Лекция 10

Криптосистеми, използващи задачата за намиране на дискретен логаритъм

10.1 Предварителни бележки

Почти всички задачи, стоящи пред асиметричната криптография могат да бъдат решавани в рамките на RSA. или на вариантите ѝ. Въпреки това криптосистемите, основаващи се на задачата за намиране на дискретен логаритъм са обект на изследвания и продължават да бъдат развивани. Причините за това са няколко.

Най-напред ще споменем техническите предимства, които дават дискретните логаритми. В случаите, в които съществуват алгоритми със сравнима функционалност, един основан на дискретен логаритъм над целите числа по модул p и друг, използващ съставно цяло число n с приблизително същия размер като n, намирането на дискретния логаритъм се оказва малко по-трудно от разлагането на n. Криптосистемите, използващи елиптични криви като че ли предоставят повече сигурност за ключове с равни дължини.

Колкото и прадоксално да звучи, следващото предимство на криптосистемите, използващи дискретни логаритми, следва от едно тяхно огарничение. Широко разпространено е схващането, че е по-лесно да използваме RSA за шифриране на съобщения, отколкото DSA. Поради това експортните забрани са не толкова строги, за оборудване, използвано за създаване на цифров подпис. Много хора харесват алгоритъма на Diffie-Hellman защото сесийният ключ, генериран с него е evanescent.

Не на последно място горчивият опит натрупан от парктическата криптография учи, че не е мъдро да се залага само на един криптографски метод. Желателно е да разполагаме с разнообразни системи, които да могат да бъдат използвани, в случай, че някоя от тях бъде компрометирана. За съжаление дискретните логаритми и факторизацията на цели числа са доста близки. Повечето алгоритми, развити да решават едната задача веднага биват модифицирани за решаване на

другата. От съображения за сигурност бихме желали малко повече разнообразие. Въпреки всичко след повече от 25 години асиметрична криптография алгоритмите на DIFFIE-HELLMAN и RSA са единствените асиметрични криптосистеми, в сигурността на които се вярва и които биват широко използвани. И парадоксално, при поранното откриване на асиметричната криптография от COCKS, ELLIS и WILLIAMSON в началото на 70-те години на XX век се е стигнало в крайна сметка до същите две системи. Има много опити за създаване на асиметрични системи, основаващи се на други принципи. Всички те са гледани с подозрение, тъй като водят до системи, чиято сигурност е компрометирана.

10.2 Дискретен логаритъм

Нека $G = \langle g \rangle$ е крайна циклична група от ред n. Нека α е произволен елемент от G. *Дискретен логаритъм на α при основа g* наричаме единственото цяло число m, $0 \leq m \leq n-1$, такова че $\alpha = g^m$. Това число ще означаваме с $\log_g \alpha$ или само с $\log \alpha$, ако пораждащият елемент g е ясен от контекста.

Пример 10.1. Нека $n = 13$. Групата $\mathbb{Z}_{13}^*$ е циклична от ред 12. Един пораждащ за $\mathbb{Z}_{13}^*$ е 2. По-долу е представена таблица на дискретните логаритми при основа 2 в $\mathbb{Z}_{13}^*$.

x	1	2	3	4	5	6	7	8	9	10	11	12
$\log_2 x$	0	1	4	2	9	5	11	3	8	10	7	6

Свойствата на дискретния логаритъм, представени по-долу, следват директно от дефиницията:

(1) $g^{\log_g \alpha} = \alpha$;

(2) $\log_g \alpha^{-1} = n - \log_g \alpha$;

(3) $\log_g(\alpha\beta) = \log_g \alpha + \log_g \beta$;

(4) $\log_g(\alpha^m) = m \log_g \alpha \pmod{n}$.

Общата задача за намиране на дискретен логаритъм, която ще означаваме с $GDLP(\langle g \rangle, \alpha)$, се формулира по следния начим:

> При зададени крайна циклична група $G = \langle g \rangle$, и произволен елемент $\alpha \in G$, да се намери $\log_g \alpha$.

Особен интерес представлява тази задача, когато G е мултипликативната група на полето с q елемента, $G = \mathbb{F}_q^*$. Трудността на задачата за намиране на дискретен алгоритъм не зависи от избора на пораждащ елемент в групата G. Наистина нека $G = \langle g \rangle = \langle h \rangle$ и нека $\alpha \in G$. Ако $x = \log_g \alpha$, $y = \log_h \alpha$ и $z = \log_g h$, то $\alpha = g^x = h^y = (h^z)^y$, откъдето

$$\log_h \alpha = (\log_g \alpha)(\log_g h)^{-1} \pmod{n}.$$

От последното равенство става ясно, че ако можем да пресмятаме ефективно дискретен логаритъм при основа g, то можем пресмятаме ефективно и дискретен логаритъм при произволна основа h. [1]

[1] Ако елементът h не поражда G, то е възможно $\log_h \alpha$ да не съществува. В този случай елементът $\log_g h$ не е обратим в $\mathbb{Z}_n^*$.

10.3 Няколко криптосистеми,използващи дискретен логаритъм

10.3.1 Система на **Diffie** и **Hellman** за обмен на ключове

Системата на Diffie и Hellman за обмен на ключове епредставена още в оригиналната им работа от 1976г използва следните публични параметри:

- $q = p^n$, където p е просто число;
- γ – примитивен елемент на $\mathbb{F}_q^*$;
- $c_U = \gamma^{x_U}$ за всеки потрбител U.

Всеки потребител U притежава таен ключ, x_U, който е цяло число в интервала $[0, q-1]$.

При описаните условия всеки два потребителя A и B могат да договорят общ ключ $k_{A,B} = \gamma^{x_A x_B}$, който да бъде използван по-нататък със симетрична криптосистема.

A получава този ключ пресмятайки $c_B^{x_A} = (\gamma^{x_A})^{x_B} = k_{A,B}$;

B получава този ключ пресмятайки $c_A^{x_B} = (\gamma^{x_B})^{x_A} = k_{A,B}$;

Пресмятането на x_A при зададено c_A е еквивалентно на намирането на дискретен логаритъм в $\mathbb{F}_q^*$, за което се приема, че е трудна задача. Не изглежда да съществува друг начин за намиране на x_A. При използване на поле с $\sim 2^{1000}$ елемента A и B се нуждаят от около 2000 умножения за пресмятане на общия ключ. Най-добрите известни алгоритми зза намиране на дискретен логаритъм са със сложност $O(e^{C\sqrt{\ln q \ln\ln q}})$.

10.3.2 Схема на **ElGamal**за цифров подпис

Схемата на ElGamal за цифров подпис [23] е недетрминистична[2] и се основава на задачата за намиране на дискретен логаритъм. Модификация на тази схема е залегнала в приетия от NIST стандарт за цифров подпис – Digital Signature Standard (DSS). Sхемата на ElGamal е разработена специално за създаване на електронни подписи за разлика от RSA, която може да бъде използвана както за таен обмен на данни, така и за създаване на цифров подпис.

Схемата на ElGamal създава подписи, които се добавят към подписваното съобщение. Тъй като няма да поставяме ограничение върху дължината на подписваното съобщение, ще се нуждаем и от хеш-функция $h\colon \{0,1\}^* \to \mathbb{Z}_p$, където p е голямо просто число. За момента няма да се интересуваме отконкретноизползвания метод за хеширане, което само по себе си е интересен въпрос. Просто ще считаме h за фиксирана и известна на всичкипартии. Сега пристъпваме към описание на самата схема.

[2] съществуват много валидни подписи за едно и също съобщение

Генериране на ключове в схемата на ElGamal. За да генерира необходимите ключове всеки участник A извършва следните стъпки:

(1) Генерира голямо просто число p и намира пораждащ елемент α в мултипликативната група $\mathbb{Z}_p^*$.

(2) Избира случайно цяло число a, $1 \leq a \leq p-2$.

(3) Пресмята $y = \alpha^a \bmod p$.

(4) Публичният ключ на A е тройката (p, α, y); частен ключ за A е цялото число a.

Генериране на подписи в схемата на ElGamal. За да генерира подпис за съобщението m, A извършва следните стъпки:

(1) Избира случайно цяло число k, $1 \leq k \leq p-2$, за което $\gcd(k, p-1) = 1$.

(2) Пресмята $r = \alpha^k \bmod p$.

(3) Решава по отношение на s сравнението $ks + ar \equiv h(m) (\mathrm{mod}(p-1))$, т.е. пресмята се $s = k^{-1}\left(h(m) - ar\right) (\mathrm{mod}(p-1))$.

(4) Подписът на A за m двойката (r, s).

Верифициране на подписи в схемата на ElGamal. За да верифицира подписа (r, s) за съобщението m всеки участник B извършва следните стъпки:

(1) Получава публичния ключ на A (p, α, y).

(2) Проверява дали $1 \leq r \leq p-1$; ако това не е изпълнено, подписът се отхвърля.

(3) Пресмята $u = y^r r^s \bmod p$.

(4) Пресмята $h(m)$ и $v = \alpha^{h(m)} \bmod p$.

(5) Подписът се приема тогава ис амо тогава, когато $u = v$.

Лесно можем да проверим, че верифициращата част на схемата е коректна, т.е. един подпис се приема, ако е бил създаден с публичния ключ на A. Наистина, в такъв случай имаме

$$\alpha^{h(m)} \equiv \alpha^{ar+ks} \equiv (\alpha^a)^r (\alpha^k)^s \equiv y^r r^s (\mathrm{mod} p).$$

Така $u = v$, което искахме да покажем.

Пример 10.2. Да приемем, че A избира просто число $p = 2357$ и пораждащ елемент на $\mathbb{Z}^*_{2357}$, $\alpha = 2$. Да допуснем, че A избира частен ключ $a = 1751$. Тогава тя пресмята $y = \alpha^a \bmod p = 2^{1751} \bmod 2357 = 1185$. Така публичният клыч на a е $(p = 2357, \alpha = 2, y = 1185)$.

За простота ще приемем, че съобщенията са елементите на $\mathbb{Z}_p$ и че $h(m) = m$. За да подпише $m = 1463$, A избира случайно цяло число, да речем $k = 1529$, пресмята $r = \alpha^k \bmod p = 2^{1529} \bmod 2357 = 1490$ и $k^{-1} \bmod (p-1) = 245$, и накрая намира

$$s = 245\,(1463 - 1751) \bmod 2356 = 1777.$$

Подписът на A за $m = 1463$ е двойката $(r = 1490, s = 1777)$.

За да верифицира подписа $(1490, 1777)$ за съобщението 1463, B пресмята $u = 1185^{1490} \cdot 1490^{1777} \bmod 2357 = 1072$ и $v = 2^{1463} \bmod 2357 = 1072$. Тъй като $u = v$, B приема подписа на A. □

Сега ще коментираме сигурността на схемата на ElGamal. Потенциален опонент би могъл да се опита да подправи подписа на A за съобщението m, избирайки случайно цяло число k и пресмятайки $r = \alpha^k \bmod p$. По-нататък той трябва да определи $s = k^{-1}\,(h(m) - ar) \bmod (p-1)$. Ако задачата за намиране на дискретен логаритъм е трудна, то опонентът няма по-добра стратегия освен да отгатне S; вероятността той да успее е $1/p$, което е пренебрежимо малко при големи p.

Важна предпазна мярка, която трябва да се има пред вид, е избирането на *различно* k за всяко ново подписвано съобщение. В противен случай частният ключ може да бъде намерен (с много голяма вероятност). нак предположим, че $s_1 = k^{-1}\,(h(m_1) - ar) \bmod (p-1)$ и $s_= k^{-1}\,(h(m_2) - ar) \bmod (p-1)$. Тогава,

$$(s_1 - s_2)k \equiv (h(m_1) - h(m_2))\,(\mathrm{mod}(p-1). \tag{10.1}$$

Ако $\gcd(s_1 - s_2, p-1) = 1$, можем да получим k от

$$k = (s_1 - s_2)^{-1}\,(h(m) - h(m_2)) \bmod (p-1).$$

Ако $1 < \gcd(s_1 - s_2, p-1) < p-1$, сравнението (10.1) има повече от едно решение, но броят на възможните k може да бъде силно намален. Информация за k не получаваме единствено в случая, когато $p-1$ дели $s_1 - s_2$. След като k е определено a може да бъде лесно пресметнато, решавайки линейното сравнение $ra + ks \equiv h(m) \bmod (p-1)$ по отношение на a.

Ако схемата не използва хеш-функция h, то сравнението определящо подписа е $s = k^{-1}(m - ar) \bmod (p-1)$ и опонентъможе да пробва екзестенциална (?) атака за подправяна по следния начин. Той избира двойка цели числа (z_1, z_2), $\gcd(z_2, p-1) = 1$, и пресмята $r = \alpha^{z_1} y^{z_2} \bmod p = \alpha^{z_1 + az_2} \bmod p$ и $s = -rz_2^{-1} \bmod (p-1)$. Сега двойката (r, s) е валиден подпис за съобщението $m = sz_1 \bmod (p-1)$, тъй като $(\alpha^m \alpha^{-ar})^{s^{-1}} = \alpha^{z_1} \alpha^{a(-rs^{-1})} = \alpha^{z_1} y^{z_2} = r$. [3]

В стъпка (2) на верифициращия алгоритъм трябва да се провери, че $0 < r < p$. ако тази проверка не с еизвърши, то опонентът може да подписва съобщения по свой избор при условие, че притежава един валиден подпис на A. Да предположим, че

[3] По-нататък $\alpha^m \alpha^{-r} = r^s$, откъдето $\alpha^m = \alpha^{ar} r^s$ и $\alpha^m = y^r r^s$.

(r, s) е подпис на A за съобщението m. Опонентът избира съобщение m' и пресмята $h(m')$ и $z = h(m') \cdot [h(m)]^{-1} \bmod (p-1)$ (при положение, че $[h(m)]^{-1} \bmod (p-1)$ съществува). По-нататък той пресмята $s' = sz \bmod (p-1)$ и r', за което $r' \equiv rz(\mathrm{mod}(p-1))$ и $r' \equiv r(\mathrm{mod}p)$. Последното е винаги възможно съгласно Китайската Теорема за остатъците. Двойката (r', s') е подпис за съобщението m', който ще бъде приет за валиден, ако липсваше стъпка (2).

Определена грижа трябва да бъде положена и при генериране на параметрите на една схема на ElGamal. На първо място простото число p трябва да бъде достатъчно голямо (толкова голямо, че да бъде сигурно срещу атака с index calculus алгоритъма. По-нататък $p-1$ трябва да има поне един достатъчно голям прост делител, за да се осигури надеждност срещу атака чрез алгоритъма на Pohlig-Hellman.

Известни предпазни мерки трябва да бъдат взети и при избора на пораждащия елемент на $\mathbb{Z}_p^*$. да предположим, че $p \equiv 1(\mathrm{mod}4)$ и че пораждащият елемент α удовлетворява следните условия:

(a) α дели $p-1$ и

(b) пресмятането на дискретни логаритми в (единствената) подгрупа S от ред α в Z_p^* може да бъде ефективно извършено (напр. чрез Pohlig-Hellman).

При такъв избор на p и α е възможно конструирането на подписи (без знанието на тайния ключ на A, които да бъдат приети от верифициращия алгоритъм. Нека положим $p-1 = \alpha q$. За да подпише съобщение m, опонентът извършва следните стъпки:

(i) Пресмята $t = (p-3)/2$ и полага $r = q$.

(ii) Определя число z, за което $\alpha^{qz} \equiv y^q(\mathrm{mod}p)$, където y е публичният ключ на A. Това е възможно, защото α^q и y^q са елементи на S и α^q е пораждащ елемент на S.

(iii) Пресмята $s = t \cdot (h(m) - qz) \bmod (p-1)$.

(iv) (r, s) е подпис за m, който ще бъде приет от верифициращия алгоритъм.

Сега ще проверим, че веривициращото сравнение $y^r r^s \equiv \alpha^{h(m)}(\mathrm{mod}p)$ наистина се удовлетворява. За да проверим това нека първо забележим, че $\alpha q \equiv -1(\mathrm{mod}p)$, т.е. $\alpha \equiv -q^{-1}(\mathrm{mod}p)$, и че $q^{(p-1)/2} \equiv -1(\mathrm{mod}p)$. Оттук можем да получим, че $q^t = q^{(p-1)/2} q^{-1} \equiv -q^{-1} \equiv \alpha(\mathrm{mod}p)$. Сега

$$r^s y^r = (q^t)^{h(m)-qz} y^q \equiv \alpha^{h(m)} \alpha^{-qz} y^q \equiv \alpha^{h(m)} y^{-q} y^q = \alpha^{h(m)}(\mathrm{mod}p).$$

да отбележим, че в случая, когато $\alpha = 2$ е пораждащ елемент, тоусловия (a) и (b) са тривиално изпълнени. Описаната атака може да бъде избягната, ако α се избира като пораждащ елемент за подгрума на $\mathbb{Z}_p^*$ от прост ред вместо пораждащ за самата Z_p^*.

10.3.3 Криптосистема на **ElGamal**

Генериране на ключове. За да генерира необходимите ключове всеки потребител A извършва следното:

(1) A генерира голямо просто число p ипораждащ γ на мултипликативната група $\mathbb{Z}_p^*$.

(2) A избира случайно цяло число x_A, $1 < x_A < p-1$ ипресмята $c_A = \gamma^{x_A} \pmod p$; публичен ключ за A представлява (p, γ, c_A), а теаен ключ е числото x_A.

Шифриране. За да изпрати съобщение $0 \leq x < p$ на A, B извършва следните стъпки:

(1) B получава публичния ключ на A: (p, γ, c_A);

(2) B избира случайно цяло число k, $1 \leq k \leq p-2$;

(3) B пресмята $y_1 = \gamma^k \pmod p$ и $y_2 = xc_A^k \pmod p$.

(4) криптотекст за x е двойката (y_1, y_2), която се изпраща на A.

Дешифриране. за да възстанови открития текст x от криптотекста (y_1, y_2), A извършва следните стъпки:

(1) A получава двойката (y_1, y_2);

(2) A пресмята $x = y_2(y_1^{x_A})^{-1} \pmod p$.

Коректността на дешифрирането се проверява лесно:

$$\begin{aligned} y_2(y_1^{x_A})^{-1} &\equiv xc_A^k(\gamma^{kx_A})^{-1} \pmod p \\ &\equiv x \cdot \gamma^{kx_A} \cdot \gamma^{-kx_A} \pmod p \\ &\equiv x \end{aligned}$$

Пример 10.3. Нека A е избрала $p = 17$, $\gamma = 3$ и $x_A = 7$. Тогава $c_A = 3^7 \pmod{17} = 11$. Акоизбраното съобщение е $x = 5$, то B извършва стъпките:
- избира случайно цяло число $1 \leq k \leq p-2$, да речем $k = 3$;
- пресмята $y_1 = 3^3 \pmod{17} = 10$ и $y_2 = 5 \cdot 11^3 \pmod{17} = 8$,
- криптотекстът за $x = 3$ е $(y_1, y_2) = (10, 8)$.

За да дешифрира A пресмята

$$y_1(y_2^{x_A})^{-1} = 8 \cdot (10^7)^{-1} = 8 \cdot 5^{-1} \pmod{17} = 5.$$

Криптосистемата на Ел Гамал се дефинира стандартно в мултипликативаната група $\mathbb{Z}_p^*$, но може лесно да се обобщи за произволна циклична група G, в която задачата за намиране на дискретен логаритъм е трудна. По специално G трябва да отговаря на две условия:

(а) *ефективност:* груповата операция в G може да се имплементира относително лесно;

(b) *сигурност:* задачата за намиране на дискретен логаритъм в G е изчислително трудна.

Така например,адитивната групаот остатъците помодул n $(\mathbb{Z}_n, +)$ не отговаря на второто условие. За групите в следния списък се счита, че изпълняват и двете условия, като на първите три се отдщля осбено голямовнимание;

(1) мултипликативната група $\mathbb{Z}_p^*$ от целите числа по модул просто число p;

(2) мултипликативната група $\mathbb{F}_{2^h}^*$ на крайното поле от рд $q = 2^h$;

(3) мултипликативната група от точките на елиптична крива над крайно поле;

(4) мултипликативната група накрайнополе $\mathbb{F}_q^*$, където $q = p^h$,p просто число;

(5) групата от обратимите елементи на $\mathbb{Z}_n^*$,където n е съставно число;

(6) якобиана на хиперелиптична крива,дефинирана над крайно поле:

(7) групата от класове на имагинерно квадратично числово поле.

10.3.4 Digital Signature Algorithm

Параметри на DSA

(a) Избираме просто число p, за което $2^{511} < p < 2^{512}$.

(b) Избираме прост делител q на $p-1$, за който е изпълнено $2^{159} < q < 2^{160}$.

(c) Избираме елемент $g = h^{(p-1)/q} \bmod p$, където h е цяло число $0 < h < p$, такова че $h^{(p-1)/q} \bmod p > 1$. (С други думи g е елемент от $\mathbb{Z}_p^*$, чийто мултипликативен ред е q.)

(d) Нека x е цяло число, за което $0 < x < q$.

(e) Нека по-нататък $y = g^x \bmod p$.

(f) С m означаваме съобщението,което ще подписваме.

(g) Нека k е с лучайноцяло числа, $0 < k < q$.

(h) Накрая, нека H е еднопосочна хеш-функция.

Целите числа p, q, g са публични и общи за всички потербители.Тайният ключ е цялото число x. Публичният ключ на потрабителят с таен ключ x е числото y. Хеш-функцията H се избира по такъв начин, че е невъзможно (изчислително невъзможно) да се създаде съобщение, което да се хешира в някаква предварително зададена стойност.

Генериране на подпис. Нека потребителят A, генерирал ключовете в предната точка,иска да подпише съобщението m. Той извършва следните пресмятания:

(a) A избира случайно цяло число k, $0 < k < q$.

(b) A пресмятаме $r = (g^k \bmod p) \bmod q$.

(c) A пресмята $s = (k^{-1}(H(m) + xr)) \bmod q$, където k^{-1} е мултипликативният обратен на k по модул q, $0 < k^{-1} < q$.

(d) Двойката r, s е подписът на A за съобщението m. Тези две числасе изпращат заедно със съобщението m.

Верификация на подпис. За да провери автентичността на подписа (r, s) на A за съобщението m, потребителят B извършва следните стъпки. Най-напред той получава p, q, g заедно с публичния ключ y. Ще предполагаме , че m', r' и s' са получените от B версии на m, r и s. B проверява дали $0 < r' < q$, $0 < s' < q$. Ако някое от тези условия не се изпълнява,топодписът се отхвърля. Ако и двете условия са изпълнени, то B пресмята:

$$\begin{aligned} w &= (s')^{-1} \bmod q; \\ u_1 &= (H(m')w) \bmod q; \\ u_2 &= (r'w) \bmod q; \\ v &= (g^{u_1} y^{u_2} \bmod p) \bmod q \end{aligned}$$

Ако $v = r'$ топодписът сеприема и B трябва да приеме, че съобщението е подписано от партията, притежаваща тайния ключ, съответстващ на y. Ако $v \neq r'$, подписът на съобщението m се счита за невалиден.

Сега ще докажем, че ако $m' = m$, $r' = r$, $s' = s$ то $v = r'$.

Лема 10.4. *For every nonnegative integer t and $g = h^{(p-1)/q} \bmod p$, we have $g^t \bmod p = g^{t \bmod q} \bmod p$.*

Лема 10.5. *For every two nonnegative a and b we have*

$$g^{a \bmod q + b \bmod q} \bmod p = g^{(a+b) \bmod q} \bmod p.$$

Лема 10.6. $y^{(rw) \bmod q} \bmod p = g^{(xrw) \bmod q} \bmod p$.

Лема 10.7. $(H(m) + xr)\, w \bmod q = k$.

Теорема 10.8. *Ако в горните означения $m' = m$, $r' = r$, $s' = s$,то $v = r'$.*

Proof.

$$\begin{aligned} v &= (g^{u_1} y^{u_2} \bmod p) \bmod q \\ &= \left(g^{H(m)w \bmod q} \cdot y^{(rw) \bmod q} \bmod p\right) \bmod q \\ &= \left(g^{(H(m)w + xrw) \bmod q} \bmod p\right) \bmod q \\ &= \left(g^k \bmod p\right) \bmod q \\ &= r = r'. \end{aligned}$$

□

Криптографски проблеми с DSA (R.Rivest and M.Hellman, Comm. ACM)

(1) Общите модули са твърде рискови. Макар DSA да не изисква общи модули,то тя окуражава такова поведение.

(2) Задачата за намиране на дискретен логаритъмелесен за някои прости числа. В редица случаи може да се окаже трудно да забележимтакива числа. Потребителите не трябва да използват прости числа,използваниот други.

(3) Лежащ в основата на DSA не е общият проблем за намиране на дискретен логаритъм, а негов специален случай.

(4) Ако се разкрие случайното число k, то тайният ключ може да бъде реконструиран. Следователноотогромно значение е сигурното генериране на всички случайни числа,използвани в алгоритъма.

(5) Ключовете са къси.

10.4 Алгоритми за намиране дискретен логаритъм

10.4.1 Baby step/giant step

Нека $G = \langle\alpha\rangle$ е циклична група от ред n. Нека β е произволен елементот G. Искаме да определим цялото число $x = \log_\alpha \beta$ Да означим $m = \lceil\sqrt{n}\rceil$. Ако $\beta = \alpha^x$, то

$$x = im + j, 0 \leq i, j \leq m.$$

Сега очевидно $\beta(\alpha^{-m})^i = \alpha^j$. Идеята е да създадем таблица

1	α
2	α^2
3	α^3
$\vdots$	$\vdots$

сортирана по втория стълб.По-нататък пресмятаме стойностите $\beta(\alpha^{-m})^i$ и сравняваме със втория стълб до съвпадение. Алгоритъмът изглежда така:

1) Set $m = \lceil\sqrt{n}\rceil$

2) Construct a table (j, α^j), $0 \leq j \leq m$ and sort it by the second column

3) Compute α^{-m}

4) Set $\gamma = \beta$

5) for $i = 0$ to $m - 1$ do

6) if $\gamma = \alpha^j$ for some $j \in \{0, \ldots, m\}$

7) then return $x = im + j$

8) $\gamma = \gamma\alpha^{-m}$

Необходимите ресурси са $O(\sqrt{n})$ памет за елементите на таблицата, $O(\sqrt{n})$ умножения за конструирането ѝ, $O(\sqrt{n}\log\sqrt{n})$ за сортирането. Цикълът 5)–8) отнема $O(\sqrt{n})$ умножения и $O(\sqrt{n})$ достъпа до таблицата. Общото време за изпълнение е $O(\sqrt{n})$.

Пример 10.9. Нека разгледаме $G = \mathbb{Z}_{113}^*$, $n = 112$, $m = \lceil\sqrt{112}\rceil = 11$. Нека по-нататък $\alpha = 3$ е примитивен елемент в G и $\beta = 57$. Търсим $x = \log_3 57$. Сега таблицата и сортирания ѝ вариант изглеждат така:

j	$3^j \pmod{113}$
0	1
1	3
2	9
3	27
4	81
5	17
6	51
7	40
8	7
9	21
10	63

j	$3^J \pmod{113}$
0	1
1	3
8	7
2	9
5	17
9	21
3	27
7	40
6	51
10	63
4	81

По-нататък пресмятаме

- $\alpha^{-1} = 3^{-1} \pmod{113} = 38$;
- $\alpha^{-m} = 38^{11} \pmod{113} = 58$;
- $\gamma = \beta\alpha^{-mi} \pmod{113}$, $i = 0, 1, 2, \ldots$.

i	0	1	2	3	4	5	6	7	8	9
$\gamma = 57 \cdot 58^i$	57	29	100	37	112	55	28	39	2	**3**

Сега $i = 9, j = 1$ и $x = 9 \cdot 11 + 1 = 100$, т.е. $\log_3 57 = 100$.

10.4.2 ρ-метод на Полард

Недостатък на предния алгоритъм е, че има големи изисквания за памет. Методът,които ще представим, запазва сложността като намалява изискванията за памет,но сега може да се оцени самоочакваното време за работа.

Нека $f : S \to S$ е случайна функция върхумножеството S, $|S| = n$. избираме случаен елемент $x_0 \in S$ и последователно пресмятаме

$$x_{i+1} = f(x_i), \quad \text{за } i \geq 0.$$

Стойностите $x_0, x_1, x_2, \ldots$ разглеждаме като детерминирано случайно блуждаене. Това означава, че процесът на пресмятане е детерминиран, но редицата $x_0, x_1, x_2, \ldots$ се държи като случайна.

Тъй като S е крайно множество, то в някоймомент получаваме $x_i = x_j$ за някаоя двойка отиндекси и тогава

$$x_{i+1} = f(x_i) = f(x_j) = x_{j+1}.$$

Ако изобразим блужданието графично, то ще се получицикълс опашка, т.е. нещо, което наподобява гръцката буква ρ. Може да се покаже, че очакваната дължина на опашката е $\sqrt{\pi n/8}$,а очакваната дължина на цикъла ще е $\sqrt{3\pi n/8}$. Целта е да се намери началото на цикъла в случайното блуждаене.Благодарение на парадокса на близнаците може да се получи, че очакваното повторение ще се случи след $\sqrt{\pi n/2}$ итерации на функцията f. Нестрого:доочакваното повторение ще измине $O(\sqrt{n})$ време, а за да засечм повторението ще ни е нужна $O(\sqrt{n})$ памет.

За да се намери повторението използваме алгоритъмана Флойд за търсене на цикли. Тои се заключава в следното: по двойката (x_1, x_2) пресмятаме двойката (x_2, x_4), след това (x_3, x_6) и т.н. След (x_i, x_{2i}) се изчислява

$$(x_{i+1}, x_{2(i+1)}) = (f(x_i)f(f(x_{2i}))).$$

Процесът завършва при установяване на равенството $x_m = x_{2m}$. Ако дължината на опашката на нашата редица е λ, а дължината на цикъла е μ, то такова съвпадение може да се очаква при

$$m = \mu(1 + \lfloor \lambda/\mu \rfloor).$$

Вземайки под внимание, че $\lambda < m \leq \lambda + m$, получаваме, че $m = O(\sqrt{n})$. Това ще е точна оценка за сложността, ако f се държи като случайна функция. Следователно можем да забележимповторение без да пазим практически нищо в паметта на компютъра.

Нека сега $G = \langle \alpha \rangle$ е група от ред n и в нея решаваме задачата за намиране на цялото x, за което $\beta = \alpha^x$.

Групата G се разбива на триприблизително равни части $S_1, S_2, S - 3$, $1 \notin S_2$:

$$G = S_1 \cup S_2 \cup S_3.$$

Принадлежността на даден елемент в S_i трябва да е лесно проверима. Определяме случайното блуждаене в G по следния начин:

$$\begin{aligned} x_0 &= 1 \\ x_{i+1} &= f(x_i) = \begin{cases} \beta x_i & x_i \in S_1 \\ x_i^2 & x_i \in S_2 \\ \alpha x_i & x_i \in S_3. \end{cases} \end{aligned} .$$

Ако $x_i = \alpha^{a_i}\beta^{b_i}$,то $a_0 = 0$, $b_0 = 0$ и

$$\begin{aligned} a_{i+1} &= \begin{cases} a_i & x_i \in S_1, \\ 2a_i & x_i \in S_2, \\ a_i + 1 & x_i \in S_3, \end{cases} \\ b_{i+1} &= \begin{cases} b_i + 1 & x_i \in S_1, \\ 2b_i & x_i \in S_2, \\ b_i + 1 & x_i \in S_3, \end{cases} \end{aligned}$$

Използвайки метода на Флойд намираме двойка (x_i, x_{2i}),за която $x_i = x_{2i}$. Това означава $\alpha^{a_i}\beta^{b_i} = \alpha^{2a_i}\beta^{2b_i}$ или $\beta^{b_i - b_{2i}} = \alpha^{a_{2i} - a_i}$. Взимайки логаритъм при основа α, получаваме:

$$(b_i - b_{2i}) \log_\alpha \beta = a_{2i} - a_i \pmod{()n}$$

или

$$\log_\alpha \beta = \frac{a_{2i} - a_i}{b_i - b_{2i}} \pmod{n}.$$

При големи n вероятността да се случи $b_i \equiv b_{2i} \pmod{n}$ е мнозго малак, така че горният израз дава наистина стойността на $\log_\alpha \beta$.

Алгоритъмът, който описахме изглжда така:

1) $x_0 = 1$, $a_0 = 0$, $b_0 = 0$

2) for $i = 1, 2, \ldots$ do

3) find (x_i, a_i, b_i) and (x_{2i}, a_{2i}, b_{2i}) using $(x_{i-1}, a_{i-1}, b_{i-1})$ and $(x_{2(i-1)}, a_{2(i-1)}, b_{2(i-1)})$

4) if $(x_i = x_{2i})$

5) then $r = b_i - b_{2i} \pmod{n}$

6) if $(r = 0)$ then failure

7) else $x = r^{-1}(a_{2i} - a_i) \pmod{n}$

Пример 10.10. Нека $\mathbb{Z}_{607}$, $G = \langle 64 \rangle$,$|G| = 101$. Дефинираме

$$\begin{aligned} S_1 &= \{x \in \mathbb{Z}_{607} \mid x \leq 201\}, \\ S_2 &= \{x \in \mathbb{Z}_{607} \mid 202 \leq x \leq 403\}, \\ S_3 &= \{x \in \mathbb{Z}_{607} \mid 404 \leq x \leq 606\}. \end{aligned}$$

Нека $\alpha = 64, \beta = 122$.Последователнополучаваме:

i	x_i	a_i	b_i	x_{2i}	a_{2i}	b_{2i}
0	1	0	0	1	0	0
1	122	0	1	316	0	2
2	316	0	2	172	0	8
3	308	0	4	170	0	18
4	172	0	8	7	0	38
5	346	0	9	309	0	78
6	137	0	18	352	0	56
7	325	0	19	167	0	12
8	7	0	38	498	0	26
9	247	0	39	172	2	52
10	309	0	78	137	4	5
11	182	0	55	7	8	12
12	352	0	56	309	16	26
13	76	0	11	352	32	53
14	167	0	12	167	64	6

10.4.3 Алгоритъм на **Pohlig-Hellman**

Задачата за намиране на дискретен логаритъм в мултипликативната група $\mathbb{F}_q^*$ е лесна, ако $q-1$ има само "малки" прости делители. Това се дължи на един алгориъм, предложен от Pohlig и Hellman. За да илюстрираме основната идея нека най напред разгледаме случая, когато $q-1=2^n$. (Ще отбележим, че най-голямото известно просточисло от вида 2^n+1 е $2^{16}+1$.)

Нека е дадено $c=\alpha^m$ като примитивният елемент α е известен. Трчбва да се определи показателят m, двоичнотопредставяне на който е

$$m=m_0+m_1\cdot 2+\ldots+m_{n-1}\cdot 2^{n-1},\quad m_i\in\{0,1\}.$$

така достатъчно е да се определят числата $m_i\in\{0,1\}$.

Тъй като α еприимитивен елементимаме $\alpha^{\frac{q-1}{2}}=-1$ (в $\mathbb{F}_q$). Следователно

$$c^{\frac{q-1}{2}}=\alpha^{m\cdot\frac{q-1}{2}}=\alpha^{(m_0+m_1 2+\ldots+m_{n-1}2^{n-1})\cdot\frac{q-1}{2}}=\alpha^{m_0\cdot\frac{q-1}{2}}=\begin{cases}1 & \text{if } m_0=0,\\ -1 & \text{if } m_0=1\end{cases}$$

За да пресметнем $c^{\frac{q-1}{2}}$ са необходими $2\lceil\log_2 q\rceil$ умножения. Да положим $c_1=c\cdot\alpha^{-m_0}$. По същия начин определяме m_1 от равенството

$$c_1^{\frac{q-1}{4}}=\alpha^{(m_1+m_2 2+\ldots+m_{n-1}2^{n-2})\cdot\frac{q-1}{2}}=\alpha^{m_1\cdot\frac{q-1}{2}}=\begin{cases}1 & \text{if } m_1=0,\\ -1 & \text{if } m_1=1\end{cases}$$

Сега полагаме $c_2=c_1\cdot\alpha^{-2m_1}$ ипресмятаме $c_2^{\frac{q-1}{8}}$.Продължаваме по същия начин докатоопределим всички m_i.

Забележка 10.11. Ако $q-1=2^t s$, топроцедурата, която току що описахме може да определи младшите t бита на числото m.

Да разгледаме сега общия случай: $q-1=p_1^{n_1}p_2^{n_2}\ldots p_k^{n_k}$. Ще приемем, че $p_1<p_2,\ldots<p_n$ са малки прости числа.

Означаваме $m^{(i)}=m \bmod p_i^{n_i}$, $i=1,\ldots,k$. Ако определим числата $m^{(i)}$ за всички $1\le i\le k$, тониеможем да възстановим m с помощта на Китайската теорема за остатъците. Последната може да се имплементира,използвайки $O(k\log_2 q)$ операции и $O(k\log_2 q)$ бита памет. Нека

$$m^{(i)}=\sum_{j=0}^{n_i-1}b_j p_i^j,\quad 0\le b_j\le p_i-1.$$

Да създадем таблицата

0	1	2	$\ldots$	p_i-1
1	β_i	β_i^2	$\ldots$	$\beta_i^{p_i-1}$

,

където $\beta_i=\alpha^{\frac{q-1}{p_i}}$ е p_i-ти корен на 1. Простите числа p_i са толкова малки, че да е възможно да запазим в паметта горната таблица. По-нататък пресмятаме

$$c^{\frac{q-1}{p_i}}=\alpha^{\frac{(q-1)m}{p_i}}=\alpha^{\frac{(q-1)m^{(i)}}{p_i}}=\alpha^{\frac{(q-1)b_0}{p_i}}=\beta_i^{b_0}.$$

и с една проверка в таблицата определяме b_0. По-нататъкпресмятаме $c_1 = c \cdot \alpha^{-b_0}$, $c_1^{\frac{q-1}{p_i^2}}$ иопределяме b_1 и т.н. доопределяме на всички b_i. Алгоритъмът на Pohlig и Hellman е илюстриран в следния пример.

Пример 10.12. Нека $q = 8101$, $q - 1 = 8100 = 2^2 \cdot 3^4 \cdot 5^2$, $\alpha = 6$, $\alpha^{-1} = 6751$. Извършвамепредварителните пресмятания:

$$\begin{aligned} \beta_1 &= \alpha^{\frac{8100}{2}} = \alpha^{4050} = 8100, \\ \beta_2 &= \alpha^{\frac{8100}{3}} = \alpha^{2700} = 5883, \\ \beta_3 &= \alpha^{\frac{8100}{5}} = \alpha^{1620} = 3547. \end{aligned}$$

$p_1 = 2$

i	0	1
β_1^i	1	8100

$p_2 = 3$

i	0	1	2
β_2^i	1	5883	2217

$p_3 = 5$

i	0	1	2	3	4
β_3^i	1	3547	356	7077	5221

$$\begin{aligned} a &\equiv 1 \bmod 4 \\ a &\equiv 0 \bmod 81 \quad \Rightarrow a \equiv 2025 \bmod 8100 \\ a &\equiv 0 \bmod 25 \\ b &\equiv 0 \bmod 4 \\ b &\equiv 1 \bmod 81 \quad \Rightarrow b \equiv 6400 \bmod 8100 \\ b &\equiv 0 \bmod 25 \\ d &\equiv 0 \bmod 4 \\ d &\equiv 0 \bmod 81 \quad \Rightarrow d \equiv 2025 \bmod 8100 \\ d &\equiv 1 \bmod 25 \end{aligned}$$

Нека $\alpha^m \equiv c \bmod 8101$, където $c = 7531$. Тогава

$p_1 = 2, n_1 = 2$

$$\begin{aligned} & c = 7531, && c^{\frac{8100}{2}} = 8100 && \Rightarrow b_0 = 1 \\ & c_1 = c\alpha^{-1} = 8006, && c_1^{\frac{8100}{4}} = 1 && \Rightarrow b_1 = 0 \\ & \Rightarrow m^{(1)} = 1 + 0 \cdot 2^1 = 1 \end{aligned}$$

$p_2 = 3, n_2 = 4$

$$\begin{aligned} & c = 7531, && c^{\frac{8100}{3}} = 2217, && \Rightarrow b_0 = 2 \\ & c_1 = c \cdot \alpha^{-2} = 6735, && c^{\frac{8100}{3^2}} = 1, && \Rightarrow b_1 = 0 \\ & c_2 = c_1 = 6735, && c_2^{\frac{8100}{3^3}} = 2217, && \Rightarrow b_2 = 2 \\ & c_3 = c_2 \cdot \alpha^{-2 \cdot 3^2} = 6992, && c_3^{\frac{8100}{3^4}} = 5883, && \Rightarrow b_3 = 1 \\ & \Rightarrow m^{(2)} = 2 + 0 \cdot 3 + 2 \cdot 3^2 + 1 \cdot 3^3 = 47 \end{aligned}$$

$p_3 = 5, n_3 = 2$

$$\begin{aligned} & c = 7531, && c^{\frac{8100}{5}} = 5221, && \Rightarrow b_0 = 4 \\ & c_1 = c \cdot \alpha^{-4} = 7613, && c_1^{\frac{8100}{5^2}} = 356, && \Rightarrow b_1 = 2 \\ & \Rightarrow m^{(3)} = 4 + 2 \cdot 5 = 14. \end{aligned}$$

$$\begin{aligned} m &= a \cdot m^{(1)} + b \cdot m^{(2)} + d \cdot m^{(3)} \\ &= 2025 \cdot 1 + 6400 \cdot 47 + 7776 \cdot 14 \\ &= 6689 \bmod 8100 \end{aligned}$$

така получаваме $6^{6689} \equiv 7531 \bmod 8101$.

10.4.4 Index calculus метод

Да разгледаме циклична група G от ред N, породена от елемента g, т.е. $G = \{e, g, g^2, \ldots, g^{N-1}\}$, $g^N = e$. Нашата цел е при зададен елемент $h = g^m$ да намерим цялото число m. Основните стъпки при index-calculus алгоритъма са следните:

(1) Избираме подмножество $S \subset G$, имащо свойството, че относително голям брой елементи от G могат да бъдат представени като произведение на елементи от S поефективен начин. Множеството S наричаме базис на множителите (factor base). Всеки елемент $g \in G$, който може да бъде представен като произведение на елементи от S наричаме *гладък по отношение на* S. Нека $k = |S|$. В следващите две стъпки всеки елемент в S ще бъде записан като степен на g.

(2) Намираме достатъчно голям набор I от такива експоненти i, които могат да бъдат ефективно като произведение на елементи от S, да речем $g^i = s_1^{u_{i,1}} s_2^{u_{i,2}} \ldots s_k^{u_{i,k}}$. Пресмятайки $\log_g$ от двете страни на това равенство, получаваме множество отлинейни сравнения

$$i \equiv u_{i,1} \log_g s_1 + u_{i,2} \log_g s_2 + \ldots + u_{i,k} \log_g s_k (\bmod N), i \in I. \tag{10.2}$$

(3) Решаваме системата (10.2) по отношение на числата $\log_g s_j$, $1 \leq j \leq k$. За да можем да решим системата трябва да има ранг k и мнзохеството I трябва да е достатъчно голямо.

(4) Избираме случайна експонента r и се опитваме да представим $g^r h$ като произведение на елементи от S. Щом това се окаже възможно, да ерчем $g^r h = s_1^{\nu_1} s_2^{\nu_2} \ldots s_k^{\nu_k}$ взимаме отново логаритъм от двете страни и получаваме

$$r + m \equiv \nu_1 \log_g s_1 + \nu_2 \log_g s_2 + \ldots + \nu_k \log_g s_k (\bmod N). \tag{10.3}$$

Стойностите на всяко от числата $\log_g s_i$ са известни (стъпка (3)), следователно можем да определим и m.

Целта на стъпки (2) и (3) е определянето на дискретения логаритъм за всички елементи в избрания базис на множителите. В стъпка (4) се опитваме да сведем задачата за намиране на дадения дисцретен логаритъм към намиране на дискретен логаритъм на елемент, който е произведение на базисните елементи.

Оптималната мощност на базиса на множителите S е компромис между количеството памет, с което разполагаме, и вероятността случаен елемент от G (именно $g^r h$)да може да бъде изразен като произведение на елементи от S. При горния подход се сблъскваме с два (сързани) неразрешени проблема:

- Как да определим добър базис на множителите?
- Как да изразим произволен елемент в G като произведение на елементи от S?

Сложността на index-calculus алгоритъма е субекспоненциална от $\log_2 N$.

Пример 10.13. Да разгледаме случая, когато $G = \mathbb{Z}_p^*$, мултипликативната група на полето $\mathbb{Z}_p$. В този случай $G = \{0, 1, \ldots, p-1\}$. Базиса на множителите избираме като първите k прости числа $p_1, p_2, \ldots, p_k$. Ако k е достатъчно голямо, то и броят на елементите на G, които мозгат да бъдат изразени като произведение на степени на тези k прости, е достатъчно голяма. За да изразим елемент от G като произведение на елементи от S, ние последователно делим този елемнт на p_i до евентуално получаване на 1. Сложността на тази модификация на index-calculus алгоритъма е $e^{C\sqrt{\ln p \ln\ln p}}$ за някаква константа C.

Нека $p = 541$. Лесно се проверява, че $g = 2$ е примитивен елемнт в $\mathbb{Z}_p^*$. Нека по-нататъ$S = \{2, 3, 5, 7, 11\}$. Сега се опитваме да намерим степени на g, които могат да се представят като произведение на елементите на S. След наыколко проби получаваме

$$\begin{aligned} 2^{14} &\equiv 2^1 \cdot 7^1 \cdot 11^1 (\mathrm{mod}\, 541); \\ 2^{81} &\equiv 2^1 \cdot 3^1 \cdot 7^2 (\mathrm{mod}\, 541); \\ 2^{207} &\equiv 5^2 \cdot 11^1 (\mathrm{mod}\, 541); \\ 2^{214} &\equiv 5^1 \cdot 7^1 (\mathrm{mod}\, 541); \\ 2^{300} &\equiv 2^5 \cdot 11^1 (\mathrm{mod}\, 541). \end{aligned}$$

Полагаме

$$m_1 = \log_2 2, m_2 = \log_2 3, m_3 \log_2 5, m_4 = \log_2 7, m_5 = \log_2 11,$$

откъдето получаваме

$$\begin{aligned} 14 &\equiv m_1 + m_4 + m_5 (\mathrm{mod}\, 540); \\ 81 &\equiv m_1 + m_2 + 2m_4 (\mathrm{mod}\, 540); \\ 207 &\equiv 2m_3 + m_5 (\mathrm{mod}\, 540); \\ 214 &\equiv m_3 + m_4 (\mathrm{mod}\, 540); \\ 300 &\equiv 5m_1 + m - 5 (\mathrm{mod}\, 540). \end{aligned}$$

Решавайки тази система получаваме

$$m_1 = \log_2 2 = 1, m_2 = \log_2 3 = 104, m_3 \log_2 5 = 496, m_4 = \log_2 7 = 258, m_5 = \log_2 11 = 295,$$

или еквивалентно

$$2^1 \equiv 2(\mathrm{mod}541), 2^{104} \equiv 3(\mathrm{mod}541), 2^{496} \equiv 5(\mathrm{mod}541), 2^{258} \equiv 7(\mathrm{mod}541), 2^{295} \equiv 11(\mathrm{mod}541).$$

Ако се окаже, че намерените линейни сравнения не са линейно независими, то заместваме някои от тях снови. Да намерим сега решението m на сравнението $2^m \equiv 345(\mathrm{mod}541)$. Пресмятаме последователно

$$\begin{aligned} 345 &\equiv 3^1 \cdot 5^1 \cdot 23^1 (\mathrm{mod}\, 541); \\ 2^2 \cdot 345 &\equiv 2^1 \cdot 149^1 (\mathrm{mod}\, 541); \\ 2^{100} \cdot 345 &\equiv 3^2 \cdot 41^1 (\mathrm{mod}\, 541); \\ 2^{13} \cdot 345 &\equiv 2^3 \cdot 7^1 (\mathrm{mod}\, 541). \end{aligned}$$

Оттук

$$13 + m \equiv 3m_1 + 1m_4 \equiv 3 \cdot 1 + 258 \equiv 261 (\mathrm{mod}\, 540),$$

следователно решението на $2^m \equiv 345 (\mathrm{mod}\, 540)$ е $m \equiv 248 (\mathrm{mod}\, 540)$.

Ще отбележим, че броят на елементите, от G, които могат да бъдат представени като произведение на елементи от S е 142. Следователно очакваме средно четири опита (избора на r) за да попаднем на $g^r h$, което може да бъде изразено като произведение на елементите от $\{2, 3, 5, 7, 11\}$.