је колико и факторизација.

Препоруке о којима треба водити рачуна приликом избора параметара за RSA, а које ће бити јасније после дела о криптоанализи:

1. p, q обично се бирају тако да $\mathrm{nzd}(p-1, q-1)$ буде мали број.

2. Оба броја p, q треба да имају велики прост чинилац.

3. Бројеви p, q не треба да буду превише близу један другом; с друге стране, однос већег и мањег од њих је обично мањи од 4. Постоје специјални алгоритми за факторизацију, који могу да се искористе ако није испоштована било која од ове три препоруке.

4. Обично је e релативно мало, да би се смањило време шифровања. Често се узима $e = 3$ или $e = 65537 = 2^{16} + 1$.

За личну употребу се обично користе бројеви n од 768 бита, односно $n \approx 10^{231}$. За комерцијалну употребу се обично користе бројеви n од 1024 бита, односно $n \approx 10^{308}$. За важне потребе се обично користе бројеви n

од 2048 бита, односно $n \approx 10^{617}$. Око 1990. године било је уобичајено да се користе 512-битни n, $n \approx 10^{154}$. Међутим, број величине око 10^{193} са конкурса фирме RSA растављен је на чиниоце 2005. године.

Када користе симетрични шифарски систем, Алиса и Бобан морају да унапред договоре заједнички кључ. То ствара потешкоће: потребно је да се они нађу (непрактично) или да кључ пошаљу необезбеђеном линијом (несигурно). На примеру система RSA видимо да је криптографија са јавним кључем потенцијално решење проблема размене кључева за симетричне системе.

16.2 Проблем дискретног логаритма у коначном пољу

Нека је $\mathbf{F}_q$ коначно поље. Нека је g генератор $\mathbf{F}_q^*$, и нека је $b \in \mathbf{F}_q^*$. Тада је $g^i = b$ за неки позитиван цели број $i \leq q-1$. Одређивање i за задате $\mathbf{F}_q$, g и b је ПДЛКП (проблем дискретног логаритма у коначном пољу, Finite Field Discrete Logarithm Problem), — проблем, који је према ономе што се зна, тежак као факторизација.

Пример. Елеменат 2 генерише $\mathbf{F}_{101}^*$. Према томе, знамо да једначина $2^i = 3$ (односно $2^i \equiv 3 \pmod{101}$) има решење. То је $i = 69$. Слично, знамо да једначина $2^i = 5$ има решење; то је $i = 24$. Како се овај проблем може решити ефикасније него грубом силом? Крај примера.

За крипографске примене се обично узима $10^{300} < q < 10^{600}$, где је q велики прост број, или је облика 2^d. Чињеницу да је $g^i = b$ можемо да запишемо у облику $\log_g b = i$. Ово треба да подсећа на обичне логаритме: $\log_{10}(1000) = 3$ јер је $10^3 = 1000$ и $\ln(e^2) = \log_e(e^2) = 2$. У горњем примеру за $q = 101$ је $\log_2(3) = 69$ (јер је $2^{69} \equiv 3 \pmod{101}$). Најбољи алгоритми за решавање ПДЛКП имају сложеност сличну факторизацији, односно субекспоненцијални су.

16.3 Протокол усаглашавања кључа Дифи–Хелман

Протокол усаглашавања кључа над коначним пољем Дифи–Хелман (ПУКДХ) омогућује да Алиса и Бобан размене кључеве без непосредног сусрета. За више корисника A, B, C, ... фиксирају се q и g, генератор $\mathbf{F}_q^*$. Бројеви q и g се користе у целом систему. Сваки корисник има свој приватни кључ a (a_A, a_B, a_C,...), $1 < a < q-1$, и јавни кључ, који је једнак g^a у пољу $\mathbf{F}_q$. Сваки корисник објављује (остатке) g^{a_A}, g^{a_B}, ... у јавном именику или на свом сајту. Често се нови пар a_A, g^{a_A} креира за сваку транскацију (размену порука). У том случају Алиса мора да пошаље Бобану g^{a_A} на почетку поруке. Ако Алиса и Бобан желе да усагласе кључ за AES , они за то користе остатак $g^{a_A a_B}$. Алиса ово може да израчуна степеновањем g^{a_B} на a_A. Бобан може да израчуна исти овај број степеновањем g^{a_A} на a_B.

Цица има q, g, g^{a_A}, g^{a_B}, али по свему судећи не може да израчуна $g^{a_A a_B}$ без решавања ПДЛКП. Ово изгледа изненађује. Она може да израчуна $g^{a_A} g^{a_B} = g^{a_A + a_B}$, али је то бескорисно. Да би израчунала $g^{a_A a_B}$, она мора да степенује нпр. g^{a_A}, на a_B. Да би добила a_B она мора да покуша да

искористи g и g^{a_B}. Али одређивање a_B полазећи од g и g^{a_B} је ПДЛКП, за шта се не зна ефикасан алгоритам.

Пример. $q = p = 97$, $g = 5$. $a_A = 36$ је Алисин приватни кључ. $g^{a_A} = 5^{36} \equiv 50 \pmod{97}$, па је $g^{a_A} = 50$ Алисин јавни кључ. $a_B = 58$ је Бобанов приватни кључ. $g^{a_B} = 5^{58} \equiv 44 \pmod{97}$, па је $g^{a_B} = 44$ Бобанов јавни кључ.

Алиса израчунава $(g^{a_B})^{a_A} = 44^{36} \equiv 75$ (у $\mathbf{F}_{97}$), а Бобан израчунава $(g^{a_A})^{a_B} = 50^{58} = 75$.

Полазећи од 97, 5, 50, 44, Цица не може лако да добије 75.

Практични детаљи: број $q - 1$ треба да има велики прост чинилац (у противном постоји специјални, ефикасан алгоритам за решавање ПДЛКП). Остатак $g^{a_A a_B}$ је отприлике исте величине као и $q \geq 10^{200}$. Да би се од овога добио кључ за AES , они се могу договорити да издвоје најнижих 128 бита бинарне репрезентације броја $g^{a_A a_B}$ ако је q прост број. Ако је $\mathbf{F}_q = \mathbf{F}_2[x]/(f(x))$, онда они могу да се договоре да искористе коефицијенте уз $x^{127}, \ldots, x^0$ у $g^{a_A a_B}$.

Често Алиса и Бобан генеришу a_A и a_B у тренутку контакта, и користе их само за ту размену порука.

17 Мање коришћени шифарски системи са јавним кључем

17.1 RSA као алгоритам за шифровање порука

RSA може да се искористи за шифровање поруке, уместо да се користи само за шифровање кључа за AES (тада нема потребе да се користи AES). Алиса кодира поруку M бројем $0 \leq M < n_B$ и шаље Бобану остатак $M^{e_B} \pmod{n_B}$. Ако је порука предугачка, она је разбија на блокове $M_1, M_2, M_3, \ldots$, $M_i < n_B$.

17.2 ЕлГамалов алгоритам за шифровање

ЕлГамалов алгоритам за шифровање више се користи у варијанти заснованој на употреби елиптичких кривих, него кад се користе коначна поља. Може да се користи за слање порука, или кључа за AES .

Поступак припреме је сличан као код поступка Дифи–Хелман.

Алиса жели да пошаље поруку M Бобану. Ако је величина поља q велики прост број p, онда се порука кодира као број између 0 и $p - 1$. Ако је пак $q = 2^d$, онда се полази од ASCII кодова из M (нпр. $101110\ldots$), па се низ бита кодира полиномом (нпр. $1x^{d-1} + 0x^{d-2} + 1x^{d-3} + \cdots$). Ако је M кључ за AES , онда се тај кључ може кодирати полиномом. Ако је M превелико, онда се разбија на блокове.

Алиса бира случајни број k, $1 < k < q$. Она бира различите k за сваку нову поруку. Она затим шаље Бобану пар (остатака у коначном пољу) $g^k, Mg^{a_B k}$.

Алиса зна g и g^{a_B} (то су јавни подаци) и k, па може да израчуна g^k и $(g^{a_B})^k = g^{a_B k}$, после чега множењем добија $Mg^{a_B k}$. Бобан прима пар. Он не може да одреди k, али му то није ни потребно. Он најпре израчунава $(g^k)^{a_B} = g^{a_B k}$ (он зна a_B, свој приватни кључ). Затим он израчунава $(g^{a_B k})^{-1}$ (у пољу $\mathbf{F}_q$), а онда множењем добија $(Mg^{a_B k})(g^{a_B k})^{-1} = M$.

Ако Цица пронађе k (за шта јој по свему судећи треба решавање ПДЛКП, јер се зна g, а послато је g^k) она може да израчуна $(g^{a_B})^k = g^{a_B k}$, а затим $(g^{a_B k})^{-1}$, па M.

Пример: $q = 97$, $g = 5$, $a_B = 58$ је Бобанов приватни кључ, $g^{a_B} = 44$ је Бобанов јавни кључ. Алиса жели да пошаље $M = 30$ Бобану. Она бира случајни кључ сесије $k = 17$, па израчунава $g^k = 5^{17} = 83$. Она зна $g^{a_B} = 44$ (то је јавни податак) и израчунава $(g^{a_B})^k = 44^{17} = 65$. Затим она израчунава $Mg^{a_B k} = 30 \cdot 65 = 10$. Она шаље Бобану g^k, $Mg^{a_B k} = 83, 10$. Бобан прима 83, 10. Он зна $a_B = 58$, па израчунава $(g^k)^{a_B} = 83^{58} = 65 = g^{a_B k}$. Затим он израчунава $(g^{a_B k})^{-1} = 65^{-1} = 3$ (тј. $65^{-1} \equiv 3 \pmod{97}$), па множењем добија $(Mg^{a_B k}(g^{a_B k})^{-1} = 10 \cdot 3 = 30 = M$.

17.3 Размена кључева Меси-Омура

Размена кључева Меси-Омура (Massey-Omura) није ни симетрични ни асиметрични шифарски систем. Може се искористити за слање кључа или поруке.

За скуп корисника фиксира се коначно поље $\mathbf{F}_q$, где је q велики број. Није потребан ни генератор, ни јавни кључеви. Пре него што Алиса пошаље Бобану поруку, Алиса бира случајни кључ за шифровање e_A, $\mathrm{nzd}(e_A, q-1) = 1$, а Бобан бира случајни кључ за шифровање e_B, $\mathrm{nzd}(e_B, q-1) = 1$. Оба ова кључа користиће се само за једну размену порука.

Алиса израчунава $d_A \equiv e_A^{-1} \pmod{q-1}$. Бобан израчунава $d_B \equiv e_B^{-1} \pmod{q-1}$. Ништа од ових бројева се не објављује.

Алиса кодира поруку елементом поља $M \in \mathbf{F}_q^*$. Ако је порука предугачка, она је разбија на блокове. Она шаље остатак M^{e_A} из поља $\mathbf{F}_q$ Бобану. Бобану је то неразумљиво, па он шаље назад Алиси $(M^{e_A})^{e_B} = M^{e_A e_B}$, што је пак њој неразумљиво. Алиса шаље Бобану $(M^{e_A e_B})^{d_A} = M^{e_A e_B d_A} \stackrel{*}{=} M^{e_A d_A e_B} = M^{e_B}$. На крају Бобан израчунава $(M^{e_B})^{d_B} = M$.

Кључни је корак означен звездицом. У суштини, Алиса навлачи чарапу на стопало. Бобан преко чарапе навлачи ципелу. Алиса затим уклања чарапу, не скидајући ципелу, а Бобан уклања и ципелу. Бобан сада види стопало, иако га Цица ни једном није видела.

Шифарски систем Меси–Омура је примењиван на мобилне телефоне.

Пример. $p = 677$. Алиса шаље Бобану биграм SC. Пошто је $S = 18$ и $C = 2$, биграм се кодира се $18 \cdot 26 + 2 = 470 = M$. Алиса бира $e_A = 255$, па је $d_A = 255^{-1} \pmod{676} = 395$. Бобан бира $e_B = 421$, па је $d_B = 421^{-1} \pmod{676} = 281$. Алиса израчунава $470^{255} \equiv 292 \pmod{677}$ и шаље 292 Бобану. Бобан израчунава $292^{421} \equiv 156 \pmod{677}$ и шаље 156 Алиси. Алиса израчунава $156^{395} \equiv 313 \pmod{677}$ и шаље 313 Бобану. Бобан израчунава $313^{281} \equiv 470 \pmod{677}$ и декодира 470 као биграм SC.

18 Елиптичке криве

Елиптичка крива је крива описана једначином облика $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$, са једном допунском 0-тачком (бесконачно удаљена тачка). Пример $y^2 + y = x^3 - x$ приказан је на Слици 1. Испоставља се да се све кубне криве могу довести на овај облик сменом промењивих; овај облик је погодан за дефинисање операције сабирања тачака, коју ћемо описати. За сада рачунамо над скупом реалних бројева. Потребна нам је нула-тачка коју ћемо означавати са $\oslash$. Замислимо да смо савили све вертикалне праве, тако да су им спојени горњи и доњи "крајеви", и да смо их онда залепили. Тачка коју смо добили зове се бесконачно удаљена тачка или 0-тачка. Она затвара нашу криву линију. Она је горњи и доњи завршетак сваке вертикалне праве.

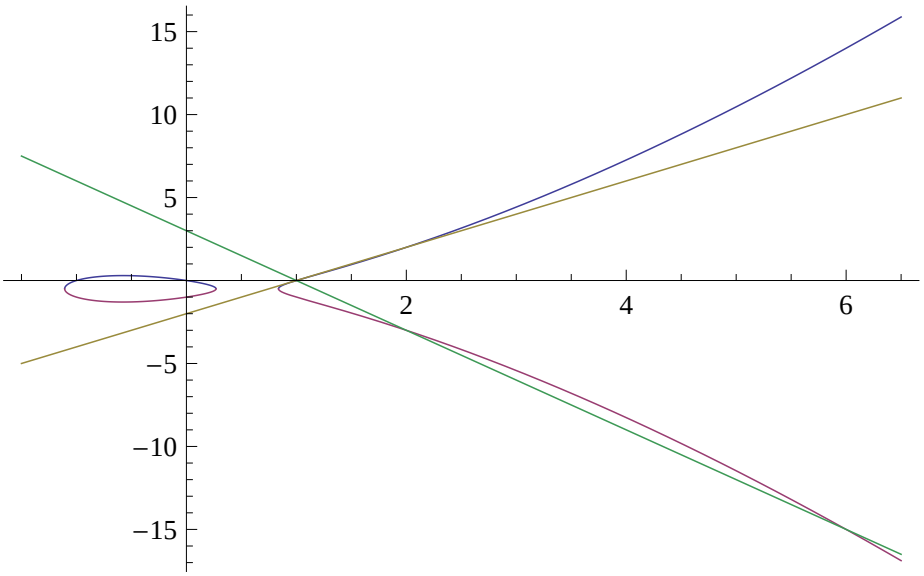
Сада можемо да дефинишемо операцију сабирања на скупу тачака елиптичке криве. Најпре, $\oslash + \oslash = \oslash$ и $\oslash + P = P + \oslash = P$ за произвољну тачку криве ($\oslash$ је неутрални елемент за сабирање тачака). Свака вертикална права, ако сече криву у једној тачки P, онда је сече у тачно још једној тачки Q (ако је један корен квадратне једначине са реалним коефицијентима реалан, онда је и други). По дефиницији је $P + Q = \oslash$ и $-P = Q$ (супротни елемент тачке). Ако нека права која није вертикална сече криву у две тачке P и Q, онда права сече криву у још једној тачки R (ако кубна једначина има два реална корена, онда је и трећи корен реалан). Збир тачака P и Q је по дефиницији $P + Q = -R$.

Наредне слике илуструју сабирање тачака криве. Вертикална права L_1 сече криву у тачкама P_1, P_2 и $\oslash$, па је $P_1 + P_2 + \oslash = \oslash$, тј. $P_1 = -P_2$, и $P_2 = -P_1$. Две различите тачке криве са истом x-координатом су инверзне/супротне једна другој; видети Слику 2.

Ако желимо да извршимо сабирање $P_1 + P_2$, тачака са различитим x-координатама, спајамо их правом и проналазимо трећу пресечну тачку P_3. Приметимо да је $P_1 + P_2 + P_3 = \oslash$, $P_1 + P_2 = -P_3$. Видети Слику 3.

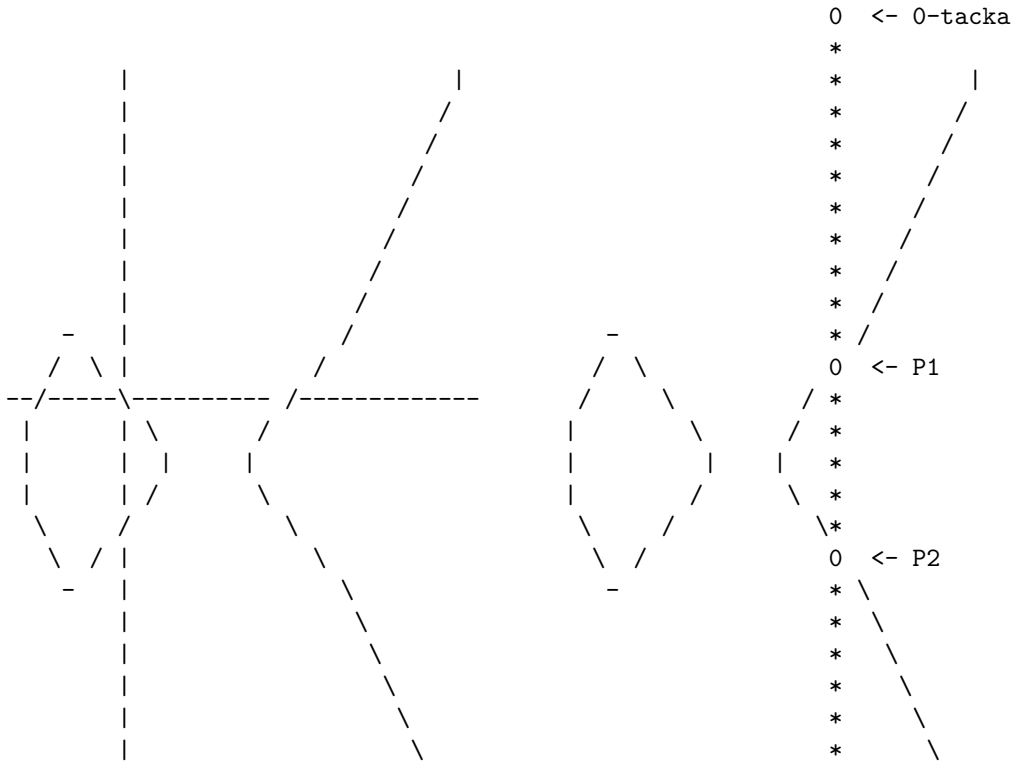
Дигресија: Где се секу $y = x^2$ и $y = 2x - 1$? Тамо где је $x^2 = 2x - 1$, тј. $x^2 - 2x + 1 = (x - 1)^2 = 0$. Оне се секу у тачки $x = 1$ два пута (због експонента), што је у вези са чињеницом да је $y = 2x - 1$ тангента на $y = x^2$, видети Слику 4.

Назад на елиптичке криве. Како се може удвостручити тачка P_1? Треба конструисати тангенту на криву и наћи другу пресечну тачку P_2. $P_1 + P_1 + P_2 = \oslash$, па је $2P_1 = -P2$; видети Слику 5.

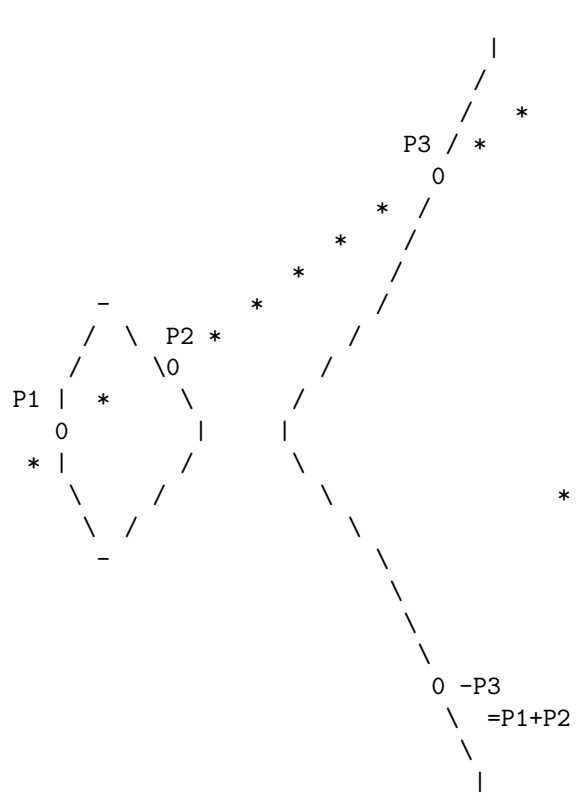


1. Елиптичка крива са x и y-осама $y^2 + y = x^3 - x$

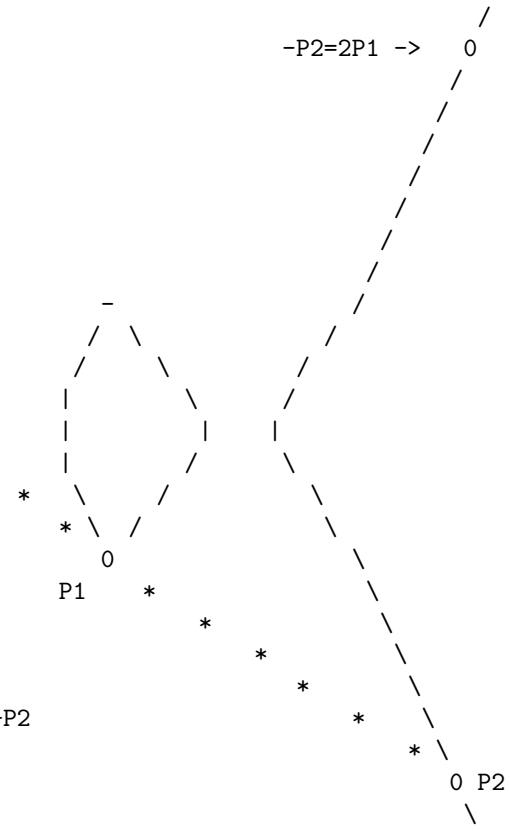
2. Елиптичка крива без оса. Одређивање супротне тачке.



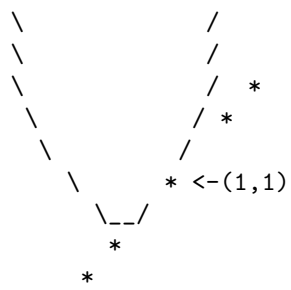
3. Сабирање $P_1 + P_2$.



5. Удвостручавање P_1.



4. $y = x^2$ i $y = 2x - 1$



Пример. Тачка $P = (1, 0)$ очигледно припада кривој $y^2 + y = x^3 - x$. Одредимо $2P$. Одређујемо тангенту у P имплицитним диференцирањем. $2y\frac{dy}{dx} + \frac{dy}{dx} = 3x^2 - 1$. Дакле, $\frac{dy}{dx} = \frac{3x^2-1}{2y+1}$ и $\frac{dy}{dx}|_{(1,0)} = 2$. Једначина тангенте је $y - 0 = 2(x - 1)$, тј. $y = 2x - 2$. Где тангента сече $y^2 + y = x^3 - x$? Тамо где је $(2x - 2)^2 + (2x - 2) = x^3 - x$, односно $x^3 - 4x^2 + 5x - 2 = 0 = (x - 1)^2(x - 2)$.

Тачка пресека $x = 1$ је двострука (пресек у $(1,0)$), а пресек у $x = 2$ је обичан. Приметимо да је трећа тачка пресека на правој $y = 2x - 2$, па је то тачка $(2,2)$. Дакле, $(1,0) + (1,0) + (2,2) = 2P + (2,2) = \oslash$, $(2,2) = -2P$, $2P = -(2,2)$. Тачка $-(2,2)$ је друга тачка на кривој са истом x-координатом. Ако је $x = 2$, онда је $y^2 + y = 6$, па је $y = 2, -3$, па је $2P = (2,-3)$.

Да бисмо одредили $3P = P + 2P = (1,0) + (2,-3)$, одређујемо једначину праве кроз $(1,0)$, $(2,-3)$. Њен нагиб је -3, па је $y - 0 = -3(x-1)$, односно $y = -3x + 3$. Где ова права сече $y^2 + y = x^3 - x$? Елиминацијом y добија се $(-3x+3)^2 + (-3x+3) = x^3 - x$, тј. $x^3 - 9x^2 + 20x - 12 = 0 = (x-1)(x-2)(x-6)$ (приметимо да ми знамо да права сече криву за $x = 1$ и $x = 2$, па је $(x-1)(x-2)$ чинилац полинома $x^3 - 9x^2 + 20x - 12$, па је одређивање трећег корена лако). Трећа пресечна тачка има x-координату $x = 6$ и лежи на правој $y = -3x + 3$, па је то тачка $(6,-15)$. Дакле, $(1,0) + (2,-3) + (6,-15) = \oslash$ и $(1,0) + (2,-3) = -(6,-15)$. Шта је $-(6,-15)$? Ако је $x = 6$, онда је $y^2 + y = 210$, $y = -15, 14$, па је $-(6,-15) = (6,14) = P + 2P = 3P$. Крај примера.

Пошто је сабирање тачака низ алгебарских операција, могу се извести изрази за збир. Нека су задате тачке $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$ и $P_1 \neq -P_2$. Да би се израчунала тачка $P_3 = (x_3, y_3) = P_1 + P_2$, најпре се израчунавају

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1}, \quad \nu = \frac{y_1 x_2 - y_2 x_1}{x_2 - x_1}$$

ако је $x_1 \neq x_2$, односно

$$\lambda = \frac{3x_1^2 + 2a_2 x_1 + a_4 - a_1 y_1}{2y_1 + a_1 x_1 + a_3}, \quad \nu = \frac{-x_1^3 + a_4 x_1 + 2a_6 - a_3 y_1}{2y_1 + a_1 x + a_3}$$

ако је $x_1 = x_2$, а затим, у оба случаја, $x_3 = \lambda^2 + a_1\lambda - a_2 - x_1 - x_2$ и $y_3 = -(\lambda + a_1)x_3 - \nu - a_3$.

Скуп тачака криве је група за овако дефинисано сабирање. Асоцијативност сабирања може се проверити непосредно (што није тривијално) применом горњих израза за збир.

Пример. Одредити $(1,0) + (2,-3)$ на кривој $y^2 + y = x^3 - x$ користећи ове изразе. $a_1 = 0$, $a_3 = 1$, $a_2 = 0$, $a_4 = -1$, $a_6 = 0$, $x_1 = 1$, $y_1 = 0$, $x_2 = 2$, $y_2 = -3$. $\lambda = \frac{-3-0}{2-1} = -3$, $\nu = \frac{0 \cdot 2 - (-3)(1)}{2-1} = 3$. Дакле, $x_3 = (-3)^2 + 0(-3) - 0 - 1 - 2 = 6$ и $y_3 = -(-3+0)(6) - 3 - 1 = 14$. Закључујемо да је $(1,0) + (2,-3) = (6,14)$.

18.1 Проблем дискретног логаритма са елиптичким кривама

Размотићемо сада проблем дискретног логаритма са елиптичким кривама (ECDLP, Elliptic curve discrete logarithm problem).

Уместо са реалним бројевима, сада рачунамо у коначном пољу. У пољу $\mathbf{F}_p$, при чему је $p \neq 2$ прост број, пола елемената су квадрати. На пример, у $\mathbf{F}_{13}$ је $1^2 = 1$, $2^2 = 4$, $3^2 = 9$, $4^2 = 3$, $5^2 = 12$, $6^2 = 10$, $7^2 = 10$, $8^2 = 12$, $9^2 = 3$, $10^2 = 9$, $11^2 = 4$, $12^2 = 1$. Једначина $y^2 = 12$ има два решења $y = \pm 5 = 5, 8$. Ако је g генератор, онда су g^{2k} квадрати, а g^{2k-1} нису.

Постоје ефикасни алгоритми за утврђивање да ли је неки елеменат поља $\mathbf{F}_p$ квадрат, односно ако јесте — за рачунање корена из њега. За $p > 3$ једначина елиптичке криве може се сменом променљивих трансформисати у облик $y^2 = x^3 + a_4x + a_6$.

Пример. Нека је E крива $y^2 = x^3+1$; одредити $E(\mathbf{F}_5)$ (тачке са координатама у $\mathbf{F}_5$). Корисно је претходно израчунати квадрате: $0^2 = 0$, $1^2 = 1$, $2^2 = 4$, $3^2 = 4$, $4^2 = 1$.

x	x^3+1	$y = \pm\sqrt{x^3+1}$	*tacke*
0	1	$\pm 1 = 1, 4$	$(0,1), (0,4)$
1	2		
2	4	$\pm 2 = 2, 3$	$(2,2), (2,3)$
3	3		
4	0	0	$(4,0)$
			$\oslash$

Скуп $E(\mathbf{F}_5)$ има дакле 6 тачака.

Тачке над коначним пољем се могу сабирати коришћењем једначина правих или применом израза за сабирање. Ако је $G = (2,3)$, онда је $2G = (0,1)$, $3G = (4,0)$, $4G = (0,4)$ (приметимо да ова тачка има исту x-координату као и $2G$, па је $4G = -2G$ и $6G = \oslash$), $5G = (2,2)$, $6G = \oslash$. Видимо да је $G = (2,3)$ генератор групе $E(\mathbf{F}_5)$.

Пример. Нека је E крива $y^2 = x^3+x+1$ над $\mathbf{F}_{109}$. Испоставља се да скуп $E(\mathbf{F}_{109})$ има 123 тачака и да је генерисан тачком $G = (0,1)$. Тачка $(39,45)$ је у $E(\mathbf{F}_{109})$ јер је $39^3 + 39 + 1 \equiv 63 \pmod{109}$ и $45^2 \equiv 63 \pmod{109}$. Дакле, $(39,45) = (0,1) + (0,1) + ... + (0,1) = n(0,1)$ за неки природни број n.

Колико је то n? Одређивање n је проблем дискретног логаритма са елиптичким кривама над коначним пољем. Проблем се може решавати грубом силом, али не ако се 109 замени простим бројем $\approx 10^{50}$. Овај проблем је тренутно теже решити него ПДЛКП, па се могу користити краћи кључеви. Друга предност је у томе што се за фиксирано коначно поље може посматрати више елиптичких кривих.

Потребна је једна или две тачке да се изгенерише $E(\mathbf{F}_p)$. Посматрајмо криву $y^2 = x^3 + 1$ над $\mathbf{F}_7$. $0^2 = 0$, $(\pm 1)^2 = 1$, $(\pm 2)^2 = 4$, $(\pm 3)^2 = 2$.

x	x^3+1	$y = \pm\sqrt{x^3+1}$	тачке
0	1	± 1	$(0,1), (0,6)$
1	2	± 3	$(1,3), (1,4)$
2	2	± 3	$(2,3), (2,4)$
3	0	0	$(3,0)$
4	2	± 3	$(4,3), (4,4)$
5	0	0	$(5,0)$
6	0	0	$(6,0)$
			$\oslash$

Према томе, $E(\mathbf{F}_7)$ има 12 тачака.

$$\begin{array}{cc} & R=(5,0) \qquad 2R=\oslash \\ Q=(1,3) & Q+R=(2,3) \\ 2Q=(0,1) & 2Q+R=(4,4) \\ 3Q=(3,0) & 3Q+R=(6,0) \\ 4Q=(0,6) & 4Q+R=(4,3) \\ 5Q=(1,4) & 5Q+R=(2,4) \\ 6Q=\oslash & \end{array}$$

Све тачке су облика $nQ+mR$, при чему $n \in \mathbf{Z}_6$ и $m \in \mathbf{Z}_2$. Приметимо да су коефицијенти криве $y^2 = x^3+1$ и координате тачака дефинисане по модулу 7, а да се тачке сабирају по модулу 6. У овом случају **две** тачке генеришу криву. И даље се може радити са дискретним логаритмом користећи нпр. тачку $G = (1,3)$ као псеудогенератор. Ова тачка генерише само половину скупа $E(\mathbf{F}_7)$.

У просеку скуп $E(\mathbf{F}_p)$ има $p+1$ тачку.

18.2 Системи са елиптичким кривама

18.2.1 Систем аналоган ПУКДХ

Најпре се бира прост број $p \approx 10^{50}$ и ради се над пољем $\mathbf{F}_p$. Пошто је ова варијанта проблема дискретног логаритма тежа од раније описане у $\mathbf{F}_p^*$, може се изабрати мање p. Фиксира се нека елиптичка крива $E(y^2 = x^3 + a_4x + a_6)$ и (псеудо) генератор — тачка $G = (x_1, y_1) \in E(\mathbf{F}_p)$, тако да је $y_1^2 = x_1^3 + a_4x_1 + a_6) \pmod p$, при чему је $nG = \oslash$ за неки велики број n. Подсетимо се да је $nG = G + G + \cdots + G$ (n сабирака). Број n треба да има бар један врло велики прост фактор и не сме да важи $n = p$, $n = p-1$, или $n = p+1$ за било који прост број p (у противном постоје специјални алгоритми за решавање ECDLP).

Сваки корисник има приватни број a_A, a_B, ... и јавни кључ — тачку $a_AG, a_BG, \ldots$. За своју комуникацију Алиса и Бобан користиће кључ a_Aa_BG.

Пример. $p = 211$, $E : y^2 = x^3 - 4$, $G = (2,2)$. Испоставља се да је $241G = \oslash$. Алисин приватни кључ је $a_A = 121$, па је њен јавни кључ $a_AG = 121(2,2) = (115,48)$. Бобанов приватни кључ је $a_B = 223$, па је његов јавни кључ $a_BG = 223(2,2) = (198,72)$. Њихов заједнички (договорени) кључ је a_Aa_BG. Према томе, А израчунава $a_A(a_BG) = 121(198,72) = (111,66)$, а Б израчунава $a_B(a_AG) = 223(115,48) = (111,66)$. Према томе, $(111,66)$ је њихов заједнички кључ, који могу да користе за неки симетрични шифарски систем.

Приметимо да је утврђивање који умножак тачке $G = (2,2)$ даје Алисин јавни кључ $(115,48)$ уствари инстанца проблема ЕЦДЛП. Алиса може да множење $121G$ изврши поновљеним удвостручавањем: $64G + 32G + 16G + 8G + 1G$. Поред тога, ако је $p \approx 10^{50}$, Алиса и Бобан могу да се договоре да последњих 128 бита бинарне представе x-координате њиховог заједничког кључа искористе као кључ за AES .

Практичне препоруке: За ПУКДХ се обично користе вредности $q > 10^{200}$, док се за ЕЦДХ обично узима $q > 10^{50}$. Разлог за ово је чињеница да је сабирање две тачке на елиптичкој кривој много спорија операција од множења у коначном пољу. Међутим, пошто је q много мање за ЕЦДХ, то надокнађује спорост основне операције. Поред тога, у погледу решавања ЕЦДЛП није било напретка у последњих 20 година, док се решавање ФФДЛП стално усавршава.

18.2.2 Систем аналоган ЕлГамаловој размени порука

Прва потешкоћа: како кодирати поруку као тачку? Вратимо се најпре на случај коначних поља. Ако радимо са $p = 29$, онда се свако слово може кодирати елементом $\mathbf{F}_{29}$, $A = 0, \ldots, Z = 25$. Шта радити са елиптичком кривом, нпр. $y^2 = x^3 - 4$ над $\mathbf{F}_{29}$? Најједноставније би било кодирати број x-координатом неке тачке, али нису сви бројеви x-координате неких тачака (само отприлике свака друга од њих). Исто тако, нису ни сви бројеви y-координате неких тачака (само отприлике свака друга од њих). Ако нпр. покушамо да кодирамо слово $I = 8$: $8^3 - 4 = 15 \neq k^2 \pmod{29}$ (15 није квадратни остатак по модулу 29).

Уместо тога, може се радити са $p = 257$ (изабрано због тога што је ово први прост број већи од $25 \cdot 10$). Број који се добија дописивањем једне цифре редном броју слова служи као код тог слова. Пошто се може изабрати једна од 10 цифара, а свака од њих даје потенцијалну x-координату тачке са вероватноћом 50%, ово решење може да функционише (у пракси се може поруци додати 8 бита, тако да је вероватноћа проблема практично једнака нули).

Нека је $p = 257$, $E : y^2 = x^3 - 4$. Порука је $L = 11$. Треба одредити тачку $(\overline{11a}, y)$ на кривој. Покушавамо најпре са $x = 110$.

$x = 110$, $110^3 - 4 \equiv 250 \neq k^2 \pmod{257}$.

$x = 111$, $111^3 - 4 \equiv 130 \neq k^2 \pmod{257}$.

$x = 112$, $112^3 - 4 \equiv 162 \equiv 26^2 \pmod{257}$.

Према томе, $(112, 26)$ је тачка на кривој, и све цифре x-координате, сем последње, су порука. Ако Алиса жели да пошаље поруку L Бобану, она најпре бира случајно k. Нека је $a_B G$ Бобанов јавни кључ. Нека је Q тачка која кодира отворени текст. Тада Алиса шаље $(kG, Q + ka_B G)$ Бобану. Бобан прима поруку, израчунава $a_B kG$; одузимајући то од $Q + ka_B G$, он добија тачку Q, отворени текст.

Пример. $p = 257$, $E : y^2 = x^3 - 4$, $G = (2, 2)$, $a_B = 101$, $a_B G = 101(2, 2) = (197, 167)$ (то је тачка која је Бобанов јавни кључ). Алиса жели да пошаље поруку L кодирану тачком $Q = (112, 26)$ Бобану. Она бира случајни *кључ поруке* (session key) $k = 41$ и израчунава $kG = 41(2, 2) = (136, 128)$, $k(a_B G) = 41(197, 167) = (68, 84)$ и $Q + ka_B G = (112, 26) + (68, 84) = (246, 174)$. Алиса шаље пар тачака $kG, Q + ka_B G$, односно $(136, 128)$, $(246, 174)$ Бобану. Бобан прима пар и израчунава $a_B(kG) = 101(136, 128) = (68, 84)$, па $(Q + ka_B G) - (a_B kG) = (246, 174) - (68, 84) = (246, 174) + (68, -84) = (112, 26)$. Он затим узима све сем последње цифре x-координате и добија $11 = L$.

И у овом контексту људи радије користе поља типа $\mathbf{F}_{2^r}$. Постоје субекспоненцијални алгоритми за решавање ФФДЛП у $\mathbf{F}_q^*$ и за факторизацију n, па се обично користи $q \approx n > 2^{200}$. Најбољи познати алгоритам за решавање ЕЦДЛП у $E(\mathbf{F}_q)$ има сложеност $O(\sqrt{q})$, што је експоненцијално. Због тога се обично ради са вредностима $q > 10^{50}$. Потребно је више времена за сабирање тачака елиптичке криве него за множење у коначном пољу (или по модулу n) за исту величину q. Међутим, пошто је q много мање, пад ефикасности није значајан. Поред тога, када се ради са елиптичким кривама, кључеви су краћи, што олакшава размену кључева, односно рад у ситуацији кад је на располагању мала меморија, односно кад израчунавања треба да буду једноставна, као кад се користе паметне картице.

19 Хеш функције, MD5, кодови за аутентикацију (MAC)

Како се може обезбедити да поруку коју сте примили није неко успут променио? Особина поруке да није доживела измене је *интегритет*. Решење је примена хеш алгоритма $H(x)$. Улаз за хеш алгоритам је променљиве дужине, а излаз је увек исте дужине. Хеш функција треба да има следеће особине

1. Потребно је да се њене вредности лако (брзо) израчунавају.
2. Хеш функција треба да буде једносмерна функција (за задато y потребно је да буде тешко одређивање таквог x да је $H(x) = y$).
3. За задато x потребно је да буде тешко одређивање другог x' таквог да је $H(x') = H(x)$. Ова особина зове се *основна отпорност на колизију*.

Ако је поред тога испуњен захтев да је тешко пронаћи било који пар x, x' такав да је $H(x) = H(x')$, онда се за H каже да има *јаку отпорност на колизију*. Може се показати (под разумним претпоставкама) да ако је функција јако отпорна на колизију, онда је и слабо отпорна на колизију. Због тога се обично од хеш функција захтева јака отпорност на колизију.

Да се направи хеш алгоритам, обично се полази од функције f која блокове од $m+t$ бита пресликава у блокове од t бита, где су и m и t велики, а f има све три наведене особине. Улаз за хеш функцију зове се порука. Излаз хеш алгоритма је хеш или хеш вредност.

Функција f може се искористити за добијање хеш функције. Претпоставимо да је порука разбијена у m-битне блокове M_1, M_2, $\ldots M_k$. Ако дужина поруке није дељива са m, онда се последњи блок допуњује до дужине m. Задат је унапред договорени блок од t бита (иницијализациони вектор IV). Тада је $H(M) = H_k$, где је $H_0 = IV$ и $H_i = f(H_{i-1}, M_i)$, $i = 1, 2, \ldots, k$.

Пример 1. $IV = 111\ldots1111$ (блок од 128 јединица). Порука се разбија у блокове од по $m = 128$ бита, а за f се узима $f(u, v) = AES_u(v)$ (ово је тзв. AES CBC-MAC).

Пример 2. Слично као у претходном примеру, изузев што се уместо фиксираног IV користи тајни кључ. Тада се хеш функција зове *аутентикациони код поруке* или MAC (message authentication code). MAC је дакле хеш функција са тајним кључем.

Пример 3. Размотримо следећи сценарио. Алиса и Бобан користе систем са јавним кључем да договоре два кључа за AES , k_1 и k_2. Алиса шаље Бобану (у режиму ECB , због једноставности) поруку шифровану алгоритмом AES . Она разбија поруку на n блокова: PT_1, ...,PT_n. Сваки од ових блокова она шифрује алгоритмом AES са кључем k_1, и добија шифрате ST_1, ...,ST_n. Затим Алиса израчунава MAC низа $PT_1PT_2 \dots PT_n$ користећи кључ k_2, као у Примеру 2, и шаље (нешифровани) MAC Бобану.

Бобан прима $ST_1, \dots, ST_n$ и дешифрује их кључем k_1. Сада Бобан, пошто има низ блокова PT_i, помоћу кључа k_2 одређује MAC . Тако он може да провери да ли се тај MAC слаже са оним добијеним из поруке. Ако постоји слагање, онда он може да буде сигуран да поруку нико успут није променио.

Без MAC Цица би могла да пресретне ST_1, ...,ST_n и да неке делове шифрата намерно промени (иако то дешифровањем не би дало смислену поруку, јер Цица не зна кључ).

Ако Цица промени поруку, она не може да креира MAC који би се слагао са MAC поруке. Крај примера.

Пример 4. Алиса и Бобан договорили су заједнички кључ K, а H је хеш функција. Нека KM означава конкатенацију K и M. Ако Алиса жели за израчуна MAC поруке M, она израчунава $H(KH(KM))$ (дакле, K се конкатенира са хеш вредношћу поруке KM).

19.1 Хеш алгоритам MD5

Један од најпопуларнијих хеш алгоритама је MD5. Он је ефикаснији од мало пре описаног алгоритма који користи AES . Он се заснива на следећој функцији f. Функција f пресликава блок од 512 бита у блок од 128 бита. Нека је M блок од 512 бита. У овом контексту ћемо рећи да је блок од 32 бита реч. Према томе, M се састоји од 16 речи. Нека је $X[0]$ прва реч, $X[1]$ следећа, ..., $X[15]$ — последња.

19.1.1 Почетно стање регистара

У току израчунавања стално се ажурира стање четири 32-битна регистра A, B, C, D (стање сваког од њих је нека реч). На почетку је $A = A_0 = 0x01234567$, $B = B_0 = 0x89abcdef$, $C = C_0 = 0xfedcba89$, $D = D_0 = 0x76543210$. Ознака $0x$ означава да иза ње следи хексадецимални број. На тај начин ниблови $0000, 0001, \dots, 1111$ представљају се цифрама $0, 1, \dots, f$.

19.1.2 Четири функције

Дефинисаћемо четири функције, од којих свака има улаз од три речи X, Y, Z.

$F(X, Y, Z) = XY \vee \bar{X}Z$,

$G(X,Y,Z) = XZ \vee Y\bar{Z}$,
$H(X,Y,Z) = X \oplus Y \oplus Z$,
$I(X,Y,Z) = Y \oplus (X \vee \bar{Z})$.

Овде је $\bar{1} = 0$, $\bar{0} = 1$ (то је негација); $\vee$ је дисјункција (или), а множење је конјункција (и). Операције над речима изводе се истовремено са свим њиховим одговарајућим битима. На пример, применимо F на три бајта (уместо на три речи). Нека је $X = 00001111$, $Y = 00110011$, $Z = 01010101$.

X	Y	Z	XY	$\bar{X}Z$	$XY \vee \bar{X}Z$
0	0	1	0	1	1
0	1	0	0	0	0
0	1	1	0	1	1
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	1	0	1
1	1	1	1	0	1

Према томе, $F(X,Y,Z) = 01010011$. Приметимо да за 4 од 8 могућих једнобитних улаза у F има вредности 0, а за остала 4 улаза има вредност 1. Ово важи такође и за G, H, I.

19.1.3 Константе

Постоје 64 константе $T[1], \ldots, T[64]$. Нека је i број радијана. Тада је $|\sin(i)|$ реални број између 0 и 1. Нека је $T[i]$ првих 32 бита после децималне тачке у бинарној репрезентацији $|sin(i)|$.

19.1.4 Ознаке за ротацију

Ако је E 32-битни блок и $1 \leq n \leq 31$, нека $E <<< n$ означава цикличку ротацију E за n бита. Тако, ако је $E = 00000000000000000111111111111111$, онда је $E <<< 3 = 00000000000001111111111111111000$.

19.1.5 Остале ознаке

Нека $+$ означава сабирање по модулу 2^{32}, при чему се претпоставља да је реч бинарно представљени број n, $0 \leq n \leq 2^{32} - 1$.

Ознака $x \leftarrow y$ означава доделу тренутне вредности променљиве y променљивој x. Тако, ако је у неком тренутку $x = 4$, онда после $x \leftarrow x + 1$ променљива x има нову вредност 5.

19.1.6 Израчунавање f

Улаз за f је M и четири речи, које означавамо са A, B, C, D. Најпре се четири речи копирају: $AA \leftarrow A$, $BB \leftarrow B$, $CC \leftarrow C$, $DD \leftarrow D$.

Затим се извршава 64 корака у четири рунде.

Рунда 1 састоји се од 16 корака.

Нека `[abcd k s i]` означава операцију $a \leftarrow b + ((a + F(b,c,d) + X[k] + T[i]) <<< s)$. Ових 16 корака су

```
[ABCD   0   7    1] [DABC   1   12  2] [CDAB     2  17   3] [BCDA    3  22   4]
[ABCD   4   7    5] [DABC   5   12  6] [CDAB     6  17   7] [BCDA    7  22   8]
[ABCD   8   7    9] [DABC   9   12 10] [CDAB    10  17  11] [BCDA   11  22  12]
[ABCD   12  7 13] [DABC 13  12 14] [CDAB    14  17  15] [BCDA   15  22  16].
```

Рунда 2 састоји се од наредних 16 корака.

Нека `[abcd k s i]` означава операцију $a \leftarrow b + ((a + G(b,c,d) + X[k] + T[i]) <<< s)$. Ових 16 корака су

```
[ABCD 1 5 17] [DABC 6 9 18] [CDAB 11 14 19] [BCDA 0 20 20]
[ABCD 5 5 21] [DABC 10 9 22] [CDAB 15 14 23] [BCDA 4 20 24]
[ABCD 9 5 25] [DABC 14 9 26] [CDAB 3 14 27] [BCDA 8 20 28]
[ABCD 13 5 29] [DABC 2 9 30] [CDAB 7 14 31] [BCDA 12 20 32]
```

Рунда 3 састоји се од наредних 16 корака.

Нека `[abcd k s i]` означава операцију $a \leftarrow b + ((a + H(b,c,d) + X[k] + T[i]) <<< s)$. Ових 16 корака су

```
[ABCD 5 4 33] [DABC 8 11 34] [CDAB 11 16 35] [BCDA 14 23 36]
[ABCD 1 4 37] [DABC 4 11 38] [CDAB 7 16 39] [BCDA 10 23 40]
[ABCD 13 4 41] [DABC 0 11 42] [CDAB 3 16 43] [BCDA 6 23 44]
[ABCD 9 4 45] [DABC 12 11 46] [CDAB 15 16 47] [BCDA 2 23 48]
```

Рунда 4 састоји се од последњих 16 корака.

Нека `[abcd k s i]` означава операцију $a \leftarrow b + ((a + I(b,c,d) + X[k] + T[i]) <<< s)$. Ових 16 корака су

```
[ABCD   0   6   49] [DABC   7   10  50] [CDAB   14  15  51] [BCDA   5   21  52]
[ABCD   12  6   53] [DABC   3   10  54] [CDAB   10  15  55] [BCDA   1   21  56]
[ABCD   8   6   57] [DABC   15  10  58] [CDAB   6   15  59] [BCDA   13  21  60]
[ABCD   4   6   61] [DABC   11  10  62] [CDAB   2   15  63] [BCDA   9   21  64]
```

На крају се додају сачуване речи са почетка: $A \leftarrow A + AA$, $B \leftarrow B + BB$, $C \leftarrow C + CC$, $D \leftarrow D + DD$. Излазна вредност f је конкатенација $ABCD$, укупно 128 бита.

19.1.7 Објашњење

Погледајмо најпре корак 1. Пре корака 1 је $A = A_0 = 01\ 23\ 45\ 67$, $B = B_0 = \ldots$, $C = C_0 = \ldots$, $D = D_0 = \ldots$. Подсетимо се да `[abcd k s i]` означава операцију $a \leftarrow b + ((a + F(b,c,d) + X[k] + T[i]) <<< s)$. Како је корак 1 `[ABCD 0 7 1]`, то значи $A \leftarrow B + ((A + F(B,C,D) + X[0] + T[1]) <<< 7)$, односно $A_1 \leftarrow B_0 + ((A_0 + F(B_0,C_0,D_0) + X[0] + T[1]) <<< 7)$. Дакле, најпре се израчунава $F(B_0,C_0,D_0)$. Затим се на то додаје $X[0]$ и $T[1]$ по модулу 2^{32}, где је $X[0]$ почетак 512-битне улазне поруке, а $T[1]$ се добија од $\sin(1)$. Добијени резултат, посматран као реч, циклички се помера за 7 бита улево. На крају се томе додаје B_0 по модулу 2^{32}. Сада је $A = A_1$, $B = B_0$, $C = C_0$, $D = D_0$.

Сада прелазимо на корак 2. Подсетимо се да `[abcd k s i]` означава операцију $a \leftarrow b + ((a + F(b,c,d) + X[k] + T[i]) <<< s)$. Како је корак 2 `[DABC 1 12 2]`, то значи $D \leftarrow A + ((D + F(A,B,C) + X[1] + T[2]) <<< 12)$, односно $D_1 \leftarrow A_1 + ((D_0 + F(A_1,B_0,C_0) + X[1] + T[2]) <<< 12)$. Сада је $A = A_1$, $B = B_0$, $C = C_0$, $D = D_1$. После 16 оваквих пррерачунавања A, B, C, D нека је резултат означен са $A_{16}, B_{16}, C_{16}, D_{16}$.

19.1.8 Хеш алгоритам

За улаз у хеш алгоритам кажемо да је порука. Нека је на пример дужина (допуњене) поруке $3 \cdot 512$ бита и нека се она састоји од три 512-битна блока $S_1S_2S_3$. Најпре узимамо S_1 и разбијамо га на 16 речи $X[0], \ldots, X[15]$. Почињемо са вредностима регистара $A = 0x67452301$, $B = 0xefcdab89$, $C = 0x98badcfe$, $D = 0x10325476$.

Затим извршавамо горе описани алгоритам и добијамо 128-битну реч $ABCD = A'_{16}B'_{16}C'_{16}D'_{16}$. (Излаз из корака $61-64$ је $A_{16}, B_{16}, C_{16}, D_{16}$; после сабирања по модулу 2^{32} $AA = A_0$, $BB = B_0$, $CC = C_0$, $DD = D_0$, излаз је $A'_{16}B'_{16}C'_{16}D'_{16}$.)

Затим се S_2 разбија на 16 речи $X[0], \ldots, X[15]$, наравно у општем случају различитих од претходних речи $X[0], \ldots, X[15]$. Затим се извршава горе описани алгоритам, са том разликом што су почетне вредности A, B, C, D једнаке излазу из претходних 64 корака $A = A'_{16}$, $B = B'_{16}$, $C = C'_{16}$, $D = D'_{16}$, а не $x67452301\ldots$ Тако се добија 128-битни излаз $ABCD = A'_{32}B'_{32}C'_{32}D'_{32}$.

Затим се S_3 разбија на 16 речи $X[0], \ldots, X[15]$ и поново извршава горе описани алгоритам, са том разликом да су почетне вредности A, B, C, D једнаке излазу из претходних 64 корака $A = A'_{32}$, $B = B'_{32}$, $C = C'_{32}$, $D = D'_{32}$. Тако се добија 128-битни излаз $ABCD = A'_{48}B'_{48}C'_{48}D'_{48}$, који је излаз из алгоритма, односно хеш вредност.

Напомене.

1. Технички, да бисмо функцију f произвели у хеш функцију, можемо да пишемо $f_{hash}(X) = f(A_0, B_0, C_0, D_0, X)$. Сада је $f_{hash}(X)$ пресликавање 512-битних блокова у 128-битне блокове.

2. Порука се увек допуњује. Она се најпре допуњује тако да њена дужина буде $-64 \equiv 448 \pmod{512}$. Допуњавање се врши једном јединицом и потребним бројем нула. Чак и ако је дужина поруке $448 \pmod{512}$, додаје се једна јединица и 511 нула. Нека је b дужина поруке пре допуњавања. Бинарна представа броја b дописује се на крај поруке, после чега дужина поруке постаје умножак 512.

   На пример, нека је дужина оригиналне поруке 1700 бита. Пошто је $1770 \equiv 164 \pmod{512}$ и $448 - 164 = 284$, додаје се једна јединица и 283 нуле. Сада је дужина поруке 1984 бита, $1984 \equiv 448 \equiv -64 \pmod{512}$. Даље, пошто је $1700 = (11010100100)_2$, поруци се дописује

   00000000000000000000000000000000000000000000000000000011010100100

тако да је сада дужина поруке $1984 + 64 = 2048 = 4 \cdot 512$.

3. Функција f_{hash} је компликована јер се у сваком кораку наизменично примењују битске операције (F, G, H, I) и сабирање по модулу 2^{32}, операција са целим блоковима (са преносима на више позиције, за разлику од битских операција). Чињеница да се сва четири регистра стално ажурирају и да се делови поруке постепено укључују такође повећава сигурност.

4. Ипак, 2004. године показано је (Wang, Feng, Lai, Yu) да MD5 није јако отпорна на колизије. Ипак је могуће да алгоритам има две потребне особине: једносмерност и слабу отпорност на колизије.

20 Потписи и аутентикација

Размотримо овај сценарио. Цица шаље поруку Амазону и каже "Ја сам Ед". Цица прави два кључа за AES и шаље их Амазону шифроване јавним кључем Амазона. Затим Цица шифрује поруке користећи први кључ за AES. Порука може да буде "Пошаљите ми Енциклопедију Британику, а рачун пошаљите Еду". На крају Цица шаље Амазону MAC поруке добијен другим кључем за AES . Ово није подметање лажне поруке, него *лажно представљање*.

Како Ед може да се заштити? Амазон може да захтева дигитални потпис. Обично се потписује MAC .

Уопште, ако добијете поруку од некога, како можете да будете сигурни да је поруку послала особа која каже да је послала? Поступак који обезбеђује да будете сигурни да порука потиче од правог пошиљаоца зове се *аутентикација*. Аутентикација повезује јавни кључ са особом или институцијом. Решење се заснива на потписима и сертификатима. Обично се за потписе користи систем са јавним кључем.

20.1 Потписи помоћу RSA

Претпоставимо да Џорџ Буш $(= G)$ и Тони Блер $(= T)$ користе RSA . У систему RSA не постоји заједнички кључ који знају само Буш и Блер, па Осама бин Ладен $(= L)$ може да пошаље поруку Бушу "Остави на миру Бин Ладена, Тони Блер", шифровану алгоритмом AES . Како Буш може да зна од кога је порука дошла? Он мора да захтева потпис.

Нека је f_G шифарска трансформација за слање порука Бушу (у систему RSA , $C \equiv f_G(P) = P^{e_G} \pmod{n_G}$). Тада је f_G^{-1} дешифровање које врши Буш (у систему RSA , $P \equiv f_G^{-1}(C) = C^{d_G} \pmod{n_G}$).

Сви знају f_G и f_T. Само G зна f_G^{-1}, само T зна f_T^{-1}.

Случај 1. T шаље поруку P особи G, без шифровања. На крају је потписује "*Toni Bler*". Он жели да G буде сигуран да је порука од њега. T израчунава $f_T^{-1}('Toni\ Bler')$ (што изгледа као шифрат) и завршава поруку тиме. Ово не

може да уради нико други осим T. G може да израчуна $f_T(f_T^{-1}('Toni\ Bler')) =' Toni\ Bler'$. Потенцијални проблем је у томе што и непријатељ може да прочита потпис. Ако T потпише име за L (бин Ладен), онда L може тај део да копира и ставља у своје поруке.

Случај 2. Тони Блер жели да цела порука буде шифрована и потписана. T најпре шаље поруку $'Poruka\ od\ Tonija'$ или, ако не жели да непријатељ сазна ко шаље поруку, онда шаље $f_G('Poruka\ od\ Tonija')$. Он затим шаље $f_G(f_T^{-1}(P))$. Џорџ зна f_G^{-1}, па израчунава $f_G^{-1}(f_G(f_T^{-1}(P))) = f_T^{-1}(P)$. Сви знају F_T, па G израчунава $f_T(f_T^{-1}(P)) = P$. Без прве поруке G би $f_T^{-1}(P)$ сматрао за шифрат и не би знао који кључ за f да искористи. B зна да поруку није послао L, јер он не зна f_M^{-1}.

Случај 3. Тони Блер шифрује отворени текст P алгоритмом AES . Он затим израчунава MAC за отворени текст, па га шифрује и потписује помоћу AES . Претходно је Тони већ послао кључеве за AES и MAC помоћу RSA .

Пример проблема са RSA . Претпоставимо да је $n_T < n_G$. Нека је $P_1 =' poruka\ od\ Tonija'$, а P_2 је стварна порука.

Тони израчунава $P_1^{e_G} \pmod{n_G}$ и шаље то G. Према томе, $0 \leq P_1^{e_G} \pmod{n_G} < n_G$. Тони израчунава $P_2^{d_T} \pmod{n_T}$, где је $0 \leq P_2^{d_T} \pmod{n_T} < n_T$. Затим он израчунава $(P_2^{d_T} \pmod{n_T}))^{e_G} \pmod{n_G}$, број мањи од n_G, и шаље то G.

У наставку ћемо подразумевати да је нпр. $P_2^{d_T}$ резултат свођења степена по одговарајућем модулу (n_T).

Џорџ израчунава $(P_1^{e_G})^{d_G} = P_1 \pmod{n_G}$, па затим $((P_2^{d_T})^{e_G})^{d_G} \stackrel{*}{=} P_2^{d_T} \pmod{n_T}$, и на крају $(P_2^{d_T})^{e_T} = P_2 \pmod{n_T}$.

Проблем је у томе што G не може да уради исту ствар кад шаље поруку T, због $n_G > n_T$. На месту означеном са $* T$ би имао израз $P_2^{d_G} \pmod{n_T}$. При томе је могуће да буде $n_T < P_2^{d_G} < n_G$. Ако је n_G за две цифре дуже од n_T, онда $P_2^{d_G} \pmod{n_T}$ може да буде конгруентан са ≈ 100 различитих бројева по модулу n_G.

Пример. $n_T = 1000$, $n_G = 100000$, и нека $P_2^{d_G}$ има вредност 10008. Број 10008 има јединствени остатак по модулу n_G, али тај број по модулу n_T даје остатак 8. Због тога је немогуће рећи да ли је тај остатак 8, 1008, 2008, ..., 99008. Ових сто вредности имају различите остатке по модулу n_G, али један исти остатак по модулу n_T.

Како се овај проблем може решити? Пошто је $n_G > n_T$, G треба да шаље $f_G^{-1}(f_T(P_2))$. T зна да да је порука од G из дела P_1, и зна да је $n_T < n_G$, па израчунава $f_G(f_G^{-1}(f_T(P_2))) = f_T(P_2)$, после чега примењује f_T^{-1}.

Дакле, кад се порука шифрује, увек се најпре примењује мало n, па велико n.

Пример потписа помоћу RSA . Још једном, најпре се примењује мало n, па велико n.

Нека је $n_G = 221$, $e_G = 187$, $d_G = 115$, и $n_T = 209$, $e_T = 191$, $d_T = 131$.

T жели да потпише и шифрује поруку 97 за G. Шта он треба да ради? Кад се шаље, најпре се примењује мало n, па велико n.

Најпре потпис: $97^{d_T} \pmod{n_T} = 97^{131} \pmod{209} = 108$, па шифровање: $108^{e_G} \pmod{n_G} = 108^{187} \pmod{221} = 56$.

T шаље G шифрат 56; непријатељ види само то, шифрат 56.

G прима 56. Прималац примењује најпре велико n, па мало n. $56^{d_G} \pmod{n_G} = 56^{115} \pmod{221} = 108$. $108^{e_T} \pmod{n_T} = 108^{191} \pmod{209} = 97$.

Сада G жели да потпише и шифрује поруку 101 за T. Кад се шаље, најпре се примењује мало n, па велико n.

Најпре шифровање: $101^{e_T} \pmod{n_T} = 101^{191} \pmod{209} = 112$, па потпис: $112^{d_G} \pmod{n_G} = 112^{115} \pmod{221} = 31$.

G шаље T потпис 31; непријатељ види 31.

T прима 31. Прималац примењује најпре велико n, па мало n. $31^{e_G} \pmod{n_G} = 31^{187} \pmod{221} = 112$. $112^{d_T} \pmod{n_T} = 112^{131} \pmod{209} = 101$. У пракси се обично потписује MAC .

20.2 Ел Гамалов потпис

Алгоритам потписивања Ел Гамал послужио је као основа за нешто сложенији стандард дигиталног потписа, DSS , Digital Signature Standard. Полази се од великог простог броја p и генератора g за $\mathbf{F}_p^*$, као и тајних кључева a_A, a_B, ..., односно јавних кључева g^{a_A}, g^{a_B}, ... Ови елементи не треба да се често мењају.

Претпоставимо да Алиса жели да пошаље поруку Бобану (ово може да буде MAC поруке). Нека је S порука, при чему је S број, $1 < S < p$. Алиса бира случајни број k, $1 < k < p$, $\mathrm{nzd}(k, p-1) = 1$, и израчунава $g^k \pmod{p} = r$.

Затим она решава по x конгруенцију $S \equiv a_A r + kx \pmod{p-1}$. Према томе, $x \equiv k^{-1}(S - a_A r) \pmod{p-1}$. Приметимо да је $g^S \equiv g^{a_A r + kx} \equiv g^{a_A r} g^{kx} \equiv (g^{a_A})^r (g^k)^x \equiv (g^{a_A})^r r^x \pmod{p}$.

Алиса шаље r, x, S Бобану као потпис. Боб утврђује да је порука од Алисе израчунавањем $(g^{a_A})^r r^x \pmod{p}$ и $g^S \pmod{p}$, односно провером да ли су ове две вредности идентичне. Сада Бобан зна да је порука од Алисе. Зашто? Само Алиса је могла да реши $x \equiv k^{-1}(S - a_A r) \pmod{p-1}$, јер једино она зна a_A.

Пример. $p = 677$, $g = 2$. Нека је MAC поруке $316 = S$. Алисин приватни кључ је $a_A = 307$, њен јавни кључ је $2^{307} \pmod{677} = 498$. Дакле, $g^{a_A} = 498$. Она бира кључ поруке $k = 401$ (што је у реду, јер је $\mathrm{nzd}(k, p-1) = 1$). Алиса израчунава $r = g^k = 2^{401} \equiv 616 \pmod{p}$, па је $r = 616$. Она решава конгруенцију $S = a_A r + kx \pmod{p-1}$, односно $316 = 307 \cdot 616 + 401 \cdot x \pmod{676}$. Дакле, $x \equiv 401^{-1}(316 - 307 \cdot 616)$. Сада је $401^{-1} \equiv 617 \pmod{676}$, па је $x \equiv 617(316 - 307 \cdot 616) \equiv 512 \pmod{676}$. Алиса шаље $(r, x, S) = (616, 512, 316)$. Бобан прима ту тројку и израчунава $g^S = 2^{316} \equiv 424 \pmod{677}$. $(g^{a_A})^r = 498^{616} \equiv 625 \pmod{677}$. $r^x = 616^{512} \equiv 96 \pmod{677}$. Он утврђује да је $g^{a_A r} g^{kx} \equiv 625 \cdot 96 \equiv 424$ исто што и $g^S \equiv 424 \pmod{677}$. Крај примера.

20.3 Шноров поступак аутентикације и потписа

Нека је p прост број, и нека је q прост број, $q|p-1$. Изаберимо a тако да буде $a^q \equiv 1 \pmod{p}$. Бројеве a, p, q користе сви учесници система. Свака особа има приватни кључ s и јавни кључ $v \equiv a^{-s} \pmod{p}$.

Аутентикација: Особа A (којој треба проверити идентитет) бира случајни број $r < q$ и израчунава $x \equiv a^r \pmod{p}$. Она шаље x особи B (коју треба да увери у свој идентитет). B шаље A случајни број e, $0 \leq e \leq 2^t - 1$ (величина t биће објашњена касније). A израчунава $y = r + s_A e \pmod{q}$ и враћа то особи B. B израчунава $a^y v_A^e \pmod{p}$, и проверава да ли је то једнако x. На тај начин се установљава да је особа која је послала y управо особа чији је јавни кључ v_A.

Шнор (Schnorr) Предлаже да се ради са вредностима параметара $p \approx 2^{512}$, $q \approx 2^{140}$ и $t = 72$.

Потпис: Алиса жели да потпише поруку M. Она бира случајно $r' < q$ и израчунава $x' = a^{r'} \pmod{p}$. Алиса надовезује M и x', па применом хеш функције добија e'. Затим израчунава $y' = r' + s_A e' \pmod{q}$. Потпис је тројка x', e', y'. Бобан има M (то је или шифровано за њега, или је порука послата нешифрована). Он израчунава $a^{y'} v_A^{e'} \pmod{p}$ и проверава да ли је то једнако e'. На крају он проверава да ли је e' једнако хеш вредности конкатенације Mx'.

Израчунавање $y = s_A e \pmod{q}$ је брзо и не укључује инверзију по модулу. Алиса може да израчуна x унапред и тако убрза време израчунавања потписа. Сигурност се по свему судећи заснива на тежини ФФДЛП. Потписи су краћи него са RSA за исти ниво сигурности (јер се израчунавања врше по модулу q, док се ФФДЛП решава у већем пољу $\mathbf{F}_p$).

20.4 Временски печат

Ако имате штампани документ и желите да докажете да је постојао одређеног датума, морате да га заведете (на пример) у суду. Ово је важно за документе за које ви имате ауторска права, да бисте доказали ауторство. Проблем је тежи са дигиталним подацима. Ако ти подаци садрже датум, датум се лако може заменити другим. Решење је *временски печат* (time-stamping). Приказаћемо сада протокол израчунавања временског печата.

Нека је Тома овлашћен да издаје временске печате. Нека је A Алисино име, и нека је Алисин захтев n-ти захтев који Тома обрађује.

1. Алиса израчунава хеш вредност H_n свог документа.
2. Алиса шаље Томи H_n и A. Претпоставимо да је особа са именом I_{n-1} била особа која је од Томе тражила временски печат непосредно пре Алисе. Нека је H_{n-1} хеш вредност коју је Томи послао I_{n-1}, и нека је T_{n-1} временски печат те поруке.
3. Тома израчунава L_n као хеш вредност четворке $(I_{n-1}, H_{n-1}, T_{n-1}, L_{n-1})$, где је L_{n-1} је претходна таква хеш вредност. Нека је t_n време кад Тома израчунава временски потпис Алисиног документа.

4. Тома потписује поруку $(n, A, H_n, t_n, I_{n-1}, H_{n-1}, T_{n-1}, L_n)$; то је T_n, временски потпис Алисиног документа.

5. После израчунавања $n+1$-ог временског печата Тома шаље Алиси име особе која је креирала $n+1$-и документ, тј. I_{n+1}.

Разлог због кога се Алиси шаље име наредне особе која је тражила временски печат је спречавање Алисе и Томе да заједно направе злоупотребу. Без оваквог повезивања са именима претходног и наредног клијента, Алиса би могла да подмити Тому и да затражи од њега да антидатира њен документ.

Ако Алиса потпише поруку, а касније одлучи да повуче свој потпис (и то је једна врста злоупотребе), она може да објави свој приватни кључ, и да каже да је било ко могао да израчуна потпис њеног документа. Ово се зове *одрицање* (repudiation) Због тога, ако неко добије потпис од Алисе, он може да захтева да потпис укључује и временски печат. То умањује могућност Алисе да се одрекне свог потписа.

20.5 КЕРБЕРОС

Протокол је низ корака које извршавају бар два учесника, чиме се извршава неки задатак. КЕРБЕРОС је протокол аутентикације који *не користи криптографију са јавним кључем*. Користи се у затвореним системима.

Протокол КЕРБЕРОС омогућује употребу лозинке за добијање приступа некој услузи. Нека је C клијент, K сервер КЕРБЕРОС, S сервер који пружа услугу коју захтева C и нека је TGS сервер за издавање овлашћења за услугу S (ticket granting service).

Протокол КЕРБЕРОС састоји се од две основне фазе:

а) КЕРБЕРОС проверава идентитет клијента C за сервер TGS.

б) TGS издаје овлашћење клијенту C за услугу сервера S.

Неопходне ознаке: ако је X учесник (тј. C, K, S или TGS), онда K_X означава његову лозинку, а x је порука која садржи име тог учесника. Нека $K_{X,Y}$ означава кључ који се користи за размену порука између X и Y. Нека $\{m\}K_X$ означава шифрат поруке m кључем K_X.

Претпоставке: само C и K имају кључ K_C; само K и TGS имају кључ K_{TGS}; само TGS и S имају кључ K_S.

1. C шаље K поруку: c, tgs.

   C можете да будете ви, а TGS може да буде linux сервер на који хоћете да се пријавите. У том случају овај корак се састоји у томе да ви укуцате своје корисничко име и лозинку, после чега ваш рачунар извршава корак 1.

2. K шаље C: $\{K_{C,TGS}\}K_C, tgs, \{c, a, v, K_{C,TGS}\}K_{TGS}$, где је a мрежна адреса клијента (то је адреса рачунара на коме C ради — тај рачунар може да користи више особа, као што је нпр. Unix сервер; v је период

важења овлашћења. Порука $\{c, a, v, K_{C,TGS}\}$ се зове овлашћење за издавање овлашћења (ticket-granting ticket, TGT).

3. C шаље TGS: tgs, $\{c, a, v, K_{C,TGS}\}K_{TGS}$, $\{c, t\}K_{C,TGS}$. Овде је t временски печат, а $\{c, t\}$ се зове аутентикатор.

4. TGS шаље C: $\{K_{C,S}\}K_{C,TGS}$, $\{c, a, v, K_{C,S}\}K_S$

5. C шаље S: $\{c, a, v, K_{C,S}\}K_S, \{c, a, t\}K_{C,S}$

Овлашћење за издавање овлашћења потврђује идентитет клијента C серверу за издавање овлашћења за услугу S. После тога сервер за издавање овлашћења даје овлашћење клијенту C за коришћење услуге S.

20.6 PKI

Инфраструктура система са јавним кључем (PKI , public key infrastructure) омогућује учесницима у мрежи да буду сигурни у погледу идентитета осталих учесника. То је аутентикација. Ако Алиса жели да провери идентитет Бобана, онда PKI обезбеђује одговарајући софтвер на њеном и његовом рачунару, и евентуално хардвер. PKI користи сертификационе низове или мрежу поверења и одређене протоколе. У наставку ћемо објаснити сертификационе низове и мрежу поверења.

20.6.1 Сертификати

Ед користи своју картицу Bank of America (BA) у филијали Fargo Bank of North Dacota (BND). BND se povezuje sa BA. BA i BND koriste RSA да успоставе заједнички кључ за AES . BND шифрује поруку "Скините $60 са Едовог рачуна у Банци Северне Дакоте да надокнадите нашу готовину". Како BND зна да јавни кључ за RSA стварно припада BA, а не неком хакеру?

Фирма Верисајн (Verisign, Mountain View, V), је аутентикациони центар (CA, certification authority). Приказаћемо укратко како све ово функционише.

BA је направила јавни кључ и документ у коме пише "јавни кључ за RSA припада Bank of America". Затим представници BA одлазе у надлежну установу са идентификационим документима. Документ се заводи, кључ за RSA шаље се у V.

V користи свој јавни кључ да потпише следећи документ:
Version: V3
Serial Number: (низ хексадекадих цифара)
Issuer: Verisign
Valid from: April 7, 2006
Valid to: April 6, 2007
Subject: bankofamerica.com
Signature Algorithm: MD5/DSS
Public key for signing: X (1024 бита кључа a_{BA} за DSS у облику низа хексадекадних цифара)

Encryption Algorithm: RSA

Public key for encryption: Y (1024 бита за n_{BA}, кратки e_{BA}, у облику низа хексадекадних цифара)

Certification path: Bank of America, Verisign.

Hash: (потпис V хеш вредности свих претходних података из овог сертификата).

BND такође користи услуге V. BND има поверење у V и јавне кључеве V. BND проверава потпис на крају сертификата BA помоћу јавног кључа V (BA такође проверава потпис V испод сертификата BND).

Прецизније, постоји сертификационо стабло са V (Верисајн) на врху. На универзитету може да постоји аутентикациони центар (CA) нижег нивоа, који користи свој јавни кључ да потпише сертификате сваког корисника сервера на универзитету.

Претпоставимо да је у горњем примеру Едов сертификат потписао нижи аутентикациони сервер на универзитету Санта Клара SCU, а да је сертификат SCU потписао V. Едов рачун је у $BASC$ (Bank of America's Santa Clara branch). Ед користи свој браузер да се пријави на сајт своје банке. Његов браузер најпре проналази сертификат $BASC$, сличан горе наведеном, изузев што у њему пише

Subject: bofasantaclara.com

Public key for signing: Z

Public key for encrypting: A

Certification path: Bank of America Santa Clara, Bank of America, Verisign.

Hash: (потпис BA (кључем X) хеш вредности свих претходних података из овог сертификата).

Едов браузер зна јавни кључ V, јер и Едов браузер користи Верисајн. Из сертификационог пута Едов браузер зна да треба да пронађе сертификат BA. Едов браузер верификује потпис V на хеш вредности сертификата BA. Сада Едов браузер проналази кључ за потписивање X који припада BA. Едов браузер верификује потпис BA на хеш вредности сертификата $BASC$. Сада Едов браузер проналази кључ A за *шифровање* који припада BA, и сада је сигуран да је A кључ за шифровање $BASC$. Кад сада сајт банке $BASC$ затражи Едову лозинку (која ће бити шифрована кључем A), Едов браузер неће уземиравати Еда упозоравајућом поруком.

У пракси постоје различити нивои сертификације. За озбиљан ниво, ви морате да покажете своју личну карту. Може се добити сертификат нижег нивоа, који у основи садржи информацију "e-mail адреса eschaefer@hotmail.com и јавни кључ n_E, e_E су повезани". Тај сертификат не гарантује да је електронска адреса повезана са особом која се зове Ед Шефер.

Не користе сви Верисајн. Претпоставимо да Финска и Иранска влада имају своје аутентикационе сервере. Сваки од њих може да верификује сертификат оног другог. То се зове *узајамна сертификација*.

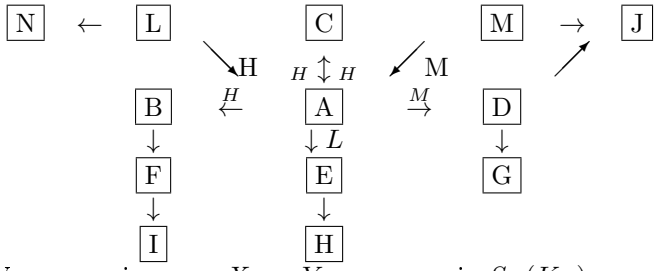
Верисајн може да опозове сертификат у било ком тренутку. Верисајн чува списак опозваних сертификата. Један од разлога због којих Верисајн може да опозове сертификат је губитак тајности јавног кључа (тј. постоји опасност да неко други искористи кључ који је записан код Верисајна).

20.6.2 PGP и мрежа поверења

PGP је програм за шифровање електронске поште, који је дистрибуирао Цимерман (Zimmerman) 1991. године. Програм користи асиметричну и симетричну криптографију, потписе и хеш функције. Најинтересантнији део је да програм уместо сертификационих ланаца користи *мрежу поверења*. Мрежа поверења је добра за појединачне кориснике, али се не користи за пословне примене.

Корисници сами одлучују коме верују. Сваки корисник чува листу потписаних кључева. Нека је $S_X(K_Y)$ потпис корисника X на јавни кључ корисника Y (у пракси, користи се цео сертификат, чији је део $S_X(K_Y)$). Сваки корисник има *прстен јавних кључева*. На пример, прстен јавних кључева Алисе садржи кључеве особа са којима Алиса жели да комуницира (и којима Алиса верује), са потписом Алисе. Прстен јавних кључева Алисе може да изгледа овако:

име	потписани кључ	ниво поверења у кључ	мера поверења у његове потписе кључева	мера поверења потписивача у серификате других кључева
Бобан	$S_A(K_B)$	Висок	Висок	
Цица	$S_A(K_C)$	Висок	Висок	
Деа	$S_A(K_D)$	Висок	Средњи	
Ед	$S_A(K_E)$	Висок	Низак	
Цица	$S_C(K_A)$			Висок
Лаза	$S_L(K_A)$			Висок
Марко	$S_M(K_A)$			Средњи



У овом дијаграму $X \to Y$ значи да је $S_X(K_Y)$ у прстеновима и X и Y. Даље, $X \overset{H}{\to} Y$ значи да је у оба прстена, а X има потпуно поверење у сертификате других јавних кључева који потичу од Y.

Претпоставимо да Алиса жели да пошаље поруку F. A контактира са F, и F шаље свој прстен A. Прстен учесника F садржи $S_B(K_F)$. Пошто A верује серификатима других кључева који потичу од B, сада A има поверење у јавни кључ учесника F.

Алиса жели да пошаље поруку G. Прстен учесника G садржи $S_D(K_G)$. Међутим, Алиса има осредње поверење у D-ове сертификате других кључева. Због тога Алиса нема поверења у G-ов јавни кључ. Слично, A нема поверење у H-ов јавни кључ.

Алиса жели да пошаље поруку J. Прстен учесника J садржи $S_D K_J$ и $S_M K_J$. Алиса има осредње поверење у D-ове и M-ове сертификате других

кључева, а две осредње оцене имају за последицу да Алиса нема поверења у J-ов јавни кључ.

Алиса жели да пошаље поруку I. Прстен учесника I садржи $S_F K_I$. Алиса има поверења у јавни кључ особе F, али нема основа да верује F-овом сертификату I, па A нема поверења у јавни кључ I.

Алиса жели да пошаље поруку N. Прстен учесника N садржи $S_L K_N$. Алиса има поверења у L-ове сертификате, па има поверење у јавни кључ особе N.

Ако Алиса шаље поруку F, PGP ће то дозволити. Ако Алиса шаље поруку I, PGP ће избацити поруку да она нема основа за поверење у јавни кључ особе I.

Рецимо да C жели да пошаље поруку B. B шаље C свој прстен, који садржи $S_A(K_B)$. Сада C има поверење у сертификате које добија од A. Дакле, C има поверења у јевни кључ B. Дакле, на неки начин је Алиса (и сваки други учесник) нека врста аутентикационог сервера.

Један од проблема са мрежом поверења је у вези са опозивањем кључева. Немогуће је гарантовати да нико неће користити опозвани кључ (пошто је дистрибуција кључева адхок).

PGP користи RSA усаглашавање кључева, DSS (DSA) за потписивање и SHA и MD5 за потписивање.

20.7 Сигурност на интернету

Два основна протокола за обезбеђење сигурности на интернету су SSL (Secure Sockets Layer) и IPSec .

20.7.1 SSL

Како се криптографија стварно примењује на вебу? Поступак који се примењује зове се SSL , а проналазач је Тахер ЕлГамал из Нетскејпа. Кад видимо https:, то значи да се примењује SSL . Постоји неколико протокола за његову примену. Описаћемо укратко како се то ради у Нетскејпу.

Ако сте на https: сајту (s је прво слово од secure) у Интернет експлореру можете да кликнете на *file*, па *properties*, па *certificates*, па *details*, и онда можете да видите сертификат тог сајта, који потврђује да јавни кључ у сертификату припада фирми чије име се појављује на сајту.

1. Кад се Бобан први пут конектује на Амазон, он шаље Амазону свој сертификат, у коме се налази јавни кључ за RSA n_B, e_B и јавни кључ g^{a_B} за потпис помоћу алгоритма DSS . Вредност p за DSS је стандардна и користе је сви који користе Нетскејп. Подсетимо се да је DSS само варијанта система ЕлГамал, па користимо нотацију из система ЕлГамал. Сертификат повезује Бобаново име са његовим јавним кључевима.

2. Амазон користи Бобанов јавни кључ за RSA да шифрује кључ за AES и кључ за MAC . Према томе, Амазон спаја два кључа и то претвара

у број. То је једноставно, низ бита се посматра као број a у систему са основом два. Затим Амазон израчунава $b = a^{e_B} \pmod{n_B}$ и шаље то Бобану.

3. Бобан дешифрује поруку b, израчунавши $a = b^{d_B} \pmod{n_B}$, и тако добија кључ за AES и кључ за MD5–MAC .

4. Бобан затим шифрује своју поруку за Амазон. Та порука може да садржи његову поруџбину, број кредитне картице, адресу и др. Порука ће бити шифрована алгоритмом AES , кључем добијеним од Амазона. Израчунати шифрат Бобан шаље Амазону.

5. Бобан одређује MD5–MAC отвореног текста послате поруке, користећи кључ за MAC који је добио од Амазона. Он користи DSS да потпише овај MAC . Структура ове поруке је S =MAC , r, x. Бобан затим шифрује потпис алгоритмом AES . Дакле, Бобан спаја S =MAC , r и x и шифрује их алгоритмом AES . Овај шифровани, па потписани MAC Бобан шаље Амазону.

6. Амазон прима поруку из 4., дешифрује је и добија отворени текст поруке.

7. Амазон затим одређује MAC примљене поруке користећи кључ за MAC.

8. Амазон затим дешифрује потписани шифровани MAC добијен од Бобана, применом алгоритма AES . Амазон тако добија S =MAC , r, x.

9. Амазон затим проверава да ли је MAC добијен из 7. исти као MAC из 8. На тај начин се проверава да поруку није неко променио.

10. Амазон проверава потпис у 8. Подсетимо се да се потпис проверава утврђивањем да ли су $(g^{a_B})^r \cdot r^x \pmod{p}$ и $g^S \pmod{p}$ једнаки. Сада Амазон зна да је потписани MAC добио од Бобана.

Напомене. Уместо RSA , могу се користити протокол Дифи–Хелман над коначним пољем (што се тренутно најчешће користи) или протокол Дифи–Хелман са елиптичким кривама. За протокол Дифи-Хелман, Бобан и Амазон морају најпре да израчунају Дифи-Хелман јавни кључ. Уместо AES неки протоколи још увек (2006. године) користе троструки DES . Ако Бобану понестане пара пошто је послао поруџбину, он не може да је се одрекне (протокол онемогућује *одрицање*). Само је Бобан могао да потпише MAC , и свако може да се увери да је то урадио Бобан, јер се потпис може да прочита једино Бобановим јавним кључем (што значи да је једино Бобан могао да га направи користећи свој тајни кључ).

Уствари, Бобан може да се одрекне поруџбине тако што објави свој тајни кључ. Тада он може да тврди да је било ко могао да фалсификује његов потпис. С друге стране, Амазон може да докаже да је Бобан разгласио свој

тајни кључ после поруџбине; наравно, ако Бобан неће да објави свој јавни кључ, он не може да порекне поруџбину.

20.7.2 IPSec

IPSec (Internet Protocol security) је конкурент протоколу SSL . Он у односу на SSL ради на другом нивоу (комуникационог протокола; не улазимо у детаље), и омогућује већу флексибилност. Међутим, IPSec је мање ефикасан од SSL . Првенствено је намењен за употребу у *виртуелним приватним мрежама* (VPN, Virtual Private Network). Виртуална приватна мрежа омогућује заштићену комуникацију неколико учесника глобалне мреже (коришћењем инфраструктуре интернета или неке друге мреже). SSL користе сви учесници интернета.

20.8 Управљање кључевима

У пракси нападачи скоро никад не проналазе кључеве применом криптоанализе. Уместо тога, они користе алкавост при избору кључева. Много је једноставније подмитити особу и тако добити кључ, или украсти кључ, него га одредити криптоанализом. Већина људи не бира кључеве на случајни начин. Цитат: "Док сам био млађи, проваљивао сам у рачунаре, претпостављајући да је лозинка пријатеља мог оца име његове најстарије ћерке." Други пример: потребно је око један сат да се на рачунару грубом силом одреди нечији кључ ако се кључ састоји од само седам малих слова (тј. седам бајтова). Међутим, ако се уместо тога користи седам бајтова из генератора случајних бројева, онда је за напад грубом силом потребно 457 година. Међутим, већине људи не воли да памти лозинке као што је 8*&u!=}. Они ће вероватно да напишу лозинку на папир и ставе је у новчаник или торбу, али то онда може да буде украдено.

Хеш вредности лозинки се често смештају на рачунару нешифроване, заједно са корисничким именом. Кад укуцате своју лозинку, рачунар израчунава њену хеш вредност, коју затим упоређује са вредношћу смештеном уз ваше корисничко име. Нападач може да израчуна хеш вредности свих речи дужине 7 или 8 састављених од малих слова, и да упореди ове хеш вредности са онима из фајла. Може да се деси да само један корисник има овакву слабу лозинку. Међутим, на неким системима је довољно да имате само једну лозинку, па да провалите у рачунар и направите штету. Пошто вам је потребна само једна лозинка, још је брже да покушате са свим обичним речима или (енглеским) именима дужине до 7 или 8 слова. Ако има довољно корисника, вероватно ће нека таква реч бити коришћена као лозинка. Ово је *речнички напад* (dictionary attack).

Неки рачунари не дозвољавају да се користе лозинке које се састоје само од слова. Уобичајено је да људи користе трикове као што су E1izabeth, Tayl0r, и слично. Речник који се користи за напад може се проширити да обухвати и овакве промењене речи.

Да се избегне речнички напад, може се применити поступак *досољавања*. Сваком кориснику је додељен случајни низ знакова, који се надовезује на лозинку, па се израчунава хеш вредност резултата. Корисничко име, случајни низ и хеш вредност се смештају у рачунару. Сада број могућих хеш вредности није више једнак броју могућих лозинки, него производу броја могућих лозинки и броја могућих случајних низова.

Ни један кључ не сме се користити неограничено дуго. Што се дуже користи, више докумената је њиме шифровано, па долази до веће штете приликом губитка кључа. Што се кључ дуже користи, већи је изазов његово разбијање, и веће време има нападач да спроведе напад.

20.9 Квантна криптографија

Постоје два начина да се без непосредног сусрета договори кључ за симетричну шифру. Први је употреба криптографије са јавним кључем, која је заснована на математици. Други је квантна криптографија.

Фотон има поларизацију — то је раван α која садржи правац његовог кретања. Нека је β нека раван нормална на правац кретања фотона. Поларизација је одређена углом пресечне праве p (двеју равни α, β) — координатне осе нове базе и старе координатне (x) осе. База може да буде (у односу на стару) ректилинеарна (хоризонтална или вертикална), дијагонална (нагиб 1 и -1), итд. Ако се поларизација фотона мери у погрешној бази, онда се добија случајан резултат и уноси се грешка у сва наредна мерења поларизације тог фотона.

Нека је потребно да Алиса и Бобан договоре симетрични кључ. Алиса шаље Бобану низ фотона. Сваки фотон има случајно изабрану поларизацију у једном од четири правца: $|, -, \backslash, /$. Овим правцима приписују се вредности редом $1, 0, 1, 0$. Нека је Алиса послала низ: $\backslash / | \ | / \backslash | - - \backslash - | /$.

Бобан има детектор поларизације. За сваки фотон он случајно бира базу, ректилинеарну или дијагоналну. Нека је низ његових избора $\times + + \times \times + + +$ $\times \times \times + +$. Сваки пут кад изабере добру базу, он тачно мери поларизацију. Ако је база погрешна, онда он добија случајни резултат. Излаз његовог детектора могао би да буде $\backslash - | \backslash / / | - / \backslash \backslash | |$.

Алиса шаље	$\backslash$	$/$	$\|$	$\|$	$/$	$\backslash$	$\|$	$-$	$-$	$\backslash$	$-$	$\|$	$/$
Бобан поставља	$\times$	$+$	$+$	$\times$	$\times$	$+$	$+$	$+$	$\times$	$\times$	$\times$	$+$	$+$
Тачно	≀		≀		≀		≀	≀		≀		≀	
Бобан добија	$\backslash$	$-$	$\|$	$\backslash$	$/$	$-$	$\|$	$-$	$/$	$\backslash$	$\backslash$	$\|$	$\|$

Приметимо да ако Бобан добро изабере базу, онда Алиса и Бобан имају исту поларизацију, која се претвара у 0 или 1. На примеру другог и последњег фотона видимо примере случајности у Бобановом мерењу ако је база изабрана погрешно.

Сада Бобан шаље Алиси отворену поруку, у којој јој јавља своја постављања базе. Алиса му одговара на које оријентације су биле тачне; остале се одбацују.

Алиса шаље	$\backslash$	$\|$	$/$	$\|$	$-$	$\backslash$	$\|$
Бобан добија	$\backslash$	$\|$	$/$	$\|$	$-$	$\backslash$	$\|$

Ове вредности се претварају у нуле и јединице:

Алиса шаље	1	1	0	1	0	1	1
Бобан добија	1	1	0	1	0	1	1

У просеку Алиса шаље Бобану $2n$ бита, да би на крају остало n бита после одбацивања оних за које је база погрешно постављена. Дакле, да би разменили 128-битни кључ за, Алиса мора у просеку да пошаље 256 бита.

Шта се дешава ако Цица мери послате фотоне? Обратимо посебно пажњу на фотоне којима је Бобан тачно погодио базу. За половину тих фотона Цица ће погрешно претпоставити базу. Кад год она погрешно постави базу, она ће Бобанов резултат мерења учинити случајним, уместо тачним.

Алиса шаље	\	|	/	|	−	\	|
Бобан поставља	×	×	×	×	+	+	+
Цица поставља	×	+	×	+	+	×	+
Бобан добија	\	−	/	|	−	/	|

Алиса шаље	1	1	0	1	0	1	1
Бобан добија	1	0	0	1	0	0	1

За други и четврти фотон, пошто је Цица погрешно оријентисала базу, Бобан добија случајне (дакле тачне са вероватноћом 1/2) бите. Дакле, ако Цица прислушкује, очекујемо да она понекад погрешно оријентише базу, па да у неким од тих случајева Бобан добије погрешну поларизацију.

Да би открили прислушкивање, Алиса и Бобан могу да се договоре да провере неке бите. У горњем примеру они могу да се договоре да у отвореном делу поруке кажу која су три бита послата. Алиса би на пример рекла 110, а Бобан 100, после чега би они знали да је неко ометао везу. После тога би они цео поступак покренули из почетка, покушавајући да некако спрече Цицу да прислушкује.

У случају да су се сложили отворено послати бити, они би могли да искористе преостала четири бита за кључ. Наравно, могуће је да Алиса и Бобан добију поклапање три бита, иако Цица прислушкује. Вероватноћа да се много бита сложи ако Цица прислушкује је врло мала. Ако се не сложе, онда ће они знати да су прислушкивани. Ако се сложе, онда они са великом сигурношћу знају да нису прислушкивани. Они би дакле одбацили контролне бите, а остале бите искористили за кључ.

У овом тренутку квантна криптографија ради на растојању од неколико километара. Квантна криптографија сматра се сигурнијом од криптографије са јавним кључем и има уграђену могућност откривања прислушкивања. Међутим, тешко је пренети много информација на овај начин, па се овај поступак користи само за договарање симетричног кључа (нпр. за AES).

20.10 Дељење тајне

Претпоставимо да у Атланти има пет менаџера Кока коле који деле тајну рецепта кока коле. Потребно је да ни један од њих не зна тајну, јер би могао да да отказ и оде у Пепси. Тајна се може кодирати битским низом S дужине l. Затим може да се изгенерише четири случајна низа бита исте дужине R_1, R_2, R_3, R_4. Првој четворици менаџера даје се по један од низова

R_1, R_2, R_3, R_4, а петом $R_1 \oplus R_2 \oplus R_3 \oplus R_4 \oplus S$. Приметимо да нико од њих појединачно нема тајну S, али ако се саберу ($\oplus$) подаци све петорице, добија се S. Било која четворица не могу добити S. Међутим, ако један од њих умре и однесе своју тајну у гроб, онда је тајна заувек изгубљена. Због тога је згодно обезбедити да нпр. било која тројица од њих могу заједно да добију S.

Овај проблем може се решити применом Лагранжовог интерполационог полинома, проналазак Шамира (S из RSA). За решење је потребан број p дужине веће од l (тј. $p \geq 2^l$). Креира се полином са три коефицијента $f(x) = ax^2 + bx + S$. Овде је $S \in [0, 2^l - 1]$ сада цели број. Бројеви a и b су изабрани случајно између 0 и $p - 1$. Менаџеру i даје се остатак $f(i) \pmod p$, $i = 1, 2, \ldots n$. Ово решење ради за сваки број менаџера $n \geq 3$ ако треба обезбедити да било која три менаџера добију тајну.

Пример. Тајни број је 401. Бира се прост број $p = 587$, $a = 322$, $b = 149$. Према томе $f(x) = 322x^2 + 149x + 401 \pmod{587}$. Тада је $f(1) = 285$, $f(2) = 226$, $f(3) = 224$, $f(4) = 279$, $f(5) = 391$. Први, четврти и пети менаџери могу да се договоре да заједно добију тајну S. Они знају $f(x) = ax^2 + bx + S \pmod{587}$, али не знају a, b, S. Они знају $f(1) = a + b + S = 285$, $16a + 4b + S = 279$ и $25a + 5b + S = 391$. Они решавају систем

$$\begin{bmatrix} 1 & 1 & 1 \\ 16 & 4 & 1 \\ 25 & 5 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ S \end{bmatrix} = \begin{bmatrix} 285 \\ 279 \\ 391 \end{bmatrix} \pmod{587}$$

да би одредили a, b, S. Њих занима само S. Решење се може уопштити тако да ради за било који број менаџера, и за било који најмањи број менаџера потребан да се реконструише тајна.

21 Криптоанализа

Криптоанализа је разбијање шифри или проучавање разбијања шифри.

Шифарски системи могу се поделити у три категорије:

1. Они који су разбијени (већина).
2. Они који нису до сада били анализирани (зато што су нови и нису широко коришћени).
3. Они који су анализирани, али нису разбијени. (RSA , системи који користе дискретни логаритам, троструки DES , AES).

Најчешћа три начина да непријатељ добије отворени текст који одговара неком шифрату су

1. Крађом, куповином кључа, односно подмићивањем.
2. Коришћењем слабости у реализацији, односно проблема са протоколом. На пример, неко користи име супружника као кључ, или неко пошаље кључ заједно са поруком.

3. Криптоанализом.

Разматраћемо само трећи од ових начина, иако се он најређе примењује. Постоје три врсте криптоанализе.

Напад са познавањем само шифрата Непријатељ је пресрео шифрат, али нема одговарајући отворени текст. Уобичајена је претпоставка да непријатељ има приступ шифрату. Могуће су две ситуације:

а) Непријатељ зна алгоритам шифровања, али не зна кључ. Ово важи за већину система у комерцијалној употреби.

б) Непријатељ не зна алгоритам шифровања. Опрезни корисници се никада не ослањају на претпоставку да ће ова ситуација потрајати дуго. На пример, алгоритам Скипџек (Skipjack) на чипу Клипер (Clipper) је тајни. Често се војни шифарски системи држе у тајности докле год је могуће. Фирма RSA покушала је да сачува у тајности неколико својих система, али су то спречили објављивањем чланови групе Cipherpunks.

Напад са познатим отвореним текстом Непријатељ има неке парове шифрат, отворени текст. Непријатељ може поред тога имати још шифрата.

Напад са познавањем изабраног отвореног текста Претпоставља се да непријатељ може изабрати отворени текст који жели да пропусти кроз процес шифровања. Иако је овакав напад мало реалан, он има теоријски значај, јер ако се зна довољно отвореног текста, онда постају употребљиве технике за напад са изабраним отвореним текстом. Поред тога, овакав напад је могућ кад се украде паметна (нпр. кредитна) картицу.

У овом курсу криптографије нећемо много времена посветити класичној криптоанализи. Анализираћемо неколико савремених криптоаналитичких метода.

Пројектанти шифарских система често праве погрешне претпоставке о тежини разбијања шифарског система. Они пројектују шифарски систем и закључе да "непријатељ мора да уради x да би разбио овај систем". Често постоји други начин да се систем разбије. Размотримо пример шифре просте замене, где се свако слово замењује неким другим словом азбуке, на пример, $A \to F$, $B \to S$, $C \to A$, $D \to L$, ... Кажемо да је ово *шифра моноалфабетске замене*. Број пермутација 26 слова које не пресликавају ни једно слово у себе је око 1.4×10^{26}. Аутори овог система оцењују да је број ових пермутација тако велики, да је систем сигуран. Међутим, они превиђају да у сваком језику постоји велика правилност.

У класичној криптоанализи се правилности у језику често користе. На пример, слова, биграми и триграми у енглеском (ни нашем) језику немају равномерну расподелу вероватноћа. Ипак, њихова расподела вероватноћа

варира од текста до текста. Наводимо примере ових вероватноћа (претпоставља се да су размаци уклоњени из текста): E - 12.3, T - 9.6, A - 8.1, O - 7.9, ..., Z - 0.01. Најчешћи биграми су TH - 6.33, IN - 3.14, ER - 2.67, ..., QX - 0. Најчешћи триграми су THE - 4.73, ING - 1.42, AND - 1.14, ... У неким табелама се као најчешћи биграми наводе TH, HE, IN, ER, јер резултат зависи од узорка. Савремене шифре као што је AES раде са блоковима од по 16 ASCII знакова који укључују велика и мала слова, размаке, цифре, интерпункцију и др. Најчешћи парови симетричних биграма (у енглеском) су $ER/RE, ES/SE, AN/NA, TI/IT, ON/NO$, и др. Приметимо да сви они садрже самогласник.

Ако је на располагању довољно дуг шифрат добијен моноалфабетском шифром замене, онда се могу пребројати учестаности појављивања слова у шифрату и претпоставити да је најчешће слово E, следеће најчешће T, итд. И ако је дужина шифрата мала, често је могућа успешна криптоанализа. На пример, ако се у шифрату наиђе на $XMOX\ XMB$, које речи могу ово да буду? (that the, onto one). Која обична реч даје шифрат $FQJFUQ$? (people).

Задатак за криптоанализу: GU P IPY AKJW YKN CJJH HPOJ RGNE EGW OKIHPYGKYW HJVEPHW GN GW DJOPZWJ EJ EJPVW P AGUUJVJYN AVZIIJV MJN EGI WNJH NK NEJ IZWGO REGOE EJ EJPVW EKRJSJV IJPWZVJA KV UPV PRPB. (Упутство: обратите пажњу на другу и последњу реч).

21.1 Вижнерова шифра

Подсетимо се Цезарове шифре у којој се свако слово замењује словом које се налази три места иза њега у абецеди: $A- > D$, $B- > E$,..., $Z- > C$. Ово је једноставни специјални случај претходно разматраног система. Ако је могуће померање у абецеди за произвољан број позиција, онда је то *моноалфабетска шифра транслације*. Могу се циклички смењивати неколико моноалфабетских шифара транслације. Такав систем зове се *Вижнерова шифра* (Vigener). Кључ је нека реч, на пример TIN. Слово T је 19-то у абецеди, I је седмо, а N је 13-то. Према томе, транслираћемо прво слово за 19, друго за 7, треће за 13, четврто за 19, итд. Могли бисмо да кажемо да је ово троалфабетна шифра транслације. Пре него што су постојали рачунари, људи су користили Вижнеров квадрат на слици. Да би се шифровала реч $CRYPTO$ кључем TIN, најпре се шифрује слово C словом T. У квадрату се прочита слово на пресеку врсте C и колоне T (или обрнуто). Резултат шифровања је $VZLIBB$. Приметимо да се слово B два пута појављује у шифрату. Ако прималац има исти кључ, онда он може да направи исту табелу и дешифровање је лако.

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
BCDEFGHIJKLMNOPQRSTUVWXYZA
CDEFGHIJKLMNOPQRSTUVWXYZAB
DEFGHIJKLMNOPQRSTUVWXYZABC
```

```
EFGHIJKLMNOPQRSTUVWXYZABCD
FGHIJKLMNOPQRSTUVWXYZABCDE
GHIJKLMNOPQRSTUVWXYZABCDEF
HIJKLMNOPQRSTUVWXYZABCDEFG
IJKLMNOPQRSTUVWXYZABCDEFGH
JKLMNOPQRSTUVWXYZABCDEFGHI
KLMNOPQRSTUVWXYZABCDEFGHIJ
LMNOPQRSTUVWXYZABCDEFGHIJK
MNOPQRSTUVWXYZABCDEFGHIJKL
NOPQRSTUVWXYZABCDEFGHIJKLM
OPQRSTUVWXYZABCDEFGHIJKLMN
PQRSTUVWXYZABCDEFGHIJKLMNO
QRSTUVWXYZABCDEFGHIJKLMNOP
RSTUVWXYZABCDEFGHIJKLMNOPQ
STUVWXYZABCDEFGHIJKLMNOPQR
TUVWXYZABCDEFGHIJKLMNOPQRS
UVWXYZABCDEFGHIJKLMNOPQRST
VWXYZABCDEFGHIJKLMNOPQRSTU
WXYZABCDEFGHIJKLMNOPQRSTUV
XYZABCDEFGHIJKLMNOPQRSTUVW
YZABCDEFGHIJKLMNOPQRSTUVWX
ZABCDEFGHIJKLMNOPQRSTUVWXY
```

Претпоставимо да је примљен следећи шифрат, добијен применом Вижнерове шифре.

```
wzggqbuawq pvhveirrbv nysttaknke nxosavvwfw frvxqumhuw
wqgwtgziih locgpnhjmn nmtzqboavv abcuawohbv rjTAMPOvkl
gpigfsmfmw vnniyhzyrv qkkiqywweh vjrjwgWEWG Zhcxucakep
wpsnjhvama hkmehnhuww vtzguwaclz stsvfxlplz muywzygagk
aofkioblwi argtvrgzit xeofswcrqb tllcmiabfk ttbwbfenvz
snlytxahuw vgtzstghut vrzwrcglpr ariltwxwTA MPOtgvwlqh
vkhkynwpmp vmwgbjxqnb tnuxhkwasa gvbwbntswm pwfdmhxnce
zinbdsqarv aihojmneqo alfwmpomqd qgmkuwvgfg husrfaqggg
vavwzyahgg wbrgjjbake axkgovnkww kdwiwhdnbo aumggbgbmv
exaoogypWE WGZvgymfrf gglbcuaq
```

Како се може одредити дужина кључа? Постоје два метода да се то уради. Први је открио Казиски (Kasiski, 19. век), а други Фридман (Friedman, 20.век).

21.1.1 Тест Казиског

Посматрајмо чести триграм као што је THE и претпоставимо да је кључ дужине пет слова. Ако триграм THE почиње на позицијама n и m у шифрату, а $m \not\equiv n \pmod 5$, онда ће они бити различито шифровани. Ако је пак $m = n \pmod 5$, онда ће триграми бити шифровани са иста три слова

кључа. На пример, нека је кључ $VOICE$. Отворени текст $THEONETHE$ даје шифрат $OVMQRZHPG$, а отворени текст $THEBOTHE$ даје шифрат $OVMDSOVM$. Наравно, исто се дешава за било који чест триграм, као што су AND, ING и сл. За било који пар идентичних триграма у отвореном тексту очекује се да ће разлика њихових позиција бити дељива са пет у просеку сваки пети пут, када ће они бити идентично шифровани. Са довољно шифрата на располагању, можемо да тражимо поновљене триграме и израчунавамо размаке између њих. Размаци би требали да обично буду дељиви са пет.

У горњем шифрату примећују се понављања $WEWWGZ$ и $TAMPO$ на растојањима $322 = 2 \cdot 7 \cdot 23$, односно $196 = 2^2 \cdot 7^2$. Исто тако, понављања $HUWW$ и $UWVG$ су на размацима $119 = 7 \cdot 17$, односно $126 = 2 \cdot 3^2 \cdot 7$. На основу тога може се претпоставити да је дужина кључа 7. При томе морамо да будемо опрезни, јер могу да постоје случајна понављања, као што су две копије AVV на размаку 43. Према томе, ако напишемо програм који проналази nzd свих размака, он ће дати резултат 1.

21.1.2 Фридманов тест

Фридманов тест омогућује да се добије оцена дужине кључа. Приметимо да ако се за слова у шифрату добијеном моноалфабетском шифром транслације направи хистограм, добиће се исти хистограм (истина испретуран) као и за отворени текст.

Сортирани проценти појављивања слова (у енглеском тексту) могли би да буду $(12.31, 9.59, 8.05, 7.94, \ldots, 0.20, 0.20, 0.10, 0.09)$ за $ETAO \ldots QXJZ$. Ако се користи двоалфабетска шифра транслације, онда је, на пример, учестаност слова A у шифрату једнака просечној учестаности два слова која се замењују са A. Сортиране учестаности могле би да буду

$$(8.09, 7.98, 7.565, 7.295, \ldots 1.115, 1.04, 0.985, 0.31).$$

Ако се користи 10-алфабетска шифра транслације, онда би учестаност неког фиксираног слова у шифрату једнака просечној учестаности десет слова која се замењују тим словом. Сортиране учестаности могле би да буду $(4.921, 4.663, 4.611, 4.589, \ldots, 3.284, 3.069, 3.064, 2.475)$. Приметимо да ако посматрамо учестаности свих слова у шифрату, онда је њихов просек $1/26$, независно од броја алфабета. Међутим, што је већи број алфабета, мања је варијанса. Према томе, за оцену дужине кључа може се искористити варијанса вектора учестаности појављивања слова у шифрату.

За ову сврху искористићемо управо статистичку варијансу. Ако се изаберу два слова иза текста (не обавезно суседна) колика је вероватноћа да та слова буду једнака? Претпоставимо да имамо текст (отворени текст или шифрат) дужине n, при чему се слово A појављује n_0 пута, слово B n_1 пута, слово Z n_{25} пута, $\sum n_i = n$. Број парова слова A је $n_0(n_0 - 1)/2$, итд, па је укупан број парова од два иста слова једнак $\sum n_i(n_i - 1)$. Укупан број парова слова је $n(n-1)/2$. Дакле, вероватноћа да су два случајно изабрана слова једнака

је

$$I = \frac{\sum_{i=0}^{25} n_i(n_i-1)/2}{n(n-1)/2} = \frac{\sum_{i=0}^{25} n_i(n_i-1)}{n(n-1)}.$$

Величина I је узорачки *индекс коинциденције* (IOC). Колики је очекивани IOC за обични енглески текст? Нека је p_0 вероватноћа слова A, $p_0 \approx 0.0805$, итд. Вероватноћа да оба слова у пару буду A је p_0^2. Вероватноћа да оба слова у пару буду Z је p_{25}^2. Према томе, вероватоћа да два случајно изабрана слова у енглеском тексту буду једнака је $\sum p_i^2 \approx 0.065$. За случајни текст са једнаким вероватноћама појављивања слова је $p_0 = p_1 = \cdots = p_{25} = 1/26$ и $\sum p_i^2 = 1/26 \approx 0.038$. Приметимо да се овакав текст добија Вижнеровом шифром са кључем бесконачне дужине ако је кључ случајан. За моноалфабетску шифру замене очекивани IOC је приближно 0.065.

Да бисмо ствари поједноставили, размотримо једноставнији пример од енглеског текста. Нека се у језику користе само слова $\alpha, \beta, \gamma, \delta$ са вероватноћама редом $0.4, 0.3, 0.2, 0.1$. Очекивани IOC је 0.3. Ако се неки текст шифрује Вижнеровом шифром са кључем $\beta\gamma$ (тј. померање за један, па два, па један, ...), онда су вероватноће појављивања слова $\alpha, \beta, \gamma, \delta$ у шифрату редом $1/2(.1)+1/2(.2) = .15$, $1/2(.4)+1/2(.1) = .25$, $1/2(.3)+1/2(.4) = .35$ и $1/2(.2)+1/2(.3) = .25$. Очекиввани IOC је тада 0.27. Приметимо да узорачки IOC опада са дужином кључа. Узорачки IOC може се искористити за доношење одлуке да ли је шифрат добијен моноалфабетском или полиалфабетском шифром.

Назад на енглески текст. Претпоставимо да имамо шифрат дужине n и кључ дужине k, који треба да одредимо. Због једноставности претпоставимо да $k|n$ и да су сва слова кључа различита. Шифрат се може исписати у облику таблице (наравно, k се за сада не зна)

$$\begin{array}{cccc} c_1 & c_2 & c_3 & c_4 \\ c_{k+1} & c_{k+2} & c_{k+3} & c_{k+4} \\ \vdots & & & \\ & & & c_n \end{array}$$

Број врста је n/k. Свака колона добија се моноалфабетском транслацијом. Вероватноћа да два слова изабрана из исте колоне буду једнака је ≈ 0.065. Вероватноћа да два слова изабрана из различитих колона буду једнака је ≈ 0.038.

Колики је очекивана вредност узорачког IOC ? Број парова у истој колони је $n((n/k)-1)/2 = (n(n-k)/(2k)$. Број парова из различитих колона је $n(N-(n/k))/2 = (n^2(K-1)/(2k)$. Очекивани број парова са једним истим словом је

$$A = 0.0065\left(\frac{n(n-k)}{2k}\right)0.0038\left(\frac{n^2(k-1)}{2k}\right).$$

Вероватноћа да се било који пар састоји од два иста слова је

$$\frac{A}{n(n-1)/2} = \frac{1}{k(n-1)}[0.027n + k(0.038n - 0.065)].$$

То је очекивана вредност IOC . Ову вредност изједначујемо са узорачким IOC и решавамо по k.

$$k \approx \frac{0.027n}{(n-1)I - 0.038n + 0.065}, \text{ где је } I = \sum_{i=0}^{25} \frac{n_i(n_i-1)}{n(n-1)}$$

У нашем примеру је $I \approx 0.04498$, $k \approx 3.844$. Добијена оцена дужине кључа је 3.844 (стварна дужина кључа је 7, тако да је ова оцена доста лоша).

21.1.3 Одређивање кључа

С обзиром на резултате тестова Казиског и Фридмана, претпоставимо да је дужина кључа 7 (а не 14, што би се добило ако би понављање $HUWW$ било случајно). Сада треба да одредимо кључ, односно колика је транслација сваке колоне. Можемо да направимо програм за одређивање хистограма учестаности појављивања слова шифрата на позицијама $1, 8, 15, 22, \ldots$ (та слова су у првој колони ако се шифрат препише у таблицу са врстом дужине 7). Добија се да су учестаности слова $A - Z$

$$[10, 0, 0, 1, 1, 3, 7, 0, 0, 5, 7, 3, 2, 2, 0, 0, 1, 0, 4, 1, 2, 3, 10, 0, 1, 6].$$

Знамо да су се у отвореном тексту најчешће појављују слова E, T, A. Размаци између њих у алфабету су $A - 4 - E - 15 - T - 7 - A$. За кључ дужине 7 и шифрат дужине 478 може се претпоставити да ће се свако од ових слова појавити бар три пута у сваком од седам скупова (хистограма). Дакле, тражимо у хистограму три слова, таква да се свако појављује бар три пута, тако да су на (цикличком) растојању $4 - 15 - 7$. Ако има више оваквих тројки, сабирамо учестаности појављивања та три слова, па највећи приоритет дајемо транслацији којој одговара највећи збир.

За горњи хистограм примећујемо да транслацији за шест одговара сума $7 + 7 + 6 = 20$, а да транслацији за 18 одговара сума 17. На сличан начин праве се хистограми за другу, трећу, ..., седму колону шифрата препакованог у таблицу $(n/7) \times 7$. За другу колону једина могућа транслација је за 4. За трећу колону померај 0 има збир 11, а померај 2 суму 17. За четврту колону померај 7 има збир 14, а померај 19 суму 20. За пету колону померај 8 има збир 16, а померај 11 суму 12. За шесту колону померај 2 има збир 13, а померај 14 суму 22. За седму колону померај 1 има збир 17, а померај 13 суму 21. Дакле, најпре покушавамо са померајима $[6, 4, 2, 19, 8, 14, 13]$. Можемо да извршимо дешифровање користећи ово као кључ; првих седам слова отвореног текста су тада $QVENINH$. Овде Q изгледа погрешно. Друга могућност за померај у првој колони била је 18. Са овим померајем добијамо резултат дешифровања $EVEN\ IN\ HIS\ OWN\ TIME\ldots$.

21.2 Криптоанализа модерних проточних шифара

21.2.1 Генератор случајних бројева b/p

Ово је пример генератора које се више не користи. Нека је p велики прост број за који 2 генерише $\mathbf{F}_p^*$. Бира се b, $1 \leq b < p$, па је b/p кључ. Израчунава се "децимални" развој броја $b/p = 0.k_1k_2k_3k_4\ldots$ у систему са основом 2 (k_i је бит). Тада је $k_1k_2k_3k_4\ldots$ низ кључа. Дужина периода тог низа је $p-1$

Пример. Нека је $p = 11$. Пошто је $2^2 \neq 1$ и $2^5 \neq 1$ у $\mathbf{F}_{11}$, видимо да 2 генерише $\mathbf{F}_{11}^*$. Нека је кључ 5/11. У систему са основом 2 то је 101/1011. "Децимални" развој:

```
          0.01110...
        -----------
1011   | 101.00000
          10 11
         ------
          10 010
           1 011
          ------
             1110
             1011
             ----
              110 итд.
```

Уместо овог изабраћемо декадни пример, због лакшег рачунања. Нека је p прост број за који 10 генерише $\mathbf{F}_p^*$. Бирамо $1 \leq b < p$. Исписујемо децимални развој $b/p = 0.n_1n_2n_3\ldots$ (где је $0 \leq n_i \leq 9$). Тада је $n_1n_2n_3\ldots$ низ кључа. Пошто је 10 генератор, дужина периода је $p-1$, дакле велика.

Нека је на пример $b = 12$, $p = 17$, $b/p = .70588235294117647058\ldots$, $n_1 = 7$, $n_2 = 0$, итд. Нека је отворени текст $MONTEREY$. Замењујући свако слово са две декадне цифре, добијамо $1214131904170424 = p_1p_2p_3\ldots$, $0 \leq p_i \leq 9$, па је $p_1 = 1$, $p_2 = 2$, $p_3 = 1$, итд.

Поступак шифровања је $c_i = p_i + n_i \pmod{10}$ а шифрат је $c_1c_2c_3\ldots$. Дешифровање се врши одузимањем $p_i = c_i - n_i \pmod{10}$.

```
Пример          (шифровање)                (дешифровање)


             OT 12141319                        ST 82629544
    + niz kljuca 70588325              - niz kljuca 70588235
     --------------------               --------------------
             ST 82629544                        OT 12141319
                      (sabiranje je bez prenosa)  M O N T
```

Приказаћемо сада напад са познатим отвореним текстом. Нека број p има l цифара и нека Цица има шифрат и првих $2l+1$ цифара (бита) отвореног текста. Она може да одреди b и p помоћу *верижних разломака*.

Верижни разломак је разломак облика

$$a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{\ddots + \frac{1}{a_n}}}}$$

где су a_i цели ненегативни бројеви, $a_i > 0$ за $i > 1$. Уобичајено је да се за верижни разломак користи запис $[a_1, a_2, \ldots, a_n]$. Сваком рационалном броју одговара коначан верижни разломак, који се добија применом Еуклидовог алгоритма. На пример $27/8 = 3+1/(2+1/(1+1/(2))) = [3, 2, 1, 2]$. Ирационалним бројевима једнозначно одговарају бесконачни верижни развоји облика $\alpha = a_1 + 1/(a_2 + 1/(a_3 + \ldots = [a1, a2, \ldots]$, односно $\alpha = \lim_{n\to\infty}[a1, a2, \ldots, a_n]$. Рационални бројеви $a_1, a_1+1/a_2, a_1+1/(a_2+1/a_3)$, (односно $[a_1], [a_1, a_2], [a_1, a_2, a_3]$) су *конвергенти* (парцијални разломци) броја α.

Конвергенти су веома добре рационалне апроксимације α. Показује се да ако је α ирационалан и $|\alpha - a/b| < \frac{1}{2b^2}$, где су a, b цели , $b \geq 1$, онда је a/b конвергент α. На пример, $22/7$ је чувена добра апроксимација π. Тај разломак јесте конвергент $\pi = 3 + 1/(7 + 1/(15 + \ldots$. Прва три конвергента π су 3, $3 + 1/7 = 22/7 = 3.14\ldots$, и $3 + 1/(7 + 1/15) = 333/106 = 3.1415\ldots$.

Одређивање верижног развоја α. Нека је $\lfloor\alpha\rfloor$ највећи цели број $\leq \alpha$. На пример, $\lfloor\pi\rfloor = 3$, $\lfloor 5\rfloor = 5$, $\lfloor -1.5\rfloor = -2$. Нека је $\alpha_1 = \alpha$ и $a_1 = \lfloor\alpha_1\rfloor$. Нека је $\alpha_2 = 1/(\alpha_1 - a_1)$ и $a_2 = \lfloor\alpha_2\rfloor$. Нека је $\alpha_3 = 1/(\alpha_2 - a_2)$ и $a_3 = \lfloor\alpha_3\rfloor$, итд.

Интересантан пример је $e = 2.71828\ldots$. Тада је $\alpha_1 = e$, $a_1 = \lfloor\alpha_1\rfloor = 2$. $\alpha_2 = 1/(e - 2) = 1.39\ldots$, $a_2 = \lfloor 1.39\rfloor = 1$. $\alpha_3 = 1/(1.39\ldots - 1) = 2.55\ldots$, $a_3 = \lfloor 2.55\ldots\rfloor = 2$. $\alpha_4 = 1/(2.55\ldots - 2) = 1.82\ldots$, $a_4 = \lfloor 1.82\ldots\rfloor = 1$. $\alpha_5 = 1/(1.82\ldots - 1) = 1.22\ldots$, $a_5 = \lfloor 1.22\ldots\rfloor = 1$. $\alpha_6 = 1/(1.22\ldots - 1) = 4.54\ldots$ и $a_6 = \lfloor 4.54\ldots\rfloor = 4$. Тачан верижни развој e је $[2, 1, 2, 1, 1, 4, 1, 1, 6, 1, 1, 8, 1, 1, 10, \ldots]$.

Вратимо се на криптоанализу. Нека Цица има шифрат и првих $3n$ "бита" отвореног текста. На основу тога она може да добије првих $3n$ "бита" низа кључа. Она одређује конвергент који се слаже са првих $2n + 1$ "бита" низа кључа и проверава да ли се и остатак слаже. Ако је $n > \log p$, онда се низ кључа успешно одређује.

У наредном примеру нека се зна првих 18 цифара ОТ, односно $n = 6$.

СТ	5309573992060	746098818740243
ОТ	0200170405201	11704
низ кључа	5109403597869	63905

Одређујемо верижни разломак за 0.510940359786963905; добијамо

$$[0, 1, 1, 22, 2, 1, 5, 1, 1, 3, 2, 4, 308404783, 1, 2, 1, 2\ldots].$$

Конвергенти су 0, 1, 1/2, 23/45, 47/92, ... Конвергент $[0, 1, 1, ...1, 3, 2] = 6982/13665 = 0.51094035858\ldots$ није тачан, али следећи $[0, 1, 1, ...1, 3, 2, 4] = 30987/60647 = .5109403597869630958815769987\ldots$ јесте. То је први конвергент који се слаже се првих 13 цифара низа кључа, а поклапа се и са наредних 5 цифара, па смо сигурни да је добар.

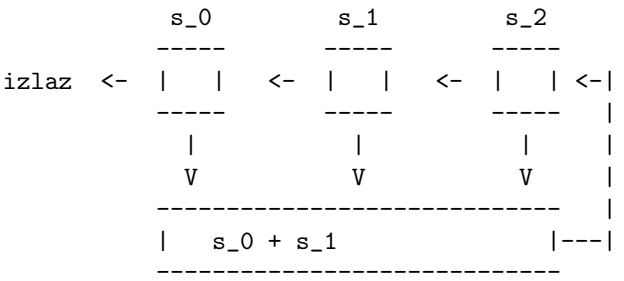
```
        ST 5309573992060 746098818740243
        OT 0200170405201 117040003081306 = CAREFULREADING
niz kljuca 5109403597869 639058815769947
```

Уствари, може се искористити било који низ узастопних бита, не само почетних — поступак је само мало компликованији. Потребно је само да буде $n \geq p$.

21.3 Померачки регистар са линеарном повратном спрегом

Наредни генератор случајних бројева је врло ефикасан, иако, као што ћемо видети, није сигуран. Зове се *линеарни померачки регистар* (ЛПР, енглески linear shift register, LSR) и био је популаран 1970-тих година. И даље се користи за хеш функције и контролне суме, а нелинеарни померачки регистри се и даље користе као генератори случајних бројева.

На слици је приказан пример једног ЛПР.



Излаз ЛПР је низ кључа. Нека је почетно стање ЛПР $(s_0, s_1, s_2) = (0, 1, 1)$. Рад ЛПР приказан је на слици 1 на следећој страни. Ћелије померачког регистра представљене су квадратима, а стрелице наниже воде само из квадрата чији садржај улази у суму. На дну слике види се да је ЛПР дошао у почетно стање, што значи да ће низ кључа почети да се периодично понавља.

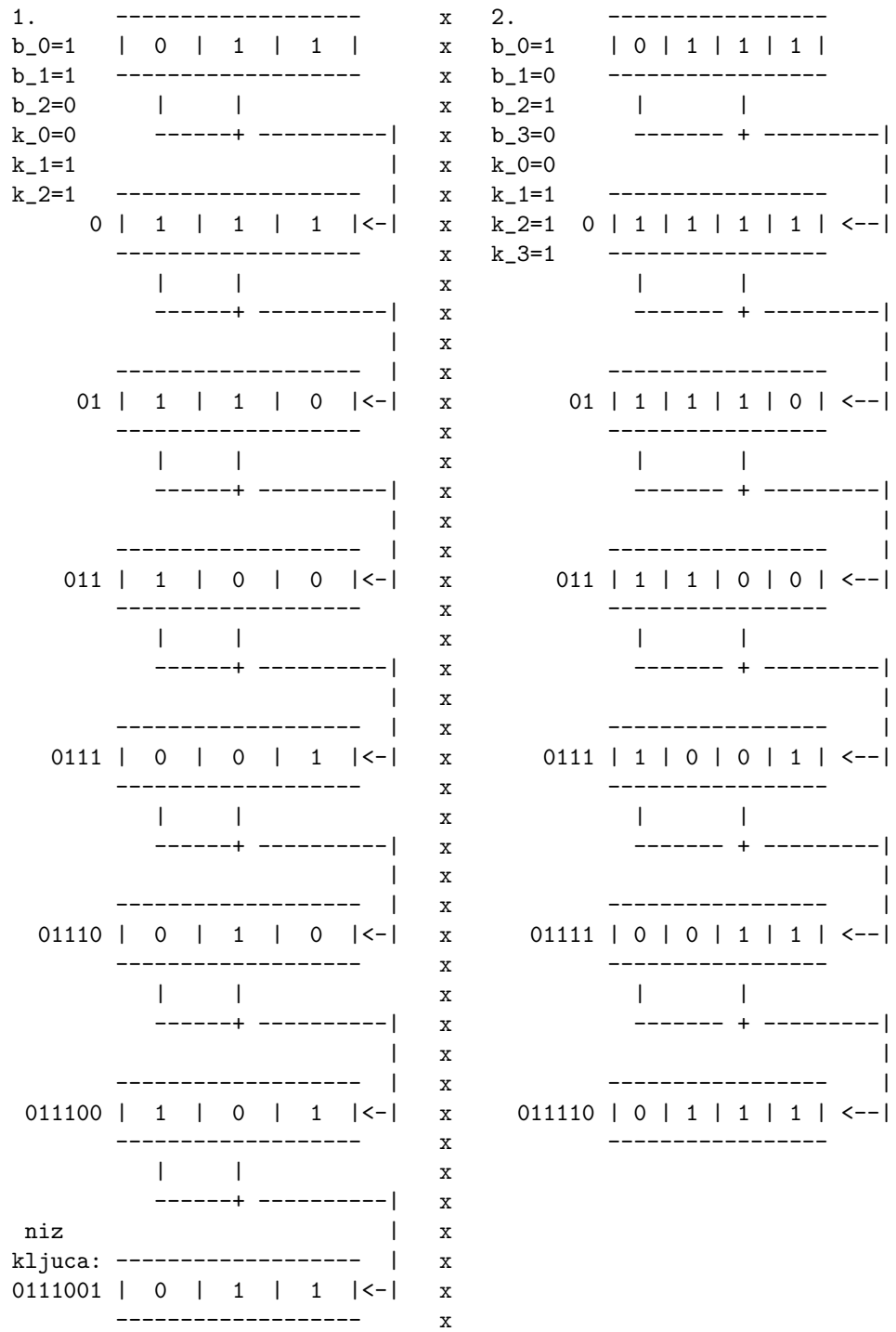
Ово је био ЛПР дужине три. За ЛПР дужине n (дужина 32 је уобичајена) нека су кључ, односно почетно стање $2n$ бита означених са $b_0, \ldots, b_{n-1}, k_0, \ldots, k_{n-1} \in \{0, 1\}$, при чему је $b_0 \neq 0$ и нису сви k_i једнаки 0.

Прва n-торка $(s_0, \ldots, s_{n-1})$ зове се почетно стање и једнака је $(k_0, \ldots, k_{n-1})$. У примеру је било $(k_0, k_1, k_2) = (0, 1, 1)$. Функција која даје последњи бит наредног стања је $f(s_0, \ldots, s_{n-1}) = b_0 s_0 + b_0 s_0 + \cdots + b_{n-1} s_{n-1}$. У примеру је $f = s_0 + s_1 = 1s_0 + 1s_1 + 0s_2$, па је $(b_0, b_1, b_2) = (1, 1, 0)$. Стање у неком тренутку је $(s_0, \ldots, s_{n-1})$; у наредном стању је s_{i+1} исто што и s_i у претходном стању, изузев s_{n-1}, које је једнако излазу функције f.

За фиксирано f (тј. фиксирану n-торку b_i) постоји 2^n могућих почетних стања за ЛПР дужине n. За два стања кажемо да су у истој орбити ако се из једног после неколико корака долази у друго стање. Стање $(0, 0, \ldots, 0)$ има орбиту величине 1. У примеру су свих осталих 7 стања у другој орбити.

Према томе, дужина периода низа кључа је $7 = 2^3 - 1$ (што је најбоља могућа вредност).

Размотримо ЛПР дужине 4 са $(b_0, b_1, b_2, b_3) = (1, 0, 1, 0)$. Одредимо орбиту овог стања, видети слику 2. Величина орбите је 6, па је дужина периода низа кључа 6. Предност би ипак имао низ кључа са периодом дужине $2^4 - 1 = 15$. ЛПР дужине 4 са $(b_0, b_1, b_2, b_3) = (1, 0, 0, 1)$ има орбите величине 15 и 1.


1.
b_0=1
b_1=1
b_2=0
k_0=0
k_1=1
k_2=1
0
01
011
0111
01110
011100
niz
kljuca:
0111001
2.
b_0=1
b_1=0
b_2=1
b_3=0
k_0=0
k_1=1
k_2=1
k_3=1
0
01
011
0111
01111
011110

Размотримо ЛПР дужине 5 са $(b_0, b_1, b_2, b_3, b_4) = (1, 1, 1, 0, 1)$. Одредимо величину орбите стања $(1, 1, 0, 0, 1)$. Видимо да је у овом случају величина орбите $31 = 2^5 - 1$. Излазни низ кључа је 1100110111110100010010101100001. Приметимо да се у овом низу појављују сва стања изузев $(0, 0, 0, 0, 0)$. То значи да се све могуће петорке бита (различите од 00000) појављују тачно једном у овом низу кључа.

Како се може установити када постоје само две орбите величина 1 и 2^n-1? Нека је $g(x) = b_0+b_1x^1+b_2x^2+\ldots+b_{n-1}x^{n-1}+x^n$. Максимална орбита (величине $2^n - 1$) добија се ако и само ако је полином $g(x)$ *примитиван* по модулу два (тј. над пољем $\mathbf{F}_2$). Кажемо да је полином $g(x)$ примитиван ако је несводљив и x је генератор $\mathbf{F}^*_{2^n} = \mathbf{F}_2[x]/(g(x))^*$. Ако је број $2^n - 1$ прост, онда је довољно да $g(x)$ буде несводљив.

Према томе, n-торку $(b_0, \ldots, b_{n-1})$ бирамо тако да $g(x)$ буде несводљив и да x генерише $\mathbf{F}^*_{2^n}$. У том случају низ кључа има највећи могући период. За ЛПР дужине 32 постоји око 67 милиона различитих несводљивих полинома.

Приметимо да је у првом примеру било $(b_0, b_1, b_2) = (1, 1, 0)$ што одговара полиному $1 + x + x^3$, који јесте несводљив и примитиван, па има орбиту максималне дужине. У другом примеру је било $(b_0, b_1, b_2, b_3) = (1, 0, 1, 0)$ што одговара полиному $1 + x^2 + x^4 = (1 + x + x^2)^2$. Овај полином није несводљив, и нисмо добили орбиту максималне дужине. У трећем примеру је било $(b_0, b_1, b_2, b_3, b_4) = (1, 1, 1, 0, 1)$ што одговара полиному $1+x+x^2+x^4+x^5$ који јест несводљив (и примитиван), орбита је била максималне дужине.

Као пример, шифрујмо ОТ Hi, низом кључа добијеним од ЛПР са

$$(b_0, \ldots, b_4), (k_0, \ldots, k_4) = (1, 1, 1, 0, 1), (1, 1, 0, 0, 1).$$

Добијамо низ кључа из трећег примера. Отворени текст је $Hi = 0100100001101001$ (ASCII).

ОТ	0100100001101001
низ кључа	1100110111110100
СТ	1000010110011101

Приметимо да је истовремено и СТ + низ кључа = ОТ СТ + ОТ = низ кључа. То је згодна особина сабирања по модулу два.

Како се ова шифра може разбити? Претпоставимо да је пресретнут шифрат и првих $2n$ бита отвореног текста (ради се дакле о нападу са познатим отвореним текстом). Затим добијамо првих $2n$ бита низа кључа $k_0, k_1, \ldots, k_{2n-1}$. После тога може се реконструисати цели низ кључа. Претпоставимо да прави корисник користи функцију $f = s_0+s_1+s_2+s_4$ иако ми то нећемо да користимо. Ми знамо $k_0k_1k_2k_3k_4k_5k_6k_7k_8k_9 = 1100110111$.

знамо	не знамо	можемо да пишемо
k_5	$= k_0 + k_1 + k_2 + k_3 + k_4$	$= b_0k_0 + b_1k_1 + b_2k_2 + b_3k_3 + b_4k_4$
k_6	$= k_1 + k_2 + k_3 + k_4 + k_5$	$= b_0k_1 + b_1k_2 + b_2k_3 + b_3k_4 + b_4k_5$
k_7	$= k_2 + k_3 + k_4 + k_5 + k_6$	$= b_0k_2 + b_1k_3 + b_2k_4 + b_3k_5 + b_4k_6$
k_8	$= k_3 + k_4 + k_5 + k_6 + k_7$	$= b_0k_3 + b_1k_4 + b_2k_5 + b_3k_6 + b_4k_7$
k_9	$= k_4 + k_5 + k_6 + k_7 + k_8$	$= b_0k_4 + b_1k_5 + b_2k_6 + b_3k_7 + b_4k_8$

У општем случају је $k_{t+n} = b_0k_t + b_1k_{t+1} + \cdots + b_{n-1}k_{t+n-1}$. Имамо систем од n линеарних једначина (над $\mathbf{F}_2$) са n непознатих b_i, $i = 0, 1, \ldots, n-1$. Претходни систем може да се напише у матричном облику.

$$\begin{bmatrix} k_0 & k_1 & k_2 & k_3 & k_4 \\ k_1 & k_2 & k_3 & k_4 & k_5 \\ k_2 & k_3 & k_4 & k_5 & k_6 \\ k_3 & k_4 & k_5 & k_6 & k_7 \\ k_4 & k_5 & k_6 & k_7 & k_8 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} = \begin{bmatrix} k_5 \\ k_6 \\ k_7 \\ k_8 \\ k_9 \end{bmatrix}$$

После замене вредности k_i добија се систем

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Решавањем ове матричне једначине на пољем $\mathbf{F}_2$ добија се $b_0b_1b_2b_3b_4 = 11101$.

Сада, кад знамо b_i, k_i, $i = 0, 1, \ldots, n-1$, можемо сами да израчунамо све чланове низа кључа. Крај примера.

У општем случају имамо систем једначина

$$\begin{bmatrix} k_0 & k_1 & k_2 & \cdots & k_{n-1} \\ k_1 & k_2 & k_3 & \cdots & k_n \\ k_2 & k_3 & k_4 & \cdots & k_{n+1} \\ & & \vdots & & \\ k_{n-1} & k_n & k_{n+1} & \cdots & k_{2n-2} \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_{n-1} \end{bmatrix} = \begin{bmatrix} k_n \\ k_{n+1} \\ k_{n+2} \\ \vdots \\ k_{2n-1} \end{bmatrix}$$

Пример. Знамо да Алиса и Бобан користе ЛПР дужине шест и да порука почиње се To (ASCII), а пресрели смо шифрат 1011 0000 1101 1000 0010 0111 1100 0011.

```
      OT  10110000110110000010011111000011
      CT  0101010001101111
            T       o
          ----------------
низ кључа 1110010010110111
```

Добијамо систем

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Решавањем добијамо $b_0b_1b_2b_3b_4b_5 = 110000$. Пошто знамо све b_i и првих шест чланова низа кључа, можемо да реконструишемо цео низ кључа и дешифрујемо остатак шифрата. Добијамо да је отворени текст $ToAl$.

Шта ако бисмо знали само да је прво слово отвореног текста T? Тада бисмо могли да преостала четири бита претпоставимо на сва могуће начине да бисмо имали укупно 12 чланова низа кључа. Приметимо да ће само неке од тих претпоставки као резултат дати инвертибилну матрицу над $\mathbf{F}_2$. Од таквих претпоставки одабраћемо ону која даје смислени текст као резултат дешифровања. Крај примера.

Понекад се као генератор случајних бројева користи нелинеарни померачки регистар, на пример $f = s_0s_2s_7 + s_1 + s_2s_4 + s_4$.

22 Линеарна и диференцијала криптоанализа

Ова два важна напада биће илустрована нападом на верзију SAES од само једне рунде, у даљем тексту 1SAES .

22.1 1SAES

Подсетимо се да је табела S нелинеарно пресликавање $S : \{0,1\} \to \{0,1\}$. Тако је нпр. $S(b_0, b_1, b_2, b_3) \neq b_0 + b_1, b_1 + b_2 + b_3, b_0, b_1 + b_2$. Нека је задат кључ $K_0 = k_0k_1k_2k_3k_4k_5k_6k_7k_8k_9k_{10}k_{11}k_{12}k_{13}k_{14}k_{15}$

$$
\begin{array}{cccccccccc}
 & k_8 & k_9 & k_{10} & k_{11} & k_{12} & k_{13} & k_{14} & k_{15} \\
 & & & & \swarrow & \searrow & & & \\
 & k_{12} & k_{13} & k_{14} & k_{15} & k_8 & k_9 & k_{10} & k_{11} \\
 & & S\downarrow & & & & & S\downarrow & \\
 & n_0 & n_1 & n_2 & n_3 & n_4 & n_5 & n_6 & n_7 \\
 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\oplus & k_0 & k_1 & k_2 & k_3 & k_4 & k_5 & k_6 & k_7 \\
= & k_{16} & k_{17} & k_{18} & k_{19} & k_{20} & k_{21} & k_{22} & k_{23} \\
\oplus & k_8 & k_9 & k_{10} & k_{11} & k_{12} & k_{13} & k_{14} & k_{15} \\
= & k_{24} & k_{25} & k_{26} & k_{27} & k_{28} & k_{29} & k_{30} & k_{31}
\end{array}
$$

Добија се $K_1 = k_{16}k_{17}k_{18}k_{19}k_{20}k_{21}k_{22}k_{23}k_{24}k_{25}k_{26}k_{27}k_{28}k_{29}k_{30}k_{31}$. Касније ћемо користити једнакости $k_{16} + k_{24} = k_8, \ldots k_{23} + k_{15} = k_{31}$.

Подсећање на поступак шифровања:

$$
\begin{array}{|c|c|}\hline p_0p_1p_2p_3 & p_8p_9p_{10}p_{11} \\ \hline p_4p_5p_6p_7 & p_{12}p_{13}p_{14}p_{15} \\ \hline \end{array}
\overset{A_{K_0}}{\to}
\begin{array}{|c|c|}\hline p_0 \oplus k_0\ p_1 \oplus k_1\ p_2 \oplus k_2\ p_3 \oplus k_3 & p_8 \oplus k_8\ p_9 \oplus k_9\ p_{10} \oplus k_{10}\ p_{11} \oplus k_{11} \\ \hline p_4 \oplus k_4\ p_5 \oplus k_5\ p_6 \oplus k_6\ p_7 \oplus k_7 & p_{12} \oplus k_{12}\ p_{13} \oplus k_{13}\ p_{14} \oplus k_{14}\ p_{15} \oplus k_{15} \\ \hline \end{array}
$$

Нека је $S(p_0p_1p_2p_3 + k_0k_1k_2k_3) = m_0m_1m_2m_3$, итд. После NS стање је

$$
\overset{NS}{\to}
\begin{array}{|c|c|}\hline m_0m_1m_2m_3 & m_8m_9m_{10}m_{11} \\ \hline m_4m_5m_6m_7 & m_{12}m_{13}m_{14}m_{15} \\ \hline \end{array}
\overset{SR}{\to}
\begin{array}{|c|c|}\hline m_0m_1m_2m_3 & m_8m_9m_{10}m_{11} \\ \hline m_{12}m_{13}m_{14}m_{15} & m_4m_5m_6m_7 \\ \hline \end{array}
$$

После MC стање је

$$\overset{MC}{\rightarrow}\begin{array}{|c|c|}\hline m_0+m_{14}\ m_1+m_{12}+m_{15}\ m_2+m_{12}+m_{13}\ m_3+m_{13} & m_6+m_8\ m_4+m_7+m_9\ m_4+m_5+m_{10}\ m_5+m_{11}\\ \hline m_2+m_{12}\ m_0+m_3+m_{13}\ m_0+m_1+m_{14}\ m_1+m_{15} & m_4+m_{10}\ m_5+m_8+m_{11}\ m_6+m_8+m_9\ m_7+m_9\\ \hline\end{array}$$

На крају се додаје K_1 и стање је

$$\overset{A_{K_1}}{\rightarrow}\begin{array}{|c|c|}\hline m_0+m_{14}+k_{16}\ldots m_3+m_{13}+k_{19} & m_6+m_8+k_{24}\ldots m_5+m_{11}+k_{27}\\ \hline m_2+m_{12}+k_{20}\ldots m_1+m_{15}+k_{23} & m_4+m_{10}+k_{28}\ldots m_7+m_9+k_{31}\\ \hline\end{array}=\begin{array}{|c|c|}\hline c_0c_1c_2c_3 & c_8c_9c_{10}c_{11}\\ \hline c_4c_5c_6c_7 & c_{12}c_{13}c_{14}c_{15}\\ \hline\end{array}$$

Додавањем на обе стране $k_{16}\ldots k_{31}$ добија се

$$\begin{array}{|c|c|}\hline m_0+m_{14}\ m_1+m_{12}+m_{15}\ m_2+m_{12}+m_{13}\ m_3+m_{13} & m_6+m_8\ m_4+m_7+m_9\ m_4+m_5+m_{10}\ m_5+m_{11}\\ \hline m_2+m_{12}\ m_0+m_3+m_{13}\ m_0+m_1+m_{14}\ m_1+m_{15} & m_4+m_{10}\ m_5+m_8+m_{11}\ m_6+m_8+m_9\ m_7+m_9\\ \hline\end{array}$$

$$=\begin{array}{|c|c|}\hline c_0+k_{16}\ c_1+k_{17}\ c_2+k_{18}\ c_3+k_{19} & c_8+k_{24}\ c_9+k_{25}\ c_{10}+k_{26}\ c_{11}+k_{27}\\ \hline c_4+k_{20}\ c_5+k_{21}\ c_6+k_{22}\ c_7+k_{23} & c_{12}+k_{28}\ c_{13}+k_{29}\ c_{14}+k_{30}\ c_{15}+k_{31}\\ \hline\end{array}$$

Подразумевани редослед ћелија у блоковима:

ћелија 1	ћелија 3
ћелија 2	ћелија 4

22.2 Линеарна криптоанализа

Линеарна криптоанализа је напад са познавањем отвореног текста. Пронашао га је Matsui 1992. године. За овај напад потребно је много парова OT/CT добијених датим кључем. Нека је блок OT $p_0\ldots p_{n-1}$, кључ $k_0\ldots k_{m-1}$, а одговарајући CT $c_0\ldots c_{n-1}$.

Претпоставимо да линеарна једначина

$$p_{\alpha_1}+p_{\alpha_2}+\cdots+p_{\alpha_a}+c_{\beta_1}+\cdots+c_{\beta_b}+k_{\gamma_1}+\cdots+k_{\gamma_g}=x$$

(где је $x\in\{0,1\}$, $1\le a,b\le n$, $1\le g\le m$) важи са вероватноћом $p>0.5$ на скупу свих парова OT/CT.

Вредност $x+p_{\alpha_1}+p_{\alpha_2}+\cdots+p_{\alpha_a}+c_{\beta_1}+\cdots+c_{\beta_b}$ се израчунава за све пресретнуте парове OT/CT. Ако међу вредностима има више нула, претпоставља се да је $k_{\gamma_1}+\cdots+k_{\gamma_g}=0$, а ако међу вредностима има више јединица, претпоставља се да је $k_{\gamma_1}+\cdots+k_{\gamma_g}=1$. На тај начин добија се линеарна веза између бита кључа. Циљ је пронаћи више оваквих веза. Кад би шифровање била линеарна операција, ове једначине би важиле са вероватноћом 1, па не би било потребно више парова OT/CT.

22.3 Линеарна криптоанализа 1SAES

Нека је $S(a_0a_1a_2a_3) = b_0b_1b_2b_3$. Тада се вредности b_i за разне a_i могу представити таблицом

a_0	a_1	a_2	a_3	b_0	b_1	b_2	b_3
0	0	0	0	1	0	0	1
0	0	0	1	0	1	0	0
0	0	1	0	1	0	1	0
0	0	1	1	1	0	1	1
0	1	0	0	1	1	0	1
0	1	0	1	0	0	0	1
0	1	1	0	1	0	0	0
0	1	1	1	0	1	0	1
1	0	0	0	0	1	1	0
1	0	0	1	0	0	1	0
1	0	1	0	0	0	0	0
1	0	1	1	0	0	1	1
1	1	0	0	1	1	0	0
1	1	0	1	1	1	1	0
1	1	1	0	1	1	1	1
1	1	1	1	0	1	1	1

Израчунавањем свих линеарних комбинација улазних бита a_i и излазних бита b_j долази се до линеарних веза које важе са вероватноћом већом од 0.5 (не постоји ни једна веза која важи са вероватноћом већом од 0.75).

$$
\begin{aligned}
a_3 + b_0 &= 1110111101011011 = 1, p = 12/16 = .75 && (1)\\
a_0 + a_1 + b_0 &= 1011010111111110 = 1, p = .75 && (2)\\
a_1 + b_1 &= 0100011010000000 = 0, p = .75 && (3)\\
a_0 + a_1 + a_2 + a_3 + b_1 &= = 0, p = .75 && (4)\\
a_1 + a_2 + b_0 + b_1 &= = 1, p = .75 && (5)\\
a_0 + b_0 + b_1 &= = 1, p = .75 && (6)
\end{aligned}
$$

22.4 Одређивање једначина

Да бисмо дошли до једначина облика

$$p_{\alpha_1} + p_{\alpha_2} + \cdots + p_{\alpha_a} + c_{\beta_1} + \cdots + c_{\beta_b} + k_{\gamma_1} + \cdots + k_{\gamma_g} = x$$

треба да елиминишемо бите m_i и n_i. Елиминисаћемо најпре бите n_i.

$$
\begin{aligned}
c_0 + k_{16} + c_8 + k_{24} = c_0 + c_8 + k_8 &= m_0 + m_{14} + m_6 + m_8 && (7)\\
c_1 + k_{17} + c_9 + k_{25} = c_1 + c_9 + k_9 &= m_1 + m_{12} + m_{15} + m_4 + m_7 + m_9 && (8)\\
c_2 + c_{10} + k_{10} &= m_2 + m_{12} + m_{13} + m_4 + m_5 + m_{10} && (9)\\
c_3 + c_{11} + k_{11} &= m_3 + m_{13} + m_5 + m_{11} && (10)\\
c_4 + c_{12} + k_{12} &= m_2 + m_{12} + m_4 + m_{10} && (11)\\
c_5 + c_{13} + k_{13} &= m_0 + m_3 + m_{13} + m_5 + m_8 + m_{11} && (12)\\
c_6 + c_{14} + k_{14} &= m_0 + m_1 + m_{14} + m_6 + m_8 + m_9 && (13)\\
c_7 + c_{15} + k_{15} &= m_1 + m_{15} + m_7 + m_9. && (14)
\end{aligned}
$$

Десне стране ових једначина зависе од све четири ћелије стања. Због тога њихово коришћење за комбиновање са једначинама као што су (1)-(6) доводи до једначина са вероватноћама блиским 0.5 (касније ћемо видети да постоји тенденција да вероватноће са којима важе једначине буду блиске 0.5).

Уместо тога, приметимо да су бити који се појављују на десној страни (7) подскуп бита који се појављују у (13). Слично, бити који се појављују на десној страни (10), (11), (14) су подскупови скупова бита који се појављују редом у (12), (9), (8). Због тога, ако саберемо једначине (7) и (13), (10) и (12), (11) и (9), односно (14) и (8), добијамо

$$\begin{aligned} c_0 + c_6 + c_8 + c_{14} + k_8 + k_{14} &= m_1 + m_9 \quad (15)\\ c_3 + c_5 + c_{11} + c_{13} + k_{11} + k_{13} &= m_0 + m_8 \quad (16)\\ c_2 + c_4 + c_{10} + c_{12} + k_{10} + k_{12} &= m_5 + m_{13} \quad (17)\\ c_1 + c_7 + c_9 + c_{15} + k_9 + k_{15} &= m_4 + m_{12}. \quad (18) \end{aligned}$$

Једначине (15)-(18) су увек тачне.

Посматрајмо једнакост (16). Циљ је изразити $m_0 + m_8$ преко p_i, k_i, једнакостима које важе се вероватноћом већом од 0.5. У оквиру једне ћелије и m_0 и m_8 одговарају биту b_0 у ниблу $b_0b_1b_2b_3$, па можемо да искористимо једнакости (1) и (2).

Искористимо најпре (1) за ћелије 1 и 3.

$p_0 + k_0\ p_1 + k_1\ p_2 + k_2\ p_3 + k_3$	$p_8 + k_8\ p_9 + k_9\ p_{10} + k_{10}\ p_{11} + k_{11}$
$p_4 + k_4\ p_5 + k_5\ p_6 + k_6\ p_7 + k_7$	$p_{12} + k_{12}\ p_{13} + k_{13}\ p_{14} + k_{14}\ p_{15} + k_{15}$

$$\overset{NS}{\rightarrow} \begin{array}{|c|c|} \hline m_0m_1m_2m_3 & m_8m_9m_{10}m_{11} \\ \hline m_4m_5m_6m_7 & m_{12}m_{13}m_{14}m_{15} \\ \hline \end{array}$$

Подсетимо се како делује табела S: $S(a_0a_1a_2a_3) = (b_0b_1b_2b_3)$.

Једначина (1) је $a_3 + b_0 = 1$, $p = 0.75$ (за ћелије 1 и 3).

$p_3 + k_3 + m_0 = 1$, $p = 0.75$

$p_{11} + k_{11} + m_8 = 1$, $p = 0.75$

$p_3 + p_{11} + k_3 + k_{11} + m_0 + m_8 = 0$, $p = (0.75)^2 + (0.25)^2 = 0.625$

$p_3 + p_{11} + k_3 + k_{11} = m_0 + m_8$, $p = 0.625$

Подсетимо се,

$c_3 + c_5 + c_{11} + c_{13} + k_{11} + k_{13} = m_0 + m_8$, (16) увек.

Комбиновањем се добија

$c_3 + c_5 + c_{11} + c_{13} + k_{11} + k_{13} = p_3 + p_{11} + k_3 + k_{11}$, $p = 0.625$

$p_3 + p_{11} + c_3 + c_5 + c_{11} + c_{13} = k_3 + k_{13}$ (19), $p = 0.625$

Ово је прва од једначина какве су потребне за линеарну криптоанализу.

Затим користимо једначину (1) на ћелију 1 и једначину (2) на ћелију 3 (то опет комбинујемо са (16)).

$p_0 + k_0\ p_1 + k_1\ p_2 + k_2\ p_3 + k_3$	$p_8 + k_8\ p_9 + k_9\ p_{10} + k_{10}\ p_{11} + k_{11}$
$p_4 + k_4\ p_5 + k_5\ p_6 + k_6\ p_7 + k_7$	$p_{12} + k_{12}\ p_{13} + k_{13}\ p_{14} + k_{14}\ p_{15} + k_{15}$

$\overset{NS}{\rightarrow}$	$m_0m_1m_2m_3$	$m_8m_9m_{10}m_{11}$
	$m_4m_5m_6m_7$	$m_{12}m_{13}m_{14}m_{15}$

Подсетимо се поново како делује табела S: $S(a_0a_1a_2a_3) = (b_0b_1b_2b_3)$.

Једначина (1) је $a_3 + b_0 = 1$, $p = 0.75$ (за ћелију 1).

Једначина (2) је $a_0 + a_1 + b_0 = 1$, $p = 0.75$ (за ћелију 3).

$p_3 + k_3 + m_0 = 1$, $p = 0.75$

$p_8 + k_8 + p_9 + k_9 + m_8 = 1$, $p = 0.75$

$p_3 + p_8 + p_9 + c_3 + c_5 + c_{11} + c_{13} = m_0 + m_8$, $p = 0.625$

Подсетимо се,

$c_3 + c_5 + c_{11} + c_{13} + k_{11} + k_{13} = m_0 + m_8$ (16) (увек)

Комбиновањем, добијамо

$p_3 + p_8 + p_9 + c_3 + c_5 + c_{11} + c_{13} = k_3 + k_8 + k_9 + k_{11} + k_{13}$ (20), $p = 0.625$

Ово је друга од једначина каква су потребне за линеарну криптоанализу.

Користећи (2) за обе ћелије 1 и 3, добијамо

$p_0 + p_1 + p_8 + p_9 + c_3 + c_5 + c_{11} + c_{13} = k_0 + k_1 + k_8 + k_9 + k_{11} + k_{13}$ (21).

Коришћењем (2) на ћелију 1 и (1) на ћелију 3 не даје нову информацију, него само оно што се може добити комбиновањем (19), (20) и (21).

Полазећи од (18) можемо такође да применимо (1) и (1), (1) и (2), (2) и (2) на ћелије 2 и 4 редом, и тако добијемо (22), (23), (24).

Полазећи од (15) можемо да применимо (3) и (3), (3) и (4), (4) и (4) на ћелије 1 и 3 редом, и тако добијемо (25), (26), (27).

Полазећи од (17) можемо да применимо (3) и (3), (3) и (4), (4) и (4) на ћелије 2 и 4 редом, и тако добијемо (28), (29), (30).

Сабирањем (15) и (16) добијамо $c_0 + c_3 + c_5 + c_6 + c_8 + c_{11} + c_{13} + c_{14} + k_8 + k_{11} + k_{13} + k_{14} = m_0 + m_1 + m_8 + m_9$.

$p_0 + k_0\ p_1 + k_1\ p_2 + k_2\ p_3 + k_3$	$p_8 + k_8\ p_9 + k_9\ p_{10} + k_{10}\ p_{11} + k_{11}$
$p_4 + k_4\ p_5 + k_5\ p_6 + k_6\ p_7 + k_7$	$p_{12} + k_{12}\ p_{13} + k_{13}\ p_{14} + k_{14}\ p_{15} + k_{15}$

$\overset{NS}{\rightarrow}$	$m_0m_1m_2m_3$	$m_8m_9m_{10}m_{11}$
	$m_4m_5m_6m_7$	$m_{12}m_{13}m_{14}m_{15}$

Подсетимо се поново како делује табела S: $S(a_0a_1a_2a_3) = (b_0b_1b_2b_3)$.

Једначина (5) је $a_1 + a_2 + b_0 + b_1 = 1$, $p = 0.75$ (за ћелије 1 и 3).

$p_1 + k_1 + p_2 + k_2 + m_0 + m_1 = 1$, $p = 0.75$

$p_9 + k_9 + p_{10} + k_{10} + m_8 + m_9 = 1$, $p = 0.75$.

$p_1 + p_2 + p_9 + p_{10} + k_1 + k_2 + k_9 + k_{10} = m_0 + m_1 + m_8 + m_9$, $p = 0.675$.

Подсетимо се,

$c_0+c_3+c_5+c_6+c_8+c_{11}+c_{13}+c_{14}+k_8+k_{11}+k_{13}+k_{14} = m_0+m_1+m_8+m_9$, (15)+(16) увек.

Комбиновањем се добија

$p_1 + p_2 + p_9 + p_{10} + c_0 + c_3 + c_5 + c_6 + c_8 + c_{11} + c_{13} + c_{14} = k_1 + k_2 + k_8 + k_9 + k_{10} + k_{11} + k_{13} + k_{14}$ (31), $p = 0.675$.

На први поглед изгледа корисно искористити (6), али се запажа да је то исто што и (2)+(3), па дакле редундантно. Сабирањем (17) и (18) такође се може искористити (5) на ћелије 2 и 4 да се добије (32).

Тако добијамо систем од 14 једначина

$$
\begin{aligned}
p_3 + p_{11} + c_3 + c_5 + c_{11} + c_{13} &= k_3 + k_{13} && (19)\\
p_3 + p_8 + p_9 + c_3 + c_5 + c_{11} + c_{13} &= k_3 + k_8 + k_9 + k_{11} + k_{13} && (20)\\
p_0 + p_1 + p_8 + p_9 + c_3 + c_5 + c_{11} + c_{13} &= k_0 + k_1 + k_8 + k_9 + k_{11} + k_{13} && (21)\\
p_7 + p_{15} + c_1 + c_7 + c_9 + c_{15} &= k_7 + k_9 && (22)\\
p_7 + p_{12} + p_{13} + c_1 + c_7 + c_9 + c_{15} &= k_7 + k_9 + k_{12} + k_{13} + k_{15} && (23)\\
p_4 + p_5 + p_{12} + p_{13} + c_1 + c_7 + c_9 + c_{15} &= k_4 + k_5 + k_9 + k_{12} + k_{13} + k_{15} && (24)\\
p_1 + p_9 + c_0 + c_6 + c_8 + c_{14} &= k_1 + k_8 + k_9 + k_{14} && (25)\\
p_1 + p_8 + p_9 + p_{10} + p_{11} + c_0 + c_6 + c_8 + c_{14} &= k_1 + k_9 + k_{10} + k_{11} + k_{14} && (26)\\
p_0 + p_1 + p_2 + p_3 + p_8 + p_9 + p_{10} + p_{11} + c_0 + c_6 + c_8 + c_{14} &= k_0 + k_1 + k_2 + k_3 + k_9 + k_{10} + k_{11} + k_{14} && (27)\\
p_5 + p_{13} + c_2 + c_4 + c_{10} + c_{12} &= k_5 + k_{10} + k_{12} + k_{13} && (28)\\
p_5 + p_{12} + p_{13} + p_{14} + p_{15} + c_2 + c_4 + c_{10} + c_{12} &= k_5 + k_{10} + k_{13} + k_{14} + k_{15} && (29)\\
p_4 + p_5 + p_6 + p_7 + p_{12} + p_{13} + p_{14} + p_{15} + c_2 + c_4 + c_{10} + c_{12} &= k_4 + k_5 + k_6 + k_7 + k_{10} + k_{13} + k_{14} + k_{15} && (30)\\
p_1 + p_2 + p_9 + p_{10} + c_0 + c_3 + c_5 + c_6 + c_8 + c_{11} + c_{13} + c_{14} &= k_1 + k_2 + k_8 + k_9 + k_{10} + k_{11} + k_{13} + k_{14} && (31)\\
p_5 + p_6 + p_{13} + p_{14} + c_1 + c_2 + c_4 + c_7 + c_9 + c_{10} + c_{12} + c_{15} &= k_5 + k_6 + k_9 + k_{10} + k_{12} + k_{13} + k_{14} + k_{15}; && (32)
\end{aligned}
$$

од којих свака важи са вероватноћом 0.625.

Овај систем линеарних једначина може се представити матричном једначином

$$
\begin{bmatrix}
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0\\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0\\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1\\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1\\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0\\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0\\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0\\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1\\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1\\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0\\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1
\end{bmatrix}
\begin{bmatrix} k_0\\ k_1\\ k_2\\ k_3\\ k_4\\ k_5\\ k_6\\ k_7\\ k_8\\ k_9\\ k_{10}\\ k_{11}\\ k_{12}\\ k_{13}\\ k_{14}\\ k_{15} \end{bmatrix}
=
\begin{bmatrix} p_3 + p_{11} + c_3 + c_5 + c_{11} + c_{13}\\ \cdots\\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ p_5 + p_6 + \cdots c_{15} \end{bmatrix}
$$

Врсте матрице система су линеарно независне, па је свих 14 једначина независно.

22.5 Напад

Полазећи од скупа познатих парова OT/CT, Цица израчунава $p_3 + p_{11} + c_3 + c_5 + c_{11} + c_{13}$ за све парове. Ако преовлађује 0, она ставља 0 на почетак вектора; ако преовлађује 1, она ставља 1 на почетак вектора. Због једноставности, претпоставимо да је Цица на свих 14 места могла донесе исправну одлуку да ли је на том месту 0 или 1. Она сада има 14 једначина са 16 непознатих $(k_0, \ldots, k_{15})$ па има $2^{16-14} = 4$ могућих решења за кључ. Најједноставније је

допунити матрицу система са још две врсте до базе (16-димензионалног $\mathbf{F}_2$ векторског простора) тако да се додају вектори придружени k_0 и k_4. Цица формира матричну једначину

$$\begin{bmatrix} 0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0\\ 0&0&0&1&0&0&0&0&1&1&0&1&0&1&0&0\\ 1&1&0&0&0&0&0&0&1&1&0&1&0&1&0&0\\ 0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&1&0&1&0&0&1&1&0&1\\ 0&0&0&0&1&1&0&0&0&1&0&0&1&1&0&1\\ 0&1&0&0&0&0&0&0&1&1&0&0&0&0&1&0\\ 0&1&0&0&0&0&0&0&0&1&1&1&0&0&1&0\\ 1&1&1&1&0&0&0&0&0&1&1&1&0&0&1&0\\ 0&0&0&0&0&1&0&0&0&0&1&0&1&1&0&0\\ 0&0&0&0&0&1&0&0&0&0&1&0&0&1&1&1\\ 0&0&0&0&1&1&1&1&0&0&1&0&0&1&1&1\\ 0&1&1&0&0&0&0&0&1&1&1&1&0&1&1&0\\ 0&0&0&0&0&1&1&0&0&1&1&0&1&1&1&1\\ 1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0 \end{bmatrix} \begin{bmatrix} k_0\\k_1\\k_2\\k_3\\k_4\\k_5\\k_6\\k_7\\k_8\\k_9\\k_{10}\\k_{11}\\k_{12}\\k_{13}\\k_{14}\\k_{15} \end{bmatrix} = \begin{bmatrix} p_3+p_{11}+c_3+c_5+c_{11}+c_{13}\\ \cdots\\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ p_5+p_6+\cdots c_{15}\\ *\\ * \end{bmatrix}$$

У вектору са десне стране она ставља нулу или јединицу — вредност која се чешће појављује у паровима OT/CT. За преостала два бита она испробава све четири могуће комбинације.

Претпоставимо да Цица жели да са вероватноћом 95% буде сигурна да су свих 14 избора тачни. Може се оценити број n парова OT/CT потребних за тај ниво сигурности успеха. За $i = 19, \ldots, 32$ нека је $\hat{p}_i$ случајна променљива чија је вредност једнака уделу парова OT/CT за које је лева страна једначине i једнака одговарајућој вредности (0 или 1) на десној страни, за дати кључ. За свако i је очекивана вредност $\hat{p}_i$ једнака 0.625, а њена варијанса је $0.625(1-0.625)/n$. Према томе, стандардна девијација $\hat{p}_i$ је $\sqrt{15/(64n)}$.

Ми хоћемо да буде $P(\hat{p}_i > 0.5,\ i = 19, \ldots, 32) = 0.95$. Можемо да претпоставимо да су $\hat{p}_i$ независне променљиве, јер о њима ништа не знамо и зато што је то вероватно блиско истини. Због тога захтевамо да буде $P(\hat{p}_i > 0.5) = \sqrt[14]{0.95} = 0.99634$, за свако i.

За велике n случајна променљива $\hat{p}_i$ има практично нормалну расподелу. Због тога од $\hat{p}_i$ може стандардизовати одузимањем математичког очекивања и дељењем са стандардном девијацијом, чиме се добија (приближно) нормално расподељена случајна променљива Z. Тада је

$$0.99634 = P(\hat{p}_i > 0.5) = P\left(\frac{\hat{p}_i - 0.625}{\sqrt{\frac{15}{64n}}} > \frac{0.5 - 0.625}{\sqrt{\frac{15}{64n}}}\right) = P\left(Z > \frac{-\sqrt{n}}{\sqrt{15}}\right).$$

Према томе,

$$1 - 0.99634 = 0.00366 = P\left(Z < \frac{-\sqrt{n}}{\sqrt{15}}\right) = P\left(Z > \frac{\sqrt{n}}{\sqrt{15}}\right).$$

Из таблице вредности за нормалну $(0, 1)$ расподелу добијамо да је $P(Z > 2.685) \approx 0.00366$. Вредност n одређујемо из $\sqrt{n}/\sqrt{15} = 2.685$; добијамо $n = 15(2.685)^2 \approx 109$.

Уопштење овог разматрања показује да за одређени ниво сигурности c, $0 < c < 1$ (ми смо изабрали $c = 0.95$) добијамо $n = 15z^2$, где је z вредност из табеле нормалне расподеле која одговара вредности $1 - \sqrt[14]{c}$. Тако је довољно само $n = 42$ парова OT/CT за сигурност изнад $c = 0.5$ Ако ни један од добијених кључева није добар, Цица може да набави још парова OT/CT. Друга могућност коју она има је да покуша да промени неке од 14 изабраних вредности бита. При томе има смисла прво променити онај бит који се појавио најмањи број пута.

22.5.1 Брзина напада

Ако Цица има само један пар OT/CT, она може да испроба свих $2^{16} = 65536$ кључева да установи који од кључева трансформише OT у CT. Ако овај услов испуњава више од једног кључа, она може да испроба кандидате на другом пару OT/CT — један од кључева ће у сваком случају бити добар у оба случаја. Ово је чисти напад грубом силом (brute force) У просеку Цица ће успети да пронађе исправан кључ после пола испробаних кључева, па се успех може очекивати после 2^{15} шифровања.

Да би линеарна криптоанализа успела са поузданошћу 95% Цица треба да израчуна вредности 14 различитих линеарних комбинација $\sum_{i \in S_1} p_i \oplus \sum_{j \in S_2} c_j$ за сваки од 109 парова OT/CT. После тога она треба да изврши напад грубом силом са четири могућа кључа.

Према томе, линеарна криптоанализа изгледа ефикасно у односу на напад грубом силом на 1SAES . Међутим, кад се повећа број рунди, потребно је више сабирања једначина да би се елиминисале непознати међубити (као што су m_i, n_i), па вероватноће придружене једначинама теже вредности 0.5 (у нашем примеру су вероватноће отишле са 0.75 на 0.625). Последица је да је за напад потребно много више парова OT/CT да би се постигао пристојан ниво поузданости избора тачних вредности $\sum_{l \in S_3} k_l$.

22.6 Диференцијална криптоанализа

Диференцијална криптоанализа (Biham, Shamir) је напад са изабраним отвореним текстом. Обично је непрактичан. Ако је на располагању огроман број парова OT/CT, онда међу њима могу да се пронађу оне који би ионако требали да буду изабрани (таква ситуација је реална са паметном картицом или кутијом за скрембловање кабловске телевизије). Цица бира два OT који се разликују само на одређеним битима, а подударни су на осталим битима,

и посматра разлике између два одговарајућа шифрата, покушавајући да на основу тога нешто закључи о кључу.

Диференцијална криптоанализа 1SAES : $A_{K_1} \circ MC \circ SR \circ NS \circ A_{K_0}$. Претпостављамо да је исти кључ коришћен за шифровање два ОТ $p_0 \dots p_{15}$ и $p_0^* \dots p_{15}^*$, при чему је $p_8 \dots p_{15} = p_8^* \dots p_{15}^*$, и тако се добијају два СТ $c_0 \dots c_{15}$ и $c_0^* \dots c_{15}^*$. Потребно је да се разликују ниблови $p_0p_1p_2p_3 \neq p_0^*p_1^*p_2^*p_3^*$ (није проблем ако су неки бити једнаки). Потребно је да буде и $p_4p_5p_6p_7 \neq p_4^*p_5^*p_6^*p_7^*$.

$p_0p_1p_2p_3$	$p_8p_9p_{10}p_{11}$
$p_4p_5p_6p_7$	$p_{12}p_{13}p_{14}p_{15}$

$p_0^*p_1^*p_2^*p_3^*$	$p_8p_9p_{10}p_{11}$
$p_4^*p_5^*p_6^*p_7^*$	$p_{12}p_{13}p_{14}p_{15}$

$\xrightarrow{NS}$

$p_0 + k_0\ p_1 + k_1\ p_2 + k_2\ p_3 + k_3$	$p_8 + k_8\ p_9 + k_9\ p_{10} + k_{10}\ p_{11} + k_{11}$
$p_4 + k_4\ p_5 + k_5\ p_6 + k_6\ p_7 + k_7$	$p_{12} + k_{12}\ p_{13} + k_{13}\ p_{14} + k_{14}\ p_{15} + k_{15}$

и

$p_0^* + k_0\ p_1^* + k_1\ p_2^* + k_2\ p_3^* + k_3$	$p_8 + k_8\ p_9 + k_9\ p_{10} + k_{10}\ p_{11} + k_{11}$
$p_4^* + k_4\ p_5^* + k_5\ p_6^* + k_6\ p_7^* + k_7$	$p_{12} + k_{12}\ p_{13} + k_{13}\ p_{14} + k_{14}\ p_{15} + k_{15}$

Примена NS (S делује на сваки нибл) даје

$m_0m_1m_2m_3$	$m_8m_9m_{10}m_{11}$
$m_4m_5m_6m_7$	$m_{12}m_{13}m_{14}m_{15}$

$m_0^*m_1^*m_2^*m_3^*$	$m_8m_9m_{10}m_{11}$
$m_4^*m_5^*m_6^*m_7^*$	$m_{12}m_{13}m_{14}m_{15}$

Примена SR (цикличко померање друге врсте) даје

$m_0m_1m_2m_3$	$m_8m_9m_{10}m_{11}$
$m_{12}m_{13}m_{14}m_{15}$	$m_4m_5m_6m_7$

$m_0^*m_1^*m_2^*m_3^*$	$m_8m_9m_{10}m_{11}$
$m_{12}m_{13}m_{14}m_{15}$	$m_4^*m_5^*m_6^*m_7^*$

Примена MC (мешање колона) даје

$m_0 + m_{14}\ m_1 + m_{12} + m_{15}\ m_2 + m_{12} + m_{13}\ m_3 + m_{13}$	$m_6 + m_8\ m_4 + m_7 + m_9\ m_4 + m_5 + m_{10}\ m_5 + m_{11}$
$m_2 + m_{12}\ m_0 + m_3 + m_{13}\ m_0 + m_1 + m_{14}\ m_1 + m_{15}$	$m_4 + m_{10}\ m_5 + m_8 + m_{11}\ m_6 + m_8 + m_9\ m_7 + m_9$

и

$m_0^* + m_{14}\ m_1^* + m_{12} + m_{15}\ m_2^* + m_{12} + m_{13}\ m_3^* + m_{13}$	$m_6^* + m_8\ m_4^* + m_7^* + m_9\ m_4^* + m_5^* + m_{10}\ m_5^* + m_{11}$
$m_2^* + m_{12}\ m_0^* + m_3^* + m_{13}\ m_0^* + m_1^* + m_{14}\ m_1^* + m_{15}$	$m_4^* + m_{10}\ m_5^* + m_8 + m_{11}\ m_6^* + m_8 + m_9\ m_7^* + m_9$

Примена A_{K_1} даје

$m_0 + m_{14} + k_{16}\ m_1 + m_{12} + m_{15} + k_{17}\ m_2 + m_{12} + m_{13} + k_{18}\ m_3 + m_{13} + k_{19}$	
$m_2 + m_{12} + k_{20}\ m_0 + m_3 + m_{13} + k_{21}\ m_0 + m_1 + m_{14} + k_{22}\ m_1 + m_{15} + k_{23}$	

$m_6 + m_8 + k_{24}\ m_4 + m_7 + m_9 + k_{25}\ m_4 + m_5 + m_{10} + k_{26}\ m_5 + m_{11} + k_{27}$
$m_4 + m_{10} + k_{28}\ m_5 + m_8 + m_{11} + k_{29}\ m_6 + m_8 + m_9 + k_{30}\ m_7 + m_9 + k_{31}$

и

$m_0^* + m_{14} + k_{16}\ m_1^* + m_{12} + m_{15} + k_{17}\ m_2^* + m_{12} + m_{13} + k_{18}\ m_3^* + m_{13} + k_{19}$	
$m_2^* + m_{12} + k_{20}\ m_0^* + m_3^* + m_{13} + k_{21}\ m_0^* + m_1^* + m_{14} + k_{22}\ m_1^* + m_{15} + k_{23}$	

$m_6^* + m_8 + k_{24}$	$m_4^* + m_7^* + m_9 + k_{25}$	$m_4^* + m_5^* + m_{10} + k_{26}$	$m_5^* + m_{11} + k_{27}$
$m_4^* + m_{10} + k_{28}$	$m_5^* + m_8 + m_{11} + k_{29}$	$m_6^* + m_8 + m_9 + k_{30}$	$m_7^* + m_9 + k_{31}$

Ово је шифрат, тј.

$c_0c_1c_2c_3$	$c_8c_9c_{10}c_{11}$
$c_4c_5c_6c_7$	$c_{12}c_{13}c_{14}c_{15}$

$c_0^*c_1^*c_2^*c_3^*$	$c_8^*c_9^*c_{10}^*c_{11}^*$
$c_4^*c_5^*c_6^*c_7^*$	$c_{12}^*c_{13}^*c_{14}^*c_{15}^*$

Сабирањем (по модулу два) претпоследњих блокова (са m_i и k_i) добијамо

$m_0 + m_0^*\ m_1 + m_1^*\ m_2 + m_2^*\ m_3 + m_3^*$	небитно
небитно	$m_4 + m_4^*\ m_5 + m_5^*\ m_6 + m_6^*\ m_7 + m_7^*$

што је једнако збиру (по модулу два) последњих блокова (са c_i)

$c_0 + c_0^*\ c_1 + c_1^*\ c_2 + c_2^*\ c_3 + c_3^*$	небитно
небитно	$c_{12} + c_{12}^*\ c_{13} + c_{13}^*\ c_{14} + c_{14}^*\ c_{15} + c_{15}^*$

Добија се да је $c_0c_1c_2c_3 + c_0^*c_1^*c_2^*c_3^*$
$= (m_0 + m_{14}, \ldots, m_3 + m_{13}) + (m_0^* + m_{14}, \ldots, m_3^* + m_{13})$
$m_0m_1m_2m_3 + m_0^*m_1^*m_2^*m_3^*$
$= S(p_0p_1p_2p_3 + k_0k_1k_2k_3) + S(p_0^*p_1^*p_2^*p_3^* + k_0k_1k_2k_3)$.
Према томе, ако је $p_8 \ldots p_{15} = p_8^* \ldots p_{15}^*$ онда је
$S(p_0p_1p_2p_3 + k_0k_1k_2k_3) + S(p_0^*p_1^*p_2^*p_3^* + k_0k_1k_2k_3) = c_0c_1c_2c_3 + c_0^*c_1^*c_2^*c_3^*$. (I).
Овде се све зна изузев $k_0k_1k_2k_3$. Приметимо да једначина ЗАВИСИ од $p_8 \ldots p_{15} = p_8^* \ldots p_{15}^*$.
Слично, $c_{12}c_{13}c_{14}c_{15} + c_{12}^*c_{13}^*c_{14}^*c_{15}^*$
$= m_4m_5m_6m_7 + m_4^*m_5^*m_6^*m_7^*$
$= S(p_4p_5p_6p_7 + k_4k_5k_6k_7) + S(p_4^*p_5^*p_6^*p_7^* + k_4k_5k_6k_7)$.
Према томе, ако је $p_8 \ldots p_{15} = p_8^* \ldots p_{15}^*$ онда је
$S(p_4p_5p_6p_7 + k_4k_5k_6k_7) + S(p_4^*p_5^*p_6^*p_7^* + k_4k_5k_6k_7) = c_{12}c_{13}c_{14}c_{15} + c_{12}^*c_{13}^*c_{14}^*c_{15}^*$. (II).
Овде се све зна изузев $k_4k_5k_6k_7$.
Пример. Цица шифрује

$$\begin{array}{rcccc} & p_0 & & & p_{15} \\ ASCII\ No = & 0100 & 1110 & 0110 & 1111 \\ CT = & 0010 & 0010 & 0100 & 1101 \end{array}$$

и

$$\begin{array}{rcccc} & p_0^* & & & p_{15}^* \\ ASCII\ to = & 0111 & 0100 & 0110 & 1111 \\ CT = & 0000 & 1010 & 0001 & 0001 \end{array}$$

I: $S(0100 + k_0k_1k_2k_3) + S(0111 + k_0k_1k_2k_3) = 0010 + 0000 = 0010$.

кандидат	A	B			
$k_0k_1k_2k_3$	$+0100$	$+0111$	$S(A)$	$S(B)$	$S(A)+S(B)$
0000	0100	0111	1101	0101	1000
0001	0101	0110	0001	1000	1001
0010	0110	0101	1000	0001	1001
0011	0111	0100	0101	1101	1000
0100	0000	0011	1001	1011	0010
0101	0001	0010	0100	1010	1110
0110	0010	0001	1010	0100	1110
0111	0011	0000	1011	1001	0010
1000	1100	1111	1100	0111	1011
1001	1101	1110	1110	1111	0001
1010	1110	1101	1111	1110	0001
1011	1111	1100	0111	1100	1011
1100	1000	1011	0110	0011	0101
1101	1001	1010	0010	0000	0010
1110	1010	1001	0000	0010	0010
1111	1011	1000	0011	0110	0101

Према томе, $k_0k_1k_2k_3 \in \{0100, 0111, 1101, 1110\}$. Приметимо да је скуп кандидата одређен другом и трећом колоном. То су парови са збиром (XOR) 0011. Према томе, ако је $p_0p_1p_2p_3 + p_0^*p_1^*p_2^*p_3^* = 0011$, добијају се исти кандидати. Због тога проналазимо други пар за који је $p_0p_1p_2p_3 + p_0^*p_1^*p_2^*p_3^* \neq 0011$, (0000).

Цица шифрује

$$\begin{array}{rcccc} & p_0 & & & p_{15} \\ ASCII\ Mr = & 0100 & 1101 & 0111 & 0010 \\ CT = & 1101 & 0101 & 1101 & 0000 \end{array}$$

и

$$\begin{array}{rcccc} & p_0^* & & & p_{15}^* \\ ASCII\ or = & 0110 & 1111 & 0111 & 0010 \\ CT = & 1100 & 0001 & 0100 & 1111 \end{array}$$

I: $S(0100+k_0k_1k_2k_3)+S(0110+k_0k_1k_2k_3) = 1101+1100 = 0001$. Добијају се кандидати $k_0k_1k_2k_3 \in \{1101, 1111\}$. Пресек два скупа је $k_0k_1k_2k_3 = 1101$.

Користећи II и ова два пара OT/CT, Цица добија да је $k_4k_5k_6k_7 = 1100$.

Да би се добили $k_8k_9k_{10}k_{11}$ и $k_{12}k_{13}k_{14}k_{15}$, потребне су још две једначине III и IV (које треба одредити за домаћи задатак).

Са довољно рунди (> 3 за DES , > 1 за AES) немогуће је пронаћи једначине са вероватноћом 1. Уместо тога, проналазе се једначине са мањом вероватноћом p. Тада се тачни бити кључа појављују са уделом p у скуповима кандидата. Остали се појављују случајно, а пошто их има много (прави DES , AES) они ће се појављивати много ређе.

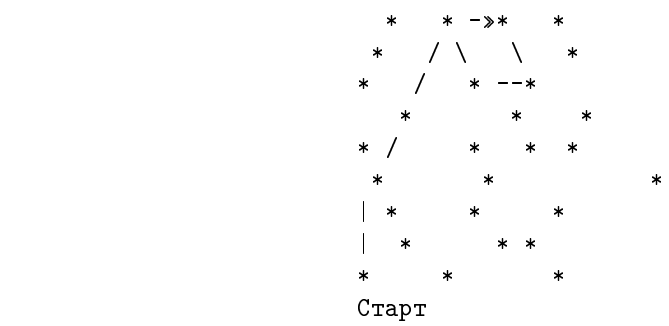
23 Рођендански парадокс

Ако се у соби налази више од 23 особе, онда је вероватноћа да неке две од њих имају рођендан истог дана већа од 1/2. Уопште, ако се $k = \alpha\sqrt{n}$ објеката вади са понављањем из скупа величине n, онда је вероватноћа да ће сви извучени објекти бити различити

$$\prod_{i=1}^{k-1} \frac{n-i}{n} = \prod_{i=1}^{k-1}(1-\frac{i}{n}) \approx \prod_{i=1}^{k-1} e^{-i/n} = e^{-\sum_{i=1}^{k-1} i/n} \approx e^{-k^2/(2n)} = e^{-\alpha^2/2}$$

Дакле, вероватноћа да нека два извучена објекта буду иста је приближно $1-e^{-\alpha^2/2}$. Ако се нпр. извлачи $\sqrt{n\ln 4} \approx 1.2\sqrt{n}$ објеката, онда је вероватноћа понављања око 1/2. У случају особа у просторији, да вероватноћа поклапања два рођендана буде 1/2 довољно је да број особа буде већи од $\sqrt{365\ln 4} \approx 23$.

У току случајног лутања по скупу од n елемената, после отприлике $1.2\sqrt{n}$ очекује се наилазак на већ раније прегледани елеменат. Коришћење ове чињенице зове се Полардов (Pollard) ρ-метод. Очекивани број корака пре повратка у неку виђену тачку је $O(\sqrt{n})$. Наредна слика илуструје зашто се овакво случајно лутање зове ρ-метод (обратите пажњу на облик).



Алгоритам факторизације заснован на овој чињеници био је први алгоритам суштински бржи од факторизације низом пробних дељења. Тај алгоритам је још увек најбољи за бројеве са $8-15$ декадних цифара. Да бисмо раставили n на чиниоце, итерирамо функцију као што је нпр. $f(x) = x^2+1 \pmod n$ (полазећи од нпр. $x = 0$), и тако добијамо случајно лутање кроз $\mathbf{Z}_n$. Ако је $n = pq$, очекујемо да ћемо се по модулу p вратити назад (на место где смо већ били), а да се то неће истовремено десити и по модулу q.

Пример. Факторизација 1357. Користимо пресликавање x^2+1, односно формирамо низ $a_{m+1} = a_m^2 + 1 \pmod{1357}$, $a_0 = 0$. Понављање је могуће открити једноставним алгоритмом који користи мало меморије и не захтева прегледање претходно добијених бројева.

1) Израчунај a_1, a_2, a_3, $\mathrm{nzd}(a_2 - a_1, n)$, запамти a_1, a_2
2) Израчунај a_2, a_3, a_4, $\mathrm{nzd}(a_4 - a_2, n)$, запамти a_2, a_4, обриши a_1, a_2
3) Израчунај a_3, a_5, a_6, $\mathrm{nzd}(a_6 - a_3, n)$, запамти a_3, a_6, обриши a_2, a_4
4) Израчунај a_4, a_7, a_8, $\mathrm{nzd}(a_8 - a_4, n)$, запамти a_4, a_8, обриши a_3, a_6

5) Израчунај a_5, a_9, a_{10}, $\mathrm{nzd}(a_{10}-a_5, n)$, запамти a_5, a_{10}, обриши a_4, a_8, итд.

i	$2i-1$	$2i$	a_i	a_{2i-1}		a_{2i}	$\mathrm{nzd}(a_{2i}-a_i, 1357)$
1	1	2	1	1	$\to$	2	1
			$\downarrow$		$\swarrow$		
2	3	4	2	5	$\to$	26	1
			$\downarrow$		$\swarrow$		
3	5	6	5	677	$\to$	1021	1
			$\downarrow$		$\swarrow$		
4	7	8	26	266	$\to$	193	1
			$\downarrow$		$\swarrow$		
5	9	10	677	611	$\to$	147	1
			$\downarrow$		$\swarrow$		
6	11	12	1021	1255	$\to$	906	23

Према томе, $23|1357$ и $1357/23 = 59$. Приметимо да је израчунавање $x^2+1 \pmod n$ и nzd једноставно.

Зашто је овај поступак успешан? Погледајмо шта се иза сцене дешава по модулу 23 и 59.

a_i	mod 1357	a_i	mod 23	a_i	mod 59
	$a_1 = 1$		1		1
	$a_2 = 2$		2		2
	$a_3 = 5$		5		5
	$a_4 = 26$		3		26
	$a_5 = 677$		10		28
	$a_6 = 1021$		9		18
	$a_7 = 266$		13		30
	$a_8 = 193$		9		16
	$a_9 = 611$		13		21
	$a_{10} = 147$		9		29
	$a_{11} = 1255$		13		16
	$a_{12} = 906$		9		21

Да ли би било ефикасније да се направи списак $a_1, \ldots, a_8$, и да се у сваком кораку израчунава nzd са свим претходним члановима низа? Не. Ако се ρ (повратак по модулу p на место на коме смо већ били) деси после m корака (у горњем примеру се то десило за $m = 8$) онда је потребно $1+2+\cdots+(m-1) \approx m^2/2$ израчунавања nzd и велики меморијски простор. У описаном алгоритму има само $4m$ корака.

Колико треба чекати до дешавања првог ρ? Ако је $n = pq$ и $p < \sqrt{n}$, онда алгоритам захтева $O(\sqrt{p})$ (за случајно лутање кроз $\mathbf{F}_p$) $= O(\sqrt[4]{n}) = e^{O(\frac{1}{4}\log n)}$. Факторизација низом пробних дељења има сложеност $= O(\sqrt{n}) = e^{O(\frac{1}{2}\log n)}$, а алгоритам сита у пољу бројева само $= e^{O((\log n)^{1/3}(\log\log n)^{2/3})}$.

Иста идеја може се искористити за решавање проблема дискретног логаритма за елиптичке криве над коначним пољем. То је за сада најбољи познати алгоритам за решавање ЕЦДЛП. Нека је $E : y^2 = x^3 + 17x + 1$ над $\mathbf{F}_{101}$. Тачка $G = (0, 1)$ генерише $E(\mathbf{F}_{101})$. Поред тога, $103G = \oslash$, па се тачке целим бројевима множе по модулу 103. Тачка $Q = (5, 98) = nG$ за неко n; потребно је одредити n. Нека x(тачка) означава x-координату тачке; тако је нпр. $x(Q) = 5$.

Посматрајмо случајно лутање кроз $E(\mathbf{F}_{101})$ описано на следећи начин. Нека је $v_0 = [0, 0]$ и $P_0 = \oslash$. Нека је вектору $v = [a, b]$ придружена тачка $P_i = a_iQ + b_iG$, где су a_i, b_i неки остаци по модулу 103. Алгоритам лутања:

Ако је $x(P_i) \leq 33$ или $P_i = \oslash$, онда $P_{i+1} = Q + P_i$, $v_{i+1} = v_i + [1, 0]$.

Ако је $33 < x(P_i) \leq 68$, онда $P_{i+1} = 2P_i$, $v_{i+1} = 2v_i$.

Ако је $68 < x(P_i)$, онда $P_{i+1} = G + P_i$, $v_{i+1} = v_i + [0, 1]$.

Ако је $P_{2j} = P_j$, крај. Тада је $P_{2j} = a_{2j}Q + b_{2j}G = a_jQ + b_jG = P_j$. Због тога је $(a_{2j} - a_j)Q = (b_j - b_{2j})G$ и $Q = (b_j - b_{2j})(a_{2j} - a_j)^{-1}G$, где је $(a_{2j} - a_j)^{-1}$ израчунато по модулу 103.

i	P_i	$[a, b]$
	0	$[0, 0]$
1	$(5, 98)$	$[1, 0]$
2	$(68, 60)$	$[2, 0]$
3	$(63, 29)$	$[2, 1]$
4	$(12, 32)$	$[4, 2]$
5	$(8, 89)$	$[5, 2]$
6	$(97, 77)$	$[6, 2]$
7	$(62, 66)$	$[6, 3]$
8	$(53, 81)$	$[12, 6]$
9	$(97, 77)$	$[24, 12]$
10	$(62, 66)$	$[24, 13]$
11	$(53, 81)$	$[48, 26]$
12	$(97, 77)$	$[96, 52]$

Приметимо да је $P_{12} = P_6$, па је $6Q + 2G = 96Q + 52G$. Због тога је $-90Q = 50G$ и $Q = (-90)^{-1}50G$. Пошто је $(-90)^{-1}50 \equiv 91 \pmod{103}$, биће $Q = 91G$. Наравно, ми уствари израчунавамо P_1, P_1, P_2 и упоређујемо P_1, P_2. Затим израчунавамо P_2, P_3, P_4 и упоређујемо P_2, P_4. Затим израчунавамо P_3, P_5, P_6 и упоређујемо P_3, P_6,....

Слична идеја може се искористити за решавање ФФДЛП.

Пример. $g = 2$ генерише $\mathbf{F}_{101}$. Нека је $y = 86 = g^x$. Одредити x.

Потребно је дефинисати случајно лутање кроз $\mathbf{F}_{101}^*$. Нека је $c_0 = 1$ и $v_0 = [0, 0]$. Вектору $v_i = [a_i, b_i]$ одговара $c_i = y^{a_i}g^{b_i} = 86^{a_i}2^{b_i}$. Приметимо да су сви a_i, b_i дефинисани по модулу 100.

Ако је $c_i \leq 33$, онда $c_{i+1} = c_iy = 86c_i$, $v_{i+1} = v_i + [1, 0]$.

Ако је $33 < c_i < 68$, онда $c_{i+1} = c_i^3$, $v_{i+1} = 3v_i$ (број 3 се користи јер је узајамно прост са 100, редом групе).

Ако је $68 \leq c_i$, онда $c_{i+1} = c_ig$, $v_{i+1} = v_i + [0, 1]$.

Лутање се прекида у тренутку кад буде испуњен услов $c_{2j} = c_j$. Тада је $c_{2j} = y^{a_{2j}} g^{b_{2j}} = y^{a_j} g^{b_j} = c_j$. Према томе, $y^{a_{2j}-a_j} = g^{b_j-b_{2j}}$ и $y = g^{(b_j-b_{2j})(a_{2j}-a_j)^{-1}}$, где се инверзија експонента врши по модулу 100. Резултат лутања:

i	$c_i = 86^{a_i} 2^{b_i}$	$[a_i, b_i]$
0	1	$[0, 0]$
1	86	$[1, 0]$
2	71	$[1, 1]$
3	41	$[1, 2]$
4	39	$[3, 6]$
5	32	$[9, 18]$
6	25	$[10, 18]$
7	29	$[11, 18]$
8	70	$[12, 18]$
9	39	$[12, 19]$
10	32	$[36, 57]$

Дакле, $32 = 86^{36} 2^{57} = 86^9 2^{18}$ и $86^{27} = 2^{18-57}$. Сада је $18-57 \equiv 61 \pmod{100}$, па због тога $86^{27} = 2^{61}$ и $86 = 2^{61(27^{-1})}$. Даље је $61(27^{-1}) \equiv 61(63) \equiv 43 \pmod{100}$ па је $86 = 2^{43}$ у $\mathbf{F}_{101}$. Приметимо да ако компоненте вектора $[a_i, b_i]$ постану ≥ 100, онда се замењују својим остатком по модулу 100. Ако $(a_{2j} - a_j)$ није инвертибилно по модулу 100, онда бисмо пронашли сва решења једначине $(a_{2j} - a_j)x = b_j - b_{2j} \pmod{100}$.

24 Факторизација

Најочигледнији начин за разбијање RSA почиње факторизацијом $n = pq$. Кад говоримо о проблему факторизације, претпостављамо да се тражи било који нетривијални фактор броја n, па претпостављамо да је n непарно. Тако је $105 = 7 \cdot 15$ успешна факторизација, без бзира на то што је 15 сложен број.

Најбољи познати алгоритам за факторизацију неког RSA броја је сито у пољу бројева. Факторизација се појављује и у другим криптографским контекстима. У вези са Полиг-Хелмановим алгоритмом видећемо да је важно раставити на чиниоце величину групе облика $\mathbf{F}_q^*$ или $E(\mathbf{F}_q)$. На тај начин постајемо сигурнији да је проблем дискретног логаритма у тим групама тежак. За такве факторизације најчешће се користе други алгоритми (не сито у пољу бројева). Због тога ћемо размотрити и друге алгоритме за факторизацију (као што су употреба верижних разломака и елиптичке криве) који се користе у тим другим криптографским контекстима.

Факторизација низом дељења је процес дељења n сваким простим бројем мањим од $\sqrt{n}$. То је врло неефикасно, али је ипак најефикаснији поступак за факторизацију бројева до 15 цифара.

Већина алгоритама за факторизацију заснива се на следећем. Због једноставности претпоставићемо да је $n = pq$, где су p и q непарни прости бројеви. Претпоставимо

да смо пронашли бројеве x и y такве да је $x^2 \equiv y^2 \pmod{n}$, али $x \not\equiv \pm y \pmod{n}$. Тада $n|x^2-y^2$, па $n|(x+y)(x-y)$, односно $pq|(x+y)(x-y)$. Надамо се да је нпр. $p|(x+y)$ и $q|(x-y)$. Ако је то испуњено, онда је $\text{nzd}(x+y,n) = p$ (рачунање nzd је ефикасно) и $q = n/p$. Ако то не важи, онда $n = pq|x+y$ или $n = pq|x-y$, па је $x \equiv \pm y \pmod{n}$, па се мора покушати поново.

Приметимо да n не мора да буде производ тачно два проста броја. Описана идеја ради и за компликованије n. У општем случају је $\text{nzd}(x-y,n)$ неки делилац n.

Размотрићемо сада неке алгоритме за налажење x и y.

24.1 Фермаова факторизација n

Овај алгоритам заснива се на чињеници да је за мале бројеве вероватније да буду квадрати, него за велике бројеве. Бескорисно је покушавати редом са $x = 1, 2, 3 \ldots$ јер је остатак $x^2 \pmod{n}$ једнак x^2. Због тога мора да буде $x^2 > n$, односно $x \sqrt{n}$. Ми хоћемо да остатак буде мали, па бирамо x тако да буде само мало веће од $\sqrt{n}$.

Најмањи цели број $\geq x$ означава се са $\lceil x \rceil$. Тако је $\lceil 1.5 \rceil = 2$ и $\lceil 3 \rceil = 3$. Најпре израчунавамо $\lceil \sqrt{n} \rceil$. Затим израчунавамо $\sqrt{\lceil \sqrt{n} \rceil^2 - n}$. Ако то није цели број, онда израчунавамо $\sqrt{(\lceil \sqrt{n} \rceil + 1)^2 - n}$. Ако то није цели број, онда израчунавамо $\sqrt{(\lceil \sqrt{n} \rceil + 2)^2 - n}$, итд. све док резултат не буде цели број.

Пример. $n = 3229799$, $\sqrt{n} \approx 1797.16$, па је $\lceil \sqrt{n} \rceil = 1798$. $1798^2 - n = 3005$, али $\sqrt{3005} \notin Z$. $1799^2 - n = 6602$, али $\sqrt{6602} \notin Z$. $1800^2 - n = 10201$ и $\sqrt{10201} = 101$. Према томе, $1800^2 - n = 101^2$ и $1800^2 - 101^2 = n$ па је $(1800 + 101)(1800 - 101) = n = 1901 \cdot 1699$.

Фермаова факторизација ради добро кад n није много веће од потпуног квадрата.

24.2 Базе фактора

За број n кажемо да је b-гладак ако су сви његови прости чиниоци $\leq b$. Тако је нпр. $5280 = 2^5 \cdot 3 \cdot 5 \cdot 11$ 20-гладак број. Неформално, број је гладак ако се разлаже у производ релативно малих простих бројева.

У већини савремених алгоритама за факторизацију (са верижним разломцима, квадратно сито или сито у пољу бројева) циљ је пронаћи x-ове такве да је $x^2 \pmod{n}$ гладак број. Кад се пронађу такви бројеви, покушава се саналажењем неког њиховог производа једнак потпуном квадрату $\pmod{n}$. Један начин да се то постигне је да сами остаци буду мали. Дакле, бројеви x бирају се тако да x^2 буде блиско неком умношку n, тј. $x^2 \approx kn$ за неко $k \in \mathbf{Z}$, тј. $x \approx \sqrt{kn}$. Затим се креира база фактора, која се састоји од простих бројева $\leq b$, изабране границе глаткости. Избор оптималног b зависи од компликованог објашњења из теорије бројева. Граница b расте са n.

Желимо да искористимо $x^2 \approx kn$ без обзира да ли је $x^2 > kn$ или $x^2 < kn$. Ако је $x^2 < kn$, онда је $x^2 \equiv -1 \cdot l \pmod{n}$, где је l мало. Због тога и -1 укључујемо у нашу базу фактора. За свако x за које је $x^2 \approx kn$ израчунава се остатак $r = x^2 \pmod{n}$. Ако је r остатак близак n, онда се уместо њега

користи $-1 \cdot (n-r)$. Затим се остатак растављa на чиниоце, користећи базу фактора и низ дељења. Ако се не успе, покушава се са другим x. Наставља се док се не дође до тога да је производ неког подскупа остатака потпуни квадрат.

Једноставан начин да се искористи ова идеја је бирати целе бројеве најближе $\sqrt{kn}$. Број таквих блиских бројева зависи од n на компликовани начин. Нека је дакле x неки цели број близак $\sqrt{kn}$.

Пример. $n = 89893$, користимо $b = 20$ и четири најближа броја за сваки $\sqrt{kn}$. Имамо да је $\sqrt{n} \approx 299.8$.

		-1	2	3	5	7	11	13	17	19
$299^2 \equiv -492$	није гладак									
$300^2 \equiv 107$	није гладак									
$298^2 \equiv -1089$	$-1 \cdot 3^2 \cdot 11^2$	1	0	0	0	0	0	0	0	0
$301^2 \equiv 708$	није гладак									
$\sqrt{2n} \approx 424.01$										
$424^2 \equiv -10$	$-1 \cdot 2 \cdot 5$	1	1	0	1	0	0	0	0	0
$425^2 \equiv 839$	није гладак									
$423^2 \equiv -857$	није гладак									
$426^2 \equiv 1690$	$= 2 \cdot 5 \cdot 13^2$	0	1	0	1	0	0	0	0	0

Проналажење неког производа остатака једнак квадрату еквивалентно је проналажењу подскупа вектора са десне стране којима је збир по модулу 2 нула вектор. Видимо да је $298^2 \cdot 424^2 \cdot 426^2 \equiv (-1) \cdot 3^2 \cdot 11^2 \cdot (-1) \cdot 2 \cdot 5 \cdot 2 \cdot 5 \cdot 13^2 \pmod{n}$. Дакле, $(298 \cdot 424 \cdot 426)^2 \equiv ((-1) \cdot 2 \cdot 3 \cdot 5 \cdot 11 \cdot 13)^2 \pmod{n}$. Подсетимо се да ако је $x^2 \equiv y^2 \pmod{n}$ и $x \not\equiv \pm y \pmod{n}$ онда је $\mathrm{nzd}(x+y, n)$ нетривијални фактор n. Сада израчунавамо остатак производа у загради: $298 \cdot 424 \cdot 426 \equiv 69938 \pmod{n}$ и $(-1) \cdot 2 \cdot 3 \cdot 5 \cdot 11 \cdot 13 \equiv 85603 \pmod{n}$. Примећујемо да је $\mathrm{nzd}(69938 + 85603, n) = 373$ и $n/373 = 241$.

Овај пример није баш репрезентативан, јер смо искористили сваки вектор који се појавио. Могли су бити добијени нпр. вектори $v_1 = [1,0,0,0,0,0,0\ldots]$, $v_2 = [1,1,0,1,0,0,0\ldots]$, $v_3 = [0,0,1,0,1,0,0\ldots]$, $v_4 = [1,0,0,0,0,1,0\ldots]$, $v_5 = [0,1,0,1,0,0,0\ldots]$ (где $\ldots$ означава нуле). Затим бисмо морали да пронађемо подскуп вектора којима је збир 0 по модулу 2. Ако са M означимо матрицу чије су колоне $v_1, \ldots, v_5$, онда треба да израчунамо нула простор матрице M по модулу 2. Било који ненула вектор из нула простора даје жељену линеарну комбинацију вектора v_i. Нула простор матрице M има базу $(1,1,0,0,1)$, па је $v_1 + v_2 + v_5 = 0$ по модулу 2.

24.3 Факторизација помоћу верижних разломака

Овај алгоритам био је најбољи алгоритам за факторизацију око 1975. године. Он је и даље најбржи за целе бројеве умерене величине. Потребно је да x^2 буде блиско умношку n. Нека је b/c неки парцијални разломак добијен полазећи од вержног развоја $\sqrt{n}$. Тада је $b/c \approx \sqrt{n}$, па је $b^2/c^2 \approx \sqrt{n}$, односно $b^2 \approx c^2 n$, тј. b^2 је блиско умношку n, односно $b^2 \pmod{n}$ је мали број.

Нека је $n = 17873$. Почетак верижног развоја $\sqrt{n}$ је $[133, 1, 2, 4, 2, 3, 1, 2, 1, 2, 3, 3, \ldots]$. Користићемо базу фактора $\{-1, 2, 3, 5, 7, 11, 13, 17, 19, 23, 29\}$ при чему нећемо уписивати нуле у таблицу.

			-1	2	3	5	7	11	13	17	19	23	29
$[133] = 133$	133^2	$\equiv -184 = -1 \cdot 2^3 \cdot 23$	1	1								1	
$[133, 1] = 134$	134^2	$\equiv 83 =$ није гладак											
$[133, 1, 2] = \frac{401}{3}$	401^2	$\equiv -56 = -1 \cdot 2^3 \cdot 7$	1	1			1						
$[133, 1, 2, 4] = \frac{1738}{13}$	1738^2	$\equiv 107 =$ није гладак											
$[133, \ldots, 2] = \frac{3877}{29}$	3877^2	$\equiv -64 = -1 \cdot 2^6$	1										
$[133, \ldots, 3] = \frac{13369}{100}$	13369^2	$\equiv 161 = 7 \cdot 23$					1					1	

$(133 \cdot 401 \cdot 13369)^2 \equiv (-1 \cdot 2^3 \cdot 7 \cdot 23)^2 \pmod{n}$. Сада је $133 \cdot 401 \cdot 13369 \equiv 1288$ и $-1 \cdot 2^3 \cdot 7 \cdot 23 \equiv 16585$, али је $1288 \equiv -16585$. То није добро. То значи да $\text{nzd}(16585 + 1288, n) = n$ и $\text{nzd}(16585 - 1288, n) = 1$, па нисмо добили ни један фактор. Настављамо даље.

			-1	2	3	5	7	11	13	17	19	23	29
$[133, \ldots, 1] = \frac{17246}{129}$	17246^2	$\equiv -77 = -1 \cdot 7 \cdot 11$	1				1	1					
$[133, \ldots, 2] = \frac{47861}{358}$	47861^2	$\equiv 149 =$ није гладак											
$[133, \ldots, 1] = \frac{65107}{487}$	65107^2	$\equiv -88 = -1 \cdot 2^3 \cdot 11$	1	1				1					

$(401 \cdot 3877 \cdot 17246 \cdot 65107)^2 \equiv ((-1)^2 \cdot 2^6 \cdot 7 \cdot 11)^2 \pmod{n}$. Сада је $401 \cdot 3877 \cdot 17246 \cdot 65107 \equiv 7272$ и $(-1)^2 \cdot 2^6 \cdot 7 \cdot 11 \equiv 4928$. Добија се $7272 - 4928 = 2344$ и $\text{nzd}(2344, n) = 293$, $n/293 = 61$. Оба чиниоца 293, 61 су прости.

24.4 Факторизација помоћу елиптичких кривих

Овај метод (аутор је H. W. Lenstra млађи) је често ефикасан када најмањи прост чинилац n има између 13 и 65 цифара, а следећи најмањи прост чинилац је много већи. Започињемо са мотивишућим примером.

Пример. Искористимо елиптичку криву $y^2 = x^3 + x + 1$ да раставимо на чиниоце 221. Тачка $R = (0, 1)$ очигледно припада кривој. По модулу 221 може се израчунати $2R = (166, 137)$, $3R = (72, 169)$, $4R = (109, 97)$ и $5R = (169, 38)$.

Да се израчуна $6R = R + 5R$ мора се прво одредити нагиб $\frac{38-1}{169-0} = \frac{37}{169} \pmod{221}$. Према томе, потребно нам је $169^{-1} \pmod{221}$. Помоћу Еуклидовог алгоритма добијамо да је $\text{nzd}(221, 169) = 13$. Према томе, $13|221$ и $221/13 = 17$.

Шта се десило иза сцене?

	mod 13	mod 17
R	$(0,1)$	$(0,1)$
$2R$	$(10,7)$	$(13,1)$
$3R$	$(7,0)$	$(4,16)$
$4R$	$(10,6)$	$(9,12)$
$5R$	$(0,12)$	$(16,4)$
$6R$	$\oslash$	$(10,12)$

Приметимо да је $18R = \oslash \pmod{17}$. Успели смо јер је $6R = \oslash$ по модулу 13, а $6R \neq \oslash$ по модулу 17. Приметимо да је по модулу било ког простог броја неки умножак R једнак $\oslash$. Крај примера.

Због једноставности разматраћемо факторизацију $n = pq$, иако метод ради за произвољне целе бројеве. Изаберимо неку елиптичку криву E и неку тачку R на њој по модулу n. Изаберимо затим неки "јако сложен" број, као што је $t!$ (величина t зависи од n), надајући се да је $t!R = \oslash$ по модулу једног простог чиниоца (нпр. p), али не и другог. Нека је k нагиб који се појављује при израчунавању $t!R$. Тада је $\mathrm{nzd}(k,n) = p$, односно добили смо чинилац n.

Зашто $t!$? Постоје неки бројеви m за које је $mR = \oslash \pmod{p}$. Ако $m|t!$ (што је тачно за довољно велико t), онда је $t! = lm$, па је $t!R = lmR = l(mR) = l\oslash = \oslash$.

До неуспеха може да дође из два разлога. Најпре $t!R$ не мора да буде $\oslash$ ни по модулу p, ни по модулу q (као $2!R$ у претходном примеру). Поред тога, може да се деси да је $t!R$ једнако $\oslash$ и по модулу p и по модулу q, па је тада $\mathrm{nzd}(k,n) = n$. У случају неуспеха треба покушати са новим E и R. Са већином других алгоритама за факторизацију такве могућности нису на располагању. Може нпр. се користити фамилија $E : y^2 = x^3 + jx + 1$ и $R = (0,1)$ за разне j.

Пример. Нека је $n = 670726081$, $E : y^2 = x^3 + 1$ и $R = (0,1)$. Тада је $(100!)R = \oslash$. Дошло је дакле до неуспеха друге врсте (приметимо да је ово лош пример, јер је $3R = \oslash$). Искористимо затим $E : y^2 = x^3 + x + 1$ и исто R. Тада је $(100!)R = (260043248, 593016337)$, па је дошло до неуспеха прве врсте. Искористимо сада $E : y^2 = x^3 + 2x + 1$ и исто R. Приликом израчунавања $(100!)R$, дошло је до грешке, јер број 54323 није могао бити инвертован по модулу n. Управо ово је знак успеха, јер је $\mathrm{nzd}(54323, n) = 54323$ и $n/54323 = 12347$.

24.5 Поље бројева

Размотрићемо најпре поља бројева да бисмо могли да разумемо сито у пољу бројева.

Нека $\mathbf{Z}$ означава скуп целих бројева, $\mathbf{Q}$ скуп рационалних бројева (разломке којима је бројилац цели број, а именилац природни број), а $\mathbf{R}$ скуп реалних бројева. Нека је $i = \sqrt{-1}$; $i^2 = -1$, $i^3 = -i$, $i^4 = 1$. Скуп $\mathbf{C} = \{a + bi) \mid a, b \in$

$\mathbf{R}\}$ је скуп комплексних бројева. На пример, број $\pi + ei$ је комплексан. Нека је $f(x) = a_n x^n = a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$, при чему је $a_i \in \mathbf{C}$, $0 \leq i \leq n$. Тада се може написати $f(x) = a_n(x - \alpha_1)(x - \alpha_2)\cdots(x - \alpha_n)$. При томе је скуп $\{\alpha_i \mid 1 \leq i \leq n\}$ једнозначно одређен. Бројеви α_i су корени f. Број α је корен f ако и само ако је $f(\alpha) = 0$.

Скуп $\mathbf{Q}(\alpha)$ је *поље бројева*. Састоји се од свих бројева добијених комбиновањем рационалних бројева и α коришћењем операција $+$, $-$, $\times$, $/$.

Пример. Нека је $f(x) = x^2 - 2 = (x + \sqrt{2})(x - \sqrt{2})$. Нека је $\alpha = \sqrt{2}$. Тада је $\mathbf{Q}(\sqrt{2}) = \{a + b\sqrt{2} \mid a, b \in \mathbf{Q}\}$. Скуп је очигледно затворен за сабирање и одузимање. Множење: $(a+b\sqrt{2})(c+d\sqrt{2}) = (ac+2bd)+(ad+bc)\sqrt{2} \in \mathbf{Q}(\sqrt{2})$. Дељење $(a + b\sqrt{2})/(c + d\sqrt{2})$:

$$\frac{a + b\sqrt{2}}{c + d\sqrt{2}} = \frac{(a + b\sqrt{2})(c - d\sqrt{2})}{(c + d\sqrt{2})(c - d\sqrt{2})} = \frac{ac - 2bd}{c^2 - 2d^2} + \frac{bc - ad}{c^2 - 2d^2}\sqrt{2} \in \mathbf{Q}(\sqrt{2}).$$

Пример. $g(x) = x^3 - 2$, $\alpha = 2^{1/3}$. $\mathbf{Q}(\alpha) = \{a + b \cdot 2^{1/3} + c \cdot 2^{2/3} \mid a, b, c \in \mathbf{Q}\}$. Овај скуп је такође затворен за сабирање, одузимање, множење, и дељење (сем нулом). Дељење је мало компликованије.

Сваки елемент коренског поља је корен неког полинома са целобројним коефицијентима ($a_i \in \mathbf{Z}$), који су као скуп узајамно прости, при чему је најстарији коефицијент позитиван ($a_n > 0$). Полином најмањег степена који задовољава овај услов је *минимални полином* броја α.

Пример. Одредити минимални полином броја $\alpha = 2^{1/3} + 1$. Можемо да искористимо дефиницију α: $(\alpha - 1)^3 = 2$, $\alpha^3 - 3\alpha^2 + 3\alpha - 1 = 2$, $\alpha^3 - 3\alpha^2 + 3\alpha - 3 = 0$. Према томе, минимални полином α је $f(x) = x^3 - 3x^2 + 3x - 3$. Јасно је да је $f(\alpha) = 0$, тј. α је корен f. Поред тога, α није корен ни једног квадратног или линеарног полинома са целобројним коефицијентима.

Ако је најстарији коефицијент минималног полинома α једнак 1, онда је α *алгебарски цели број*. Ово је у складу са уобичајеном дефиницијом за рационалне бројеве. Минимални полином броја 5 је $1x - 5$, а минимални полином броја $3/4$ је $4x - 3$.

Ако у пољу бројева K важи $\alpha = \beta\gamma$, и сва ова три броја су алгебарски цели бројеви, онда кажемо да $\beta|\alpha$. У пољу бројева K се за алгебарски цели број α каже да је *прост* ако из $\alpha|\beta\gamma$ следи $\alpha|\beta$ или $\alpha|\gamma$, где су β, γ алгебарски цели бројеви у K. На жалост, немају сва поља бројева "довољно" простих бројева. То отежава имплементацију сита у пољу бројева.

На пример, поље $\mathbf{Q}(\sqrt{-5})$ је једно од проблематичних поља бројева. Цели бројеви у њему су облика $a + b\sqrt{-5}$, $a, b \in \mathbf{Z}$. У том пољу важе разлагања $6 = (1 + \sqrt{-5})(1 - \sqrt{-5}) = 2 \cdot 3$. Број 2 је несводљив у скупу целих алгебарских бројева: једини начин да се он растави на чиниоце је $2 = 2 \cdot 1 = (-2) \cdot (-1)$. Међутим, 2 није прост. Заиста, $2|(1 + \sqrt{-5})(1 - \sqrt{-5})$, али $2 \not| 1 + \sqrt{-5}$ и $2 \not| 1 - \sqrt{-5}$. Чињеница да на пример $2 \not| 1 + \sqrt{-5}$ следи из тога што је минимални полином броја $(1 + \sqrt{-5})/2$ једнак $2x^2 - 2x + 3$, а тај број није алгебарски цели број. Приметимо такође да у овом пољу факторизација није јединствена.

У скупу $\mathbf{Z}$ факторизација је јединствена (основна теорема аритметике). С друге стране, $14 = 7 \cdot 2 = (-7) \cdot (-2)$. Кажемо да су 7 и -7 спрегнути (придружени) прости бројеви, јер је њихов количник јединица (у овом случају -1, инвертибилни алгебарски цели број).

У наставку ћемо због једноставности радити са пољем бројева $\mathbf{Q}(i) = \{a+bi \mid a, b \in \mathbf{Q}\}$. Алгебарски цели бројеви у $Q(i)$ су $\mathbf{Q}(i) = \{a+bi \mid a, b \in \mathbf{Z}\}$. Овај скуп означава се обично са $\mathbf{Z}(i)$. Ово је поље бројева које се добро понаша. Његове јединице су $\{\pm 1, \pm i\}$. Ако је $p \in \mathbf{Z}_{>0}$ прост број и $p \equiv 3 \pmod 4$, онда је p прост и у $\mathbf{Z}(i)$. Ако је пак $p \equiv 1 \pmod 4$, онда се p може представити у облику $p = a^2 + b^2$, $a, b \in \mathbf{Z}$, па је $p = (a+bi)(a-bi)$, при чему су $a+bi$, $a-bi$ непридружени прости бројеви. Другим речима, p се разлаже у производ два "мања" проста броја. На пример, $17 = 4^2 + 1^2 = (4+i)(4-i)$, и такође $17 = (1+4i)(1-4i)$. Ово је у реду, јер је $(1-4i)i = 4+i$, а i је јединица, па су $1-4i$ и $4+i$ придружени. За ову чињеницу може се употребити ознака $1-4i \sim 4+i$. Број i је јединица јер је његов минимални полином x^2+1, па је алгебарски цели број, и $i \cdot i^3 = 1$, па $i \mid 1$. Уствари, $1-4i \sim 4+i \sim -1+4i \sim -4-i$ и $1+4i \sim 4-i \sim -1-4i \sim -4+i$ (јер су ± 1, $\pm i$ јединице). С друге стране, ни један број из прве групе није придружен ни једном из друге групе. Између придружених бројева увек ћемо бирати представника облика $a \pm bi$ за кога важи $a \geq b \geq 0$.

Број 2 није прост, јер је $2 = (1+i)(1-i)$, при чему је $(1-i)i = 1+i$, односно $1-i \sim 1+i$. Дакле, 2 је дељиво простим бројем $1+i$.

Погледајмо неке просте бројеве у $\mathbf{Z}$ и њихове факторизације у $\mathbf{Z}[i]$: $2 = (1+i)^2 i^3$, $3 = 3$, $5 = (2+i)(2-i)$, $7 = 7$, $11 = 11$, $13 = (3+2i)(3-2i)$, $17 = (4+i)(4-i)$, $19 = 19$, $23 = 23$, $29 = (5+2i)(5-2i)$, $31 = 31$, $37 = (6+i)(6-i)$.

Пресликавање $N : \mathbf{Q}(i) \to \mathbf{Q}$ дефинисано са $N(a+bi) = a^2 + b^2$ зове се норма. На пример, $N(2+i) = 5$, $N(7) = 49$. Ако је $a+bi \in \mathbf{Z}[i]$, p је прост у $\mathbf{Z}$ и $p | N(a+bi)$ (тј. $p | a^2+b^2$) онда неки прости делилац p дели $a+bi$. Ово олакшава факторизацију алгебарских целих бројева у $\mathbf{Z}[i]$.

Факторизација $5+i$. Пошто је $N(5+i) = 26$, сви фактори су у скупу $\{i, 1+i, 3+2i, 3-2i\}$. Даље, $3+2i | 5+i$ ако је $(5+i)/(3+2i)$ цели број.

$$\frac{5+i}{3+2i}\left(\frac{3-2i}{3-2i}\right) = \frac{17}{13} + \frac{-7}{13}i$$

па $3+2i \not| 5+i$.

Факторизација $7+i$. Најпре је $N(7+i) = 50 = 2 \cdot 5^2$. Даље, $(7+i)/(2+i) = 3-i$ и $N(3-i) = 10$; $(3-i)/(2+i) = (1-i)$ и $N(1-i) = 2$; $(1-i)/(1+i) = -i = i^3$, па је $7+i = i^3(1+i)(2+i)^2$.

Следећа чињеница је такође корисна за факторизацију у $\mathbf{Z}(i)$. Ако је $a+bi \in \mathbf{Z}(i)$ и $\mathrm{nzd}(a,b) = 1$ и $N(a+bi) = p_1^{\alpha_1} \cdots p_r^{\alpha_r}$ где су p_i позитивни прости бројеви, онда $p_i \not\equiv 3 \pmod 4$ и $a+bi = i^{\alpha_0} \pi_1^{\alpha_1} \cdots \pi_r^{\alpha_r}$, где је π_i неки од простих делилаца p_i. При томе се у овом изразу никад не појављују истовремено два проста чиниоца p_i.

У последњем примеру $N(7+i) = 2^1 5^2$, па знамо да је $7+i = i^{\alpha_0}(1+i)^1(2 \pm i)^2$. Довољно је да одредимо α_0 и знак у $\pm$. Други пример. $N(17-6i) = 325 = 5^2 \cdot 13$, па је $7-6i = i^{\alpha_0}(2 \pm i)^2(3 \pm 2i)^2$, при чему се $\pm$ не морају поклапати.

Ако су α и β елементи $\mathbf{Q}(i)$, онда је $N(\alpha\beta) = N(\alpha)N(\beta)$.

24.6 Сито у пољу бројева

Сито у пољу бројева (Pollard, Adleman, H. Lenstra) је тренутно најбољи познати алгоритам за факторизацију бројева $n > 10^{130}$, при чему је најмањи прост чинилац n бар 10^{65}. RSA бројеви су овог типа. Број RSA –193 (који има 193 цифре) растављен је на чиниоце применом овог алгоритма.

Најпре се бира степен d (он зависи од n, $d \approx \sqrt{\ln n / \ln\ln(n)}$). Нека је $m = \lfloor \sqrt[d]{n} \rfloor$; n се разлаже у систему са основом m. Дакле, $n = m^d + a_{d-1}m^{d-1} + \cdots + a_0$, при чему је $0 \leq a_i < m$. Нека је $f(x) = x^d + a_{d-1}x^{d-1} + \cdots + a_0$. Нека је α корен f. Ради се у пољу $Q(\alpha)$.

Нека је потребно раставити на чиниоце број 2501. Пошто је $\sqrt{\ln n / \ln\ln(n)} \approx 1.95$), нека је $d = 2$. Пошто је $\lfloor \sqrt[d]{n} \rfloor = 50$ и $2501 = 50^2 + 1$, биће $f(x) = x^2 + 1$; корен f је i. Приметимо да 50 делује као i у $\mathbf{Z}/2501\mathbf{Z}$ јер је $50^2 \equiv -1 \pmod{2501}$. Дефинишемо пресликавања $h : \mathbf{Z}[i] \to \mathbf{Z}$, $h(a + bi) = a + b50$, и $\hat{h} : \mathbf{Z}[i] \to \mathbf{Z}_{2501}$, $\hat{h}(a + bi) = a + b50 \pmod{2501}$. Пресликавање $\hat{h}$ има особине $\hat{h}(\alpha + \beta) = \hat{h}(\alpha) + \hat{h}(\beta)$ и $\hat{h}(\alpha\beta) = \hat{h}(\alpha)\hat{h}(\beta)$. Пресликавање h има прву, али не и другу од ових особина.

Следећи корак је тражење бројева облика $\alpha = a+bi$, $a, b \in \mathbf{Z}$, $a \geq 0, b \neq 0$, $\mathrm{nzd}(a, b) = 1$, при чему је $a + bi$ гладак у $\mathbf{Z}[i]$, а $h(a + bi)$ гладак у $\mathbf{Z}$. Ово је као тражење игле у пласту сена, због чега је факторизација још увек тешка. Потребно је наћи $\alpha_1, \ldots, \alpha_r$, такве да је $\alpha_1\alpha_2 \ldots \alpha_r = \beta^2$ у $\mathbf{Z}[i]$ и $h(\alpha_1)h(\alpha_2) \ldots h(\alpha_r) = t^2$ у $\mathbf{Z}$. Тада је $h(\beta)^2 = h(\beta)h(\beta) \equiv \hat{h}(\beta)\hat{h}(\beta) \equiv \hat{h}(\beta^2) \equiv \hat{h}(\alpha_1\alpha_2 \ldots \alpha_r) = \hat{h}(\alpha_1) \ldots \hat{h}(\alpha_r) = t^2$. Посматрајмо остатке целих бројева $h(\beta)$ и t по модулу n. Из $h(\beta)^2 \equiv t^2 \pmod{n}$ ако је $h(\beta) \not\equiv \pm t \pmod{n}$, следи да је $\mathrm{nzd}(h(\beta) + t, n)$ нетривијални чинилац n.

Раставимо сада 2501 помоћу сита у пољу бројева. Користићемо базу фактора $i, 1+i, 2\pm i, 3\pm 2i, 4\pm i, 5\pm 2i$ за алгебарске целе бројеве. Ови бројеви деле целе бројеве $1, 2, 5, 13, 17, 29$. За целе бројеве користићемо базу фактора $-1, 2, 3, 5, 7, 11, 13, 17, 19, 23, 29$. Функција h дефинисана је са $h(a + bi) =$

$a + b \cdot 50$.

α	факторизација α	$h(\alpha)$	факторизација $h(\alpha)$	
i	$= i$	50	$= 2 \cdot 5^2$	
$1+i$	$= 1+i$	51	$= 3 \cdot 17$	
$2+i$	$= 2+i$	52	$= 2^2 \cdot 13$	$*$
$4+i$	$= 4+i$	54	$= 2 \cdot 3^3$	$*$
$7+i$	$= i^3(1+i)(2+i)^2$	57	$= 3 \cdot 19$	$*$
$1-i$	$= i^3(1+i)$	-49	$= -1 \cdot 7^2$	$*$
$2-i$	$= 2-i$	-48	$= -1 \cdot 2^4 \cdot 3$	
$4-i$	$= 4-i$	-46	$= -1 \cdot 2 \cdot 23$	$*$
$5-i$	$= i^3(3+2i)(1+i)$	-45	$= -1 \cdot 3^2 \cdot 5$	
$8-i$	$= (3-2i)(2+i)$	-42	$= -1 \cdot 2 \cdot 3 \cdot 7$	
$5+2i$	$= 5+2i$	105	$= 3 \cdot 5 \cdot 7$	
$1-2i$	$= i^3(2+i)$	-99	$= -1 \cdot 3^2 \cdot 11$	$*$
$9-2i$	$= (4+i)(2-i)$	-91	$= -1 \cdot 7 \cdot 13$	
$5-2i$	$= 5-2i$	-95	$= -1 \cdot 5 \cdot 19$	
$5-3i$	$= i^3(1+i)(4+i)$	-145	$= -1 \cdot 5 \cdot 29$	
$3+4i$	$= (2+i)^2$	203	$= 7 \cdot 29-$	
$2+5i$	$= i(5-2i)$	252	$= 2^2 \cdot 3^2 \cdot 7$	
$3+5i$	$= (4+i)(1+i)$	253	$= 11 \cdot 23$	$*$
$3-5i$	$= i^3(1+i)(4-i)$	-247	$= -1 \cdot 13 \cdot 19$	$*$

Овакве вредности α смештају се као вектори по модулу два. Тако на пример последњој од њих одговара вектор

$$3 - 5i \sim (1,1,0,0,0,0,1,0,0,1,0,0,0,0,0,1,0,1,0,0)$$

што одговара чиниоцима редом

$$(i, 1+i, 2+i, 2-i, 3+2i, 3-2i, 4+i, 4-i, 5+2i, 5-2i, -1, 2, 3, 5, 7, 11, 13, 17, 19, 23, 29).$$

Решавањем система линеарних једначина одређују се обећавајуће комбинације чинилаца. У овом примеру, ако се саберу сви вектори обележени звездицом, добија се нула вектор. Дакле, множењем одговарајућих α добија се квадрат у скупу алгебарских целих, а множењем одговарајућих $h(\alpha)$ добија се квадрат целог броја. Производ звездицама означених алгебарских целих је $i^{12}(1+i)^4(2+i)^4(4+i)^2(4-i)^2$ а производ одговарајућих целих бројева је $(-1)^4 \cdot 2^4 \cdot 3^6 \cdot 7^2 \cdot 11^2 \cdot 13^2 \cdot 19^2 \cdot 23^2$.

Нека је

$$\beta = i^6(1+i)^2(2+i)^2(4+i)(4-i) = 136 - 102i.$$

Дакле, $h(\beta) = 136 - 102 \cdot 50 = -4964 \equiv 38 \pmod{2501}$.

Нека је

$$t = (-1)^2 \cdot 2^2 \cdot 3^3 \cdot 7 \cdot 11 \cdot 13 \cdot 19 \cdot 23 = 47243096 \equiv 1807 \pmod{2501}.$$

Према томе $38^2 \equiv 1444 \equiv 1807^2 \pmod{2501}$). Из $\text{nzd}(1807 - 38, 2501) = 61$ и $2501/61 = 41$ следи $2501 = 61 \cdot 41$.

Реч *cumo* односи се на поступак бирања само таквих α који имају већу вероватноћу да буду глатки, а да при томе и $h(\alpha)$ буде гладак. На пример, број n, $1 \leq n \leq 1000$, дељив са 36 вероватније ће бити гладак од произвољног броја из тог интервала. Могу се одредити услови које остаци a и b по модулу 36 треба да задовољавају да би се гарантовало да $h(a + bi)$ буде умножак 36.

25 Решавање проблема дискретног логаритма у $\mathbf{F}_q^*$

Размотрићемо два метода за решавање ФФДЛП. Први је метод Полига-Хелмана, који је ефикасан ако је величина $\mathbf{F}_q^*$ гладак број. Метод сличан овом може се применити за решавање ЕЦДЛП. Други метод је метод израчунавања индекса, за који се не зна варијанта која би решавала ЕЦДЛП. Методи се заснивају на кинеској теореми о остацима.

25.1 Дигресија: употреба кинеске теореме о остацима за дешифровање RSA

Нека је $n = pq$. Претпоставимо да је $M^e \equiv C \pmod{n}$. Тада је дешифровање израчунавање $C^d \equiv M \pmod{n}$, за шта је потребно време $O(\log^3(n))$. Нека је $M \equiv M_p \pmod{p}$, $M \equiv M_q \pmod{q}$, $C \equiv C_p \pmod{p}$, $C \equiv C_q \pmod{q}$, $d \equiv d_p \pmod{p-1}$, $d \equiv d_q \pmod{q-1}$. Сви бројеви M_p, M_q, C_p, C_q, d_p, d_q су мањи од $2\sqrt{n}$. Бројеви d_p и d_q могу се унапред израчунати. Израчунавање остатака C_p и C_q има сложеност $O(\log^2(n^{1/2}))$, што није много. Израчунавања $C_p^{d_p} \equiv M_p \pmod{p}$ и $C_q^{d_q} \equiv M_q \pmod{q}$ имају сложеност $O(log^3(n^{1/2})$. Према томе, свако од ових израчунавања узима 1/8 времена потребног за израчунавање $C^d \pmod{n}$. Одређивање M полазећи од $M_p \pmod{p}$ и $M_q \pmod{q}$ помоћу кинеске теореме о остацима захтева време $O(log^3(n^{1/2}))$, али је спорије од израчунавања $C_p^{d_p} \equiv M_p \pmod{p}$. У пракси, два поновљена квадрирања по модулима p и q и употреба кинеске теореме о остацима трају око два пута мање од израчунавања $C^d \equiv M \pmod{n}$.

25.2 Полиг-Хелманов алгоритам

Овај алгоритам за решавање проблема дискретног логаритма у $\mathbf{F}_q^*$, где је q прост број или степен простог броја, ефикасан је само ако је број $q-1$ (ред мултипликативне групе поља) гладак. Зато се у системима који користе проблем дискретног логаритма захтева да q буде изабрано тако да $q-1$ има бар један велики прост чинилац. Најпре ће бити изложен алгоритам, па пример рада алгоритма и објашњење зашто он ради.

Нека је g генератор $\mathbf{F}_q^*$. За дато $y \in \mathbf{F}_q^*$ потребно је решити $g^x = y$ по x. Нека је $q-1 = p_1^{\alpha_1} \cdots p_r^{\alpha_r}$, где су p_i прости бројеви. За сваки од бројева p_i решава се једначина $z^{p_i} = 1$ (при чему је $z \neq 1$) у $\mathbf{F}_q^*$. Таква решења зову се *примитивни p_i-ти корени из јединице* и означавају се са ζ_{p_i}. Пошто g генерише $\mathbf{F}_q^*$, видимо да је $\zeta_{p_i} = g^{(q-1)/p_i}$ један од примитивних p_i-тих корена из јединице. За сваки број p_i унапред се израчунавају сва решења једначине $z^{p_i} = 1$. То су $\zeta_{p_i}^0$, $\zeta_{p_i}^1$, $\ldots$, $\zeta_{p_i}^{p_i-1}$. Ове вредности се памте заједно са одговарајућим експонентима ζ_{p_i}.

Подсетимо се да се у $\mathbf{F}_q^*$ са експонентима рачуна по модулу $q-1$. Зато, када се одреде остаци $x \pmod{p_i^{\alpha_i}}$, x се може одредити помоћу кинеске теореме о остацима. Дакле, остаје да се одреди $x \pmod{p^\alpha}$ (због једноставности су испуштени индекси). Претпоставимо да смо остатак x по модулу p^α написали у систему са основом p. То још увек не знамо да урадимо, али развој по степенима p постоји: $x \equiv x_0 + x_1 p + x_2 p^2 + \cdots x_{\alpha-1} p^{\alpha-1} \pmod{p^\alpha}$, $0 \leq x_i < p$. Прелазимо на одређивање коефицијената x_i.

Израчунавамо $y^{(q-1)/p} \pmod q$. Тај број је на списку степенова ζ_p^i. На основу једнакости $y^{(q-1)/p} \equiv \zeta_p^{x_0} \pmod q$ добијамо вредност x_0.

Нека је $y_1 \equiv y/(g^{x_0}) \pmod q$. Одредивши $y_1^{(q-1)/p^2} \equiv \zeta_p^{x_1} \pmod q$ добијамо x_1.

Нека је $y_2 \equiv y/(g^{x_0+x_1 p})$. Одредивши $y_2^{(q-1)/p^3} \equiv \zeta_p^{x_2}$ добијамо x_2.

Нека је $y_3 \equiv y/(g^{x_0+x_1 p+x_2 p^2})$. Одредивши $y_3^{(q-1)/p^4} \equiv \zeta_p^{x_3}$ добијамо x_3, итд.

Пример. Нека је $q = 401$, $g = 3$ и $y = 304$. Потребно је решити $3^x = 304 \pmod{401}$. Разлагање $q-1$ је $q-1 = 2^4 \cdot 5^2$. Најпре одређујемо $x \pmod{16}$. Почињемо израчунавањем $g^{400/2} = \zeta_2 = \zeta_2^1 = 400$ и $\zeta_2^2 = \zeta_2^0 = 1$ (рачуна се по модулу 401). У разлагању $x = x_0 + x_1 2 + x_2 4 + x_3 8 \pmod{16}$ потребно је одредити коефицијенте x_i.

$$y^{(q-1)/p} = 304^{400/2} \equiv 400 \pmod{401} = \zeta_2^1 \quad \text{па је } x_0 = 1.$$

$$\frac{y}{g^{x_0}} = 304/3^1 \equiv 235 \pmod{401} = y_1. \qquad y_1^{(q-1)/p^2} = 235^{400/(2^2)} \equiv 400 = \zeta_2^1 \quad \text{па је } x_1 = 1.$$

$$\frac{y}{g^{x_0+x_1 p}} = 304/(3^{1+1\cdot 2}) \equiv 338 = y_2. \qquad y_2^{(q-1)/p^3} = 338^{400/(2^3)} \equiv 1 = \zeta_2^0 \quad \text{па је } x_2 = 0.$$

$$\frac{y}{g^{x_0+x_1 p+x_2 p^2}} = 304/(3^{1+1\cdot 2+0\cdot 4}) \equiv 338 = y_3. \qquad y_3^{(q-1)/p^4} = 338^{400/(2^4)} \equiv 400 = \zeta_2^1 \quad \text{па је } x_3 = 1.$$

Прелазимо на одређивање $x \pmod{25}$. Најпре израчунавамо $g^{400/5} = \zeta_5 = 72$, $\zeta_5^2 = 372$, $\zeta_5^3 = 318$, $\zeta_5^4 = 39$, и $\zeta_5^5 = \zeta_5^0 = 1$. Треба одредити коефицијенте у разлагању $x = x_0 + x_1 5 \pmod{25}$.

$$y^{(q-1)/p} = 304^{400/5} \equiv 372 \pmod{401} = \zeta_5^2 \quad \text{па је } x_0 = 2.$$

$$\frac{y}{g^{x_0}} = 304/3^2 \equiv 212 = y_1. \qquad y_1^{(q-1)/p^2} = 212^{400/(5^2)} \equiv 318 = \zeta_5^3 \quad \text{па је } x_1 = 3.$$

Према томе $x \equiv 2 + 3 \cdot 5 = 17 \pmod{25}$. Из $x \equiv 11 \pmod{16}$ и $x \equiv 17 \pmod{25}$ на основу кинеске теореме о остацима следи $x = 267$. Дакле $3^{267} = 304 \pmod{401}$.

Зашто ово ради? Размотримо један једноставнији пример. Нека је $q = 17$, $g = 3$. Тада је $q - 1 = 16 = 2^4$. $3^{1\cdot 8} = 16 = \zeta_2^1$ и $3^{0\cdot 8} = 1 = \zeta_2^0$.

Приметимо да је $m^{16} \equiv 1 \pmod{17}$ за свако $1 \leq m \leq 16$. У једнакости $3^{11} \equiv 7 \pmod{17}$ претпоставимо да је 11 непознато. Дакле $3^{1+1\cdot 2+0\cdot 4+1\cdot 8} \equiv 7 \pmod{17}$.

знамо		исто, али не знамо експонент	
7,	$7^8 \equiv 16 = (3^8)^1$	$3^{1+1\cdot 2+0\cdot 4+1\cdot 8}$,	$3^{1\cdot 8+16n} = 3^{1\cdot 8}(3^{16})^n = (3^8)^1$
$\frac{7}{3^1}$,	$\left(\frac{7}{3^1}\right)^4 = 16 = (3^8)^1$	$3^{1\cdot 2+0\cdot 4+1\cdot 8}$,	$3^{1\cdot 8+16n} = (3^8)^1$
$\frac{7}{3^{1+1\cdot 2}}$,	$\left(\frac{7}{3^{1+1\cdot 2}}\right)^2 = 1 = (3^8)^0$	$3^{0\cdot 4+1\cdot 8}$,	$3^{0\cdot 8+16n} = (3^8)^0$
$\frac{7}{3^{1+1\cdot 2+0\cdot 4}}$,	$\left(\frac{7}{3^{1+1\cdot 2+0\cdot 4}}\right)^1 = 16 = (3^8)^1$	$3^{1\cdot 8}$	$= (3^8)^1$

25.3 Алгоритам за израчунавање индекса

Алгоритам за израчунавање индекса је метод за решавање проблема дискретног логаритма у пољима типа $\mathbf{F}_q$, где је q прост број или степен простог броја. Ако је p прост број и $p \approx 2^r$, и користи се алгоритам за израчунавање индекса, онда је лакше решити проблем дискретног логаритма у $\mathbf{F}_{2^r}$ него у $\mathbf{F}_p$. Међутим, лакше је имплементирати шифарске системе над $\mathbf{F}_{2^r}$, па рад са оваквим пољима остаје популаран. Због тога ћемо демонстрирати овај алгоритам над таквим пољима.

Подсетимо се да је $\mathbf{F}_2[x]/(x^3+x+1) = \{a_0+a_1x+a_2x_2 \mid a_i \in \mathbf{F}_2\}$. За ово поље користимо ознаку $\mathbf{F}_8$. У овом пољу је $2 = 0$ и $x^3+x+1 = 0$, односно $x^3 = -x-1 = x+1$. $\mathbf{F}_8^*$ је $\mathbf{F}_8$ без 0. Скуп $\mathbf{F}_8^*$ има $8-1 = 7$ елемената и они су генерисани елементом x. Тако је $x^1 = 1$, $x^2 = x^2$, $x^3 = x+1$, $x^4 = x^2+x$, $x^5 = x^2+x+1$, $x^6 = x^2+1$ и $x^7 = 1$. Приметимо да је $x^{12} = x^7 \cdot x^5 = x^5$, јер се са експонентима рачуна по модулу 7 (што је број елемената у $\mathbf{F}_8^*$).

Подсетимо се, $\log_b m = a$ значи $b^a = m$, па је $\log_x(x^2+x+1) = 5$ јер је $x^5 = x^2+x+1$. Основу x логаритма обично нећемо писати. Логаритми дају експоненте, па логаритми раде по модулу 7. Приметимо да из $(x^2+1)(x+1) = x^2$ следи $\log(x^2+1)(x+1) = \log(x^2+1)+\log(x+1) = 6+3 = 9$, док је $\log(x^2) = 2$; ово међутим није контрадикција, јер је $9 \equiv 2 \pmod{7}$.

Размотримо општи случај. Нека је $f(x)$ несводљив (над $\mathbf{F}_2$) полином степена d. Тада је $\mathbf{F}_q = \mathbf{F}_2[x]/(f(x))$, где је $q = 2^d$. Претпоставимо да је g генератор $\mathbf{F}_q^*$. Ако је $g^n = y$ онда кажемо да је $\log_g y = n$, односно $\log y = n$. Важе идентитети $\log(uv) \equiv \log(u)+\log(v) \pmod{q-1}$ и $\log(u^r) \equiv r\log(u) \pmod{q-1}$.

Формулација проблема дискретног логаритма у $\mathbf{F}_q$ је следећа. Нека је $g^n = y$. Ако је дато g и y, одредити остатак $n \pmod{q-1}$, односно одредити $n = \log_g y$. Приметимо да је $\log_g g = 1$.

Припремни корак за алгоритам израчунавања индекса је избор броја m, $1 < m < d$ (како се он бира — компликовано објашњење се заснива на теорији бројева и статистици; за $d = 127$ бира се $m = 17$).

Корак 1. Нека је $h_1, \ldots, h_r$ скуп несводљивих полинома у $\mathbf{F}_2[x]$ степена $\leq m$. Одредити логаритме сваког елемента $\{h_i\}$. У том циљу израчунати степене g^t у нади да је $g^t = h_1^{\alpha_1}h_2^{\alpha_2}\cdot\cdots\cdot h_r^{\alpha_r}$ (неки од експонената α_i могу да буду 0). Другим речима, желимо да g^t буде m-гладак. Логаритмовањем се

добија $t = \alpha_1 \log(h_1) + \cdots + \alpha_r \log(h_r)$. То је систем линеарних једначина по $\log(h_i)$ (то су једине непознате). Одредити више оваквих линеарних једначина све дотле док не буде могуће одредити све вредности $\log(h_i)$. Кад се то заврши, знају се сви логаритми $a_i = \log(h_i)$.

Корак 2. Израчунати yg^t за разне вредности t све док се не добије $yg^t = h_1^{\beta_1} \cdot \cdots \cdot h_r^{\beta_r}$. Тада је $\log y + t \log g = \beta_1 \log h_1 + \cdots + \beta_r \log h_r$, односно $\log y + t = \beta_1 a_1 + \cdots + \beta_r a_r$. Овде је непознато само $\log y$. Када се ради у коначним пољима често се уместо log користи ознака ind.

Пример. Нека је $f(x) = x^{11} + x^4 + x^2 + x + 1$. Овај полином је несводљив по модулу 2 Рачунамо у пољу $\mathbf{F}_2[x]/(f(x)) = \mathbf{F}_q$, где је $q = 2^{11}$. Елеменат $g = x$ је генератор $\mathbf{F}_q^*$. Бирамо $m = 4$. Желимо да решимо $g^n = y = x^9 + x^8 + x^6 + x^5 + x^3 + x^2 + 1$ по n, односно да одредимо $\log(y)$. Први корак нема никакве везе са y. Нека је

$$\begin{array}{llll} 1 = \log(x) & a = \log(x+1) & c = \log(x^2+x+1) & d = \log(x^3+x+1) \\ e = \log(x^3+x^2+1) & h = \log(x^4+x+1) & j = \log(x^4+x^3+1) & k = \log(x^4+x^3+x^2+x+1). \end{array}$$

Израчунавајући разне степенове g^t добијамо

$$\begin{array}{ll} g^{11} = (x+1)(x^3+x^2+1) & 11 = a + e \pmod{2047 = q-1} \\ g^{41} = (x^3+x^2+1)(x^3+x+1)^2 & 41 = e + 2d \\ g^{56} = (x^2+x+1)(x^3+x+1)(x^3+x^2+1) & 56 = c + d + e \\ g^{59} = (x+1)(x^4+x^3+x^2+x+1)^2 & 59 = a + 2k \\ g^{71} = (x^3+x^2+1)(x^2+x+1)^2 & 71 = e + 2c \end{array}$$

Приметимо да иако имамо четири једначине по a, c, d, e (прва, друга, трећа и пета), пета једначина је једнака два пута помноженој трећој од које је одузета друга, па је сувишна. Због тога настављамо са тражењем једначина.

$$g^{83} = (x^3+x+1)(x+1)^2, \quad 83 = d + 2a.$$

Сада су прва, друга, трећа и најновија четири независне једначине (по модулу 2047) са четири непознате. Другим речима, 4×4 матрица система је инвертибилна по модулу 2047. Решавањем система добијамо $a = 846$, $c = 453$, $d = 438$, $e = 1212$. Затим се може одредити k: $k = (59 - a)/2 \pmod{q-1} = 630$. Затим одређујемо h и j. Тражимо само једначине у које улазе ове две непознате

$g^{106} = (x+1)(x^4+x^3+1)(x^4+x^3+x^2+x+1)$, па је $106 = a+j+k$ и $j = 677$.

$g^{126} = (x^4+x+1)(x^4+x^3+x^2+x+1)(x+1)2$ па је $126 = h+k+2a$ и $h = 1898$.

Дакле $a = 846$, $c = 453$, $d = 438$, $e = 1212$, $h = 1898$, $j = 677$, $k = 630$. Сада прелазимо на други корак. Израчунавамо yg^t за разне вредности t. Проналазимо да је $yg^{19} = (x^4+x^3+x^2+x+1)^2$. Дакле $\log(y)+19\log(g) = 2k$. Пошто је $\log(g) = \log(x) = 1$, добија се $\log(y) = 2k - 19 \equiv 1241 \pmod{2047}$, односно $x^{1241} = y$.

Сито у пољу бројева може се комбиновати са алгоритмом за израчунавање индекса. Тако се долази до оцене да је временска сложеност решавања ФФДЛП у суштини иста као и сложеност факторизације.

26 Задаци

1. Доказати да $6|m(m^2+5)$, $30|m^5-m$, $30|mn(m^4-n^4)$, $42|m^7-m$ важи за произвољне природне бројеве m, n.

2. Доказати да је за $m \in N$ производ $(m+1)(m+2)\cdot s(m+m)$ дељив са 2^m.

3. Нека је $\mathrm{nzd}(a,b)=1$. Доказати да је $\mathrm{nzd}(a+b,a-b)\leq 2$.

4. Израчунати НЗД $\mathrm{nzd}(549,387)$, $\mathrm{nzd}(589,343)$, $\mathrm{nzd}(12606,6494)$, $\mathrm{nzd}(6188,4709)$, а затим изразити НЗД као линеарну комбинацију тих бројева.

5. Одредити $160^{-1} \pmod{841}$.

6. Колико чинилаца има број 945?

7. Одредити $2^{1000000} \mod 7$.

8. Одредити сва решења конгруенције а) $3x \equiv 4 \pmod 7$; б) $3x \equiv 4 \pmod{12}$; в) $9x \equiv 12 \pmod{21}$; г) $27x \equiv 25 \pmod{256}$; д) $27x \equiv 72 \pmod{900}$; ђ) $103x \equiv 612 \pmod{676}$;

9. Одредити $\varphi(n)$ за $n = 90, 91, \ldots, 100$.

10. Доказати: број m је прост ако и само ако је $\varphi(m) = m-1$.

11. Раставити на просте чиниоце бројеве 82798848, 81057226635.

12. Раставити на просте чиниоце бројеве 10!, 15!, 20!, 30!.

13. Са колико нула се завршавају бројеви 50!, 100!?

14. Одредити $\varphi(n)$ за $n = 375, 720, 957, 988, 1200, 4320$.

15. Колико има бројева од 1 до 120 који нису узајамно прости са 30?

16. Зна се да је $\varphi(a) = 120$ и да је $a = pq$, где су p, q прости бројеви. Одредити a, ако је $p-q=2$.

17. Доказати да збир квадрата пет узастопних целих бројева не може бити потпуни квадрат.

18. Одредити сва целобројна решења једначина $53x+47y=1$, $22x+32y=18$.

19. Ако је $\mathrm{nzd}(a,n)=1$, доказати да је $a^{-1} \equiv a^{\varphi(n)-2} \pmod n$.

20. Направити табелу индекса по модулу 29 са основом 2 и табелу индекса по модулу 23 са основом 3.

21. Решити једначине $52^x \equiv 38 \pmod{29}$, $23^x \equiv 9 \pmod{29}$, $3^x \equiv 7 \pmod{23}$, $3^x \equiv 6 \pmod{23}$.

22. Одредити све генераторе $\mathbf{Z}_n^*$ за $n = 23, 29$.

23. Израчунати $5^{-1} \pmod{29}$, $7^{-1} \pmod{29}$, $21^{-1} \pmod{29}$, $39^{-1} \pmod{23}$, $(-1)^{-1} \pmod{23}$.

24. Одредити све несводљиве полиноме степена ≤ 3 у $\mathbf{F}_2[x]$.

25. Да ли је несводљив полином $x^4 + x^3 + x^2 + x + 1$ у $\mathbf{F}_2[x]$?

26. Направити таблицу множења у а) $\mathbf{F}_2[x]/(x^2 + x + 1)^*$; б) $\mathbf{F}_2[x]/(x^3 + x^2 + 1)^*$.

27. Помоћу Еуклидовог алгоритма одредити $(x^4)^{-1}$ у пољу $\mathbf{F}_2[x]/(x^5 + x^2 + 1)$.

28. Одредити матрице A, B, такве да се друга компонента пресликавања S у алгоритму §AES може представити у облику $AY + B$, где је Y вектор–колона — нибл добијен из прве компоненте S (инверзије).

29. Представити проширивање кључа у SAES (односно AES — већим дијаграмом) дијаграмом са шест чворова, три реда по два чвора, тако да у i-том реду буду чворови $W[2i]$, $W[2i + 1]$, $i = 0, 1, 2$.

30. Ако се погрешно пренесе један бит шифрата, колико ће то изазвати грешака у дешифрованој поруци ако се користи

    а) синхрона проточна шифра;

    а) самосинхронишућа проточна шифра код које c_i зависи од p_{i-k}, $p_{i-k+1}, \ldots, p_i$?

31. Колики највећи период може имати излазни низ из алгоритма RC4 (одговор треба да зависи од параметра n)?

32. Користећи чињеницу да функције MC у SAES трансформише колону $\begin{array}{|c|}\hline b_0b_1b_2b_3 \\ \hline b_4b_5b_6b_7 \\ \hline \end{array}$ у колону $\begin{array}{|cccc|}\hline b_0 \oplus b_6 & b_1 \oplus b_4 \oplus b_7 & b_2 \oplus b_4 \oplus b_5 & b_3 \oplus b_5 \\ \hline b_2 \oplus b_4 & b_0 \oplus b_3 \oplus b_5 & b_0 \oplus b_1 \oplus b_6 & b_1 \oplus b_7 \\ \hline \end{array} = \begin{array}{|c|}\hline b'_0b'_1b'_2b'_3 \\ \hline b'_4b'_5b'_6b'_7 \\ \hline \end{array}$
    Одредити матрицу A која вектор $b = [b_0\ b_1\ \ldots b_7]^T$ пресликава у $b' = [b'_0\ b'_1\ \ldots b'_7]^T = Ab$

33. Од колико најмање бита стања алгоритма AES зависи неки бит стања после

    а) функције $ByteSub$;

    б) функције $ShiftRow$;

    в) функције $MixColumn$;

    г) функције $AddRoundKey$;

    д) једне рунде?

34. Колико бита је погрешно после дешифровања поруке ако је шифровање извршено алгоритмом AES у режиму ECB, CBC, CFB, OFB?

35. Користећи табелу која описује функцију S у SAES написати изразе за четири излазна бита e, f, g, h преко улазних бита a, b, c, d.

36. Израчунати $57^{1616} \pmod{97}$.

37. Оценити сложеност израчунавања B^N; факторизације N дељењем са бројевима мањим од $\sqrt{N}$.

38. Направити таблицу степенова, односно логаритама, у $\mathbf{F}_2[x]/(x^3+x^2+1)^*$ ако је генератор x. Користећи ове табеле израчунати $(x^2+1)(x^2+x+1)$.

39. Нека је n производ различитих простих бројева. Нека су бројеви $d, e \in N$ такви да за сваки прост $p|n$ важи $p-1|de-1$. Доказати да је $a^{de} \equiv a \pmod{n}$ за свако a, чак и ако је $\mathrm{nzd}(a, n) > 1$.

40. Доказати да у пољу $\mathbf{F}_q$, ако је $q = p^k$, p – прост број, важи $(a+b)^p = a^p + b^p$.

41. На елиптичкој кривој $y^2 = x^3 - 36^2$ нека је $P = (-3, 9)$ и $Q = (-2, 8)$. Одредити $P + Q$ и $2P$.

42. Колико решења има једначина $x^m \equiv 1 \pmod{n}$ за различите вредности m, $m < n$?

43. Ако је корисник грешком у систему RSA изабрао $n = p$ (p – прост), доказати да је систем лако разбити.

44. Показати да ако $x^2 \equiv d \pmod{n}$ има решење, и $n = pq$ (p, q – прости), онда ова конгруенција има 4 решења.

45. Број x, $1 \leq x \leq n-1$ је непокретна тачка за RSA систем са модулом n ако се пресликава у самог себе. Доказати: ако је x непокретна тачка, онда је и $n - x$ непокретна тачка.

46. Показати да у систему RSA са параметрима p, q, e, d има $r + s + rs$ непокретних тачака, где је $r = \mathrm{nzd}(p-1, e-1)$, $s = \mathrm{nzd}(q-1, e-1)$

47. Претпоставимо да сваки корисник A има тајни пар трансформација $f_A : \mathcal{P} \to \mathcal{P}$ где је $\mathcal{P}$ фиксирани скуп могућих отворених текстова. Алиса жели да Бобану сигурно пренесе поруку применом Меси–Омура поступка, тј. Алиса шаље $f_A(P)$ Бобану, који јој затим шаље $f_B(f_A(P))$, итд. Описати услове које треба да задовољи систем функција f_A да би систем функционисао.

48. За елиптичку криву $E : y^2 = x^3 + Ax + B$ извести изразе за збир две тачке $P + Q$, односно $P + P$.

49. Доказати да елиптичка крива над пољем $\mathbf{F}_q$ има највише $2q+1$ тачку.

50. За елиптичку криву $E : y^2 = x^3 + 3x + 8$ над пољем $\mathbf{F}_{13}$

    а) показати да је скуп тачака криве

    $$E(\mathbf{F}_{13}) = \{\oslash, (1,5), (1,8), (2,3), (2,10), (9,6), (9,7), (12,2), (12,11)\};$$

    б) израчунати $(1,8) + (9,7)$, $2(9,7)$;

    в) показати да $(1,5)$ генерише криву, а затим направити табелу умножака те тачке и табелу логаритама;

    г) користећи ову табелу израчунати $(12,11) + (2,3)$, $(12,2) + (9,6)$, $25(9,7)$.

51. За ПУКДХ користи се елиптичка крива из претходног задатка. Ако се користи генератор $(2,3)$, а тајни кључеви су $e_A = 4$, $e_B = 5$, одредити тачку која се добија као резултат усаглашавања.

52. За систем ЕлГамал користи се елиптичка крива из претпретходног задатка, а генератор из претходног задатка. Ако су тајни кључеви $e_A = 5$, $e_B = 3$, приказати поступак шифровања поруке $M = (12,11)$ (користи се случајни број $k = 4$), а затим поступак дешифровања шифрата.

53. У систему аутентикације заснованом на RSA корисник A изабрао је јавни кључ $e = 7$ и $n = 77$. Ако је он од B добио број 23, како треба да гласи његов одговор, да би саговорника убедио у свој идентитет?

54. За дигиталне потписе засноване на систему RSA корисници A иB имају јавне кључеве $e_A = 3$, $n_A = 15$, односно $e_B = 7$, $n_B = 77$. A жели да пошаље поруку $M = 4$ као потпис неког текста. Који цели број он треба да пошаље?

55. Доказати да за дигиталне потписе засноване на RSA важи следеће тврђење: ако је S_1 потпис поруке m_1, а S_2 потпис поруке m_2, онда је S_1S_2 потпис поруке m_1m_2.

56. Корисник A има јавни кључ $e = 11$, $n = 899$. Како гласи његов RSA дигитални потпис поруке 876?

57. ЕлГамал дигитални потпис. Особа A бира прост број $p = 21739$ и примитивни коррен $g = 7$, а затим тајни кључ $a_A = 15140$.

    а) Који је њен јавни кључ?

    б) Шта је потпис поруке $S = 5331$ ако се користи кључ сесије $k = 10727$?

    б) Како прималац B проверава потпис?

58. Одредити верижни развој бројева $\sqrt{3}$, $\sqrt{5}$, $\sqrt{7}$, $\frac{7}{23}$, π и одредити првих 8 парцијалних разломака.

59. Одредити орбите за линеарне померачке регистре: $1 + x^2 + x^3 + x^4$, $1 + x + x^5$, $1 + x + x^6$.

60. Одредити линеарне комбинације улаза и излаза најближе константи за табелу S у SAES , и за sbox-ове $S1$ и $S4$ за DES .

61. Раставити на чиниоце 14873 поступком који користи случајно лутање по модулу 14873.

62. Применити Фермаов поступак факторизације на број 21079.

63. Доказати да су у пољу $\mathbf{Q}(\sqrt{-5})$ алгебарски цели бројеви облика $a + b\sqrt{-5}$, $a, b \in \mathbf{Z}$.

64. Доказати да се у пољу $\mathbf{Q}(\sqrt{-5})$ број 2 не може раставити у нетривијални производ алгебарских целих бројева.

65. У скупу $\mathbf{Z}(i)$ раставити на чиниоце бројеве $5 + 3i$, $11 - 3i$.

66. Помоћу Полиг-Хелмановог алгоритма израчунати логаритам броја 303, при чему је све остало исто као у примеру у тексту.