 DOI: https://doi.org/10.12349/iser.v4i1.959

Research on the Interval Discrete Log Problem Solving Algorithm

Xiangfu Meng[1] Tianyu Sun[2] Jiaqi Hu[3]

1.Unit 93010, Shenyang, Liaoning, 110000, China
2.Unit 95988, Changchun, Jilin, 130062, China
3.Zhengzhou University, Zhengzhou, Henan, 450000, China

Abstract

Discrete logarithm problem is a basic problem in cryptography. This paper first introduces the significance of studying the discrete logarithm problem, several exponential algorithms and research status, and then discusses the idea of kangaroo algorithm for interval discrete logarithm problem. Several ways to improve. Combined with the corresponding examples, the specific solution idea, the solution method and the selection basis of the corresponding parameters are given. Finally, the advantages and disadvantages of several exponential algorithms in solving the discrete logarithm problem of the interval are simply compared, and the prospects for solving the discrete logarithm problem of the interval kangaroo algorithm are also presented.

Keywords

interval discrete logarithm; exponent algorithm; kangaroo algorithm

区间离散对数问题求解算法的研究

孟祥福[1] 孙天宇[2] 胡佳奇[3]

1.93010 部队，中国 · 辽宁 沈阳 110000
2.95988 部队，中国 · 吉林 长春 130062
3. 郑州大学，中国 · 河南 郑州 450000

摘 要

离散对数问题是密码学中基础性的困难问题，论文首先介绍了研究离散对数问题的意义，几种求解的指数算法以及研究现状，接着针对区间离散对数问题，论述了kangaroo算法的思想及几种改进方式。并结合相应的实例，给出了具体的求解思想，求解方法，相应参数的选取依据。最后，简单对比了几种指数算法在求解区间离散对数问题中的优劣并对kangaroo算法求解区间离散对数问题的前景提出了展望。

关键词

区间离散对数；指数算法；kangaroo算法

1 概述

1.1 研究背景

随着密码学的发展，在解决实际问题中出现了一些特殊的离散对数问题，比如在某个区间上的离散对数问题[1]。该问题的出现是由于在实际应用中，如果没有进行有效的保护措施，敌手可能通过某些攻击手段获取关于离散对数的某些比特位信息，从而推断出待求解离散对数所在的区间范围，之后敌手在进行离散对数求解时只需要在该区间范围内求解即可。所以，求解区间上的离散对数问题在实际的密码学系统中有很重要的用途，是现代密码学中一个非常重要的问题。

经典离散对数问题 (DLP) 是指 : 给定群元素 g，y，计算 x 使得 $h=gx$。许多密码系统和协议 (如 *Diffie-Hellman* 密钥交换协议[2]、*ELGamal* 加密系统) 的安全性都是基于离散对数问题的困难性。目前存在的解决经典离散对数问题的算法有大步小步算法、*Pollard's rho* 算法以及 *Pollard's kangaroos* 算法等指数算法。

区间上的离散对数问题 (IDLP) 是指 : 给定群元素 g，y，已知待求解的离散对数 x 在区间 $[a, b]$，计算 $a \leqslant x \leqslant b$ 使得 $y=gx(b$ 小于 g 的阶)。这一基础性的计算问题在实际中有着广泛应用，如 *c-bit* 指数的 *DLP*(*c-DLSE*)[3]、*Boneh-Goh-Nissim* 同态加密方案、有限域上的椭圆曲线或者阿贝尔变体的点个数计算、强 *Diffie-Hellman* 问题的分析、边信道或者小子群攻击。

利用大步小步算法解决离散对数问题最早是由 *Shanks* 提出，这种算法求解速度要比 *Kangaroo* 算法更快。最快的大步小步算法是由 *Pollard* 提出，需要进行 $4/3\sqrt{N}$ 次群运算来解决区间离散对数问题。然而这个算法不能用于大指数，

【作者简介】孟祥福（1996–），男，蒙古族，中国吉林松原人，助理工程师，从事信息安全研究。

因为该算法需要存储空间为$O(\sqrt{n})$。因此该算法需要较大的存储空间并且查表的速度对算法的影响很大，所以大步小步算法只适合在群的规模较小时使用。一般在一个确定的群中解决区间离散对数问题时，指数所在范围为 N，若 $N < 230$ 通常使用大步小步算法，若 $N > 230$ 通常使用袋鼠算法。

使用 *Pollard's rho* 算法计算离散对数，最早出 *John M. Pollard* 在 1975 年所写的论文，算法的时间复杂度是$O(\sqrt{n})$，但只需要 *O(1)* 的空间存储。虽然看起来 *Pollard's rho* 算法与大步小步算法的计算复杂度基本相同，但是大步小步算法中涉及到大量的查表运算和存储过程使得在实践中大步小步算法的速度比 *Pollard's rho* 算法慢很多。

并且以上两种算法并没有将离散对数所在的区间范围，这一关键指标利用好。而论文所主要研究的 *Pollard's kangaroos* 算法是最好的通用低存储算法来解决区间离散对数问题并且 *Pollard's kangaroos* 算法对于传统的离散对数问题也有很好的应用。

1.2 研究现状

从区间离散对数问题的提出，至今已经引起了人们的广泛关注，并取得了一系列进展，对于袋鼠算法有了许多不同的改进方案。

袋鼠算法最早由 *Pollard* 提出在 1978 年，估计平均运算时间为$3.3\sqrt{N}$ 次群运算。进一步的改进是由 *van Oorshot* 和 *Wiener* 完成的，是在多项式大小的存储空间上通过$(2+o(1))\sqrt{N}$次群运算解决区间大小为 N 的离散对数问题 . 在之后的 15 年中这一直是最快的袋鼠算法。直到 *Galbraith, Pollard* 和 *Ruprai* 提出了三只袋鼠和四只袋鼠模型，它们分别需要进行$(1.818+o(1))\sqrt{N}$次和$(1.714+o(1))\sqrt{N}$次群运算来解决区间离散对数问题。

目前最快的袋鼠算法是 *Galbraith, Pollard* 和 *Ruprai* 的 4 只袋鼠跳跃方法。在大小为 N 的区间上，该算法平均运行时间为$(1.714+o(1))\sqrt{N}$次群运算，并且需要$O(log(N))$ 存储空间。但是没有证明来说明四只袋鼠算法是最快的，这也是袋鼠算法体系的一大缺陷。

2 基础知识

2.1 Diffie–Hellman 密钥交换协议

现代公钥密码体制的安全性理论是建立在复杂性理论基础上的。在这个基础之上，公钥密码体制的安全性是基于某些假设有条件安全的，这些假设就是假设某些问题是困难的。Diffie-Hellman 密钥交换协议的安全性就是基于求解离散岁数问题的困难性。

Alice 与 Bob 交换公钥并生成相同的共享密钥。Diffie-Hellman 密钥交换算法的有效性依赖于计算离散对数的难度。算法描述如下：

①有两个全局公开的参数，一个素数 q 和一个整数 g，g 是 q 的一个原根。

②假设用户 A 和 B 希望交换一个密钥，用户 A 选择一个作为私有密钥的随机数 $XA \square q$，并计算公开密钥 $Y_A \equiv g^{X_A} \bmod q$。A 对 *XA* 的值保密存放而使 *YA* 能被 B 公开获得。类似地，用户 B 选择一个私有的随机数 $XB \square q$，并计算公开密钥$Y_B \equiv g^{X_B} \bmod q$。B 对 *XB* 的值保密存放而使 *YB* 能被 A 公开获得。

③用户 A 产生共享秘密密钥的计算方式是$K \equiv {Y_B}^{X_A} \bmod q$。同样，用户 B 产生共享秘密密钥的计算是$K \equiv {Y_A}^{X_B} \bmod q$。

这两个计算产生相同的结果：

$$\begin{aligned} K &= (Y_B)^{X_A} \bmod q \\ &= (g^{X_B} \bmod q)^{X_A} \bmod q \\ &= (g^{X_B})^{X_A} \bmod q \\ &= g^{X_A X_B} \bmod q \\ &= (g^{X_A})^{X_B} \bmod q \\ &= (g^{X_A} \bmod q)^{X_B} \bmod q \\ &= (Y_A)^{X_B} \bmod q \end{aligned}$$

作为攻击者我们要知道 Alice 与 Bob 之间的通信内容，我们就要求解一个离散对数问题。我们只要知道 *XA* 或者 *XB* 中的一个我们就可以获得通信密钥 K，进而获取通信内容的明文信息。

2.2 Pollard's Rho 算法

先定义一个 Hash 函数，选定一个群 G，群的阶 ||G||=q 为素数，随机选取生成元g，h。输入$x, y \in \{1, \cdots, q\}$，定义 Hash 函数：H(x,y)= g x g y。并且我们知道 H(x,y) 寻找碰撞与求解群 G 中的离散对数问题等价。也就是说，如果能找到 *H(x,y)* 的一个碰撞，就能够解决群 *G* 中的离散对数问题。

引理：设数列 $x_1, \cdots, xq$ 满足关系 xm=f(xm − 1)，xI=XJ（其中 1≤I≤J≤q），则存在某个 i < J，使得 xi=x2i。

证明：

令周期 M=J − I，则 xj=xj+kM 对于所有整数 j≥I，k≥0 都成立。

取 i 是满足 i≥I 的周期 M 的最小整数倍：i=kM。

由此可以知道 i < J，否则若 i > J，且 J − I=M，这与取 i 是满足 i≥I 的周期 M 的最小整数倍：i=kM 相矛盾，所以 i < J。

又因为 i≥I，且 2i − i=i 是 M 的整数倍，故 xi=x2i。

证毕。

寻找一对碰撞：$g^i h^j = g^k h^l$。

选择合适的随机函数$f: {F_p}^* \to {F_p}^*$。

要求函数具有较好的随机性，并且较为容易追踪指数的轨迹。

Pollard 建议的函数：

$$f(x) = \begin{cases} gx & \text{if } 0 \le x < p/3, \\ x^2 & \text{if } p/3 \le x \le 2p/3, \\ hx & \text{if } 2p/3 \le x < p. \end{cases}$$

虽然该函数不能从理论上证明具有足够的随机性，但是在实践中有非常好的效果。

迭代计算 xi 的值：

初始值：$x0=1$，$\alpha_0=\beta_0=0$

$$x_i=\left(\underbrace{fff\cdots f}_{i\uparrow f}\right)(1)=g^{\alpha_i}h^{\beta_i}$$

指数 α_i 的迭代关系：

$$\alpha_{i+1}=\begin{cases}\alpha_i+1 & \text{if } 0\leqslant x<p/3,\\ 2\alpha_i & \text{if } p/3\leqslant x<2p/3,\\ \alpha_i & \text{if } 2p/3\leqslant x<p.\end{cases}$$

指数β_i的迭代关系：

$$\beta_{i+1}=\begin{cases}\beta_i & \text{if } 0\leqslant x<p/3,\\ 2\beta_i & \text{if } p/3\leqslant x<2p/3,\\ \beta_i+1 & \text{if } 2p/3\leqslant x<p.\end{cases}$$

根据费马定理且 p 为素数，我们知道$g^{p-1}=1\,mod\,p$；$h^{p-1}=1\,mod\,p$。

所以每一次迭代过程中 α_i 和β_i均需模 p-1。

假定最终找到一组碰撞：$x2i=xi$。

即：

$$g^{\alpha_i}h^{\beta_i}=g^{\gamma_i}h^{\delta_i}$$

取$u\equiv\alpha_i-\gamma_i(mod\,p-1)$，$v\equiv\delta_i-\beta_i(mod\,p-1)$。

$$g^u=h^v\ in\ F_p$$

两边同时以 g 为底取对数得：

$$v\log_g(h)\equiv u(mod\,p-1)$$

两侧同时乘以 v 的逆，就可计算出离散对数。

但这是在$gcd(v,p-1)=1$的情况下，v 才存在逆元。

当$d=gcd(v,p-1)\geq 2$时，计算s 满足$sv\equiv d\ mod\ p-1$。

可得方程：$d\cdot\log_g(h)\equiv w\ \text{mod}\ p-1$，$w\equiv su\ mod\ p-1$。

解为：$log_g(h)\in\left\{\frac{w}{d}+k\frac{p-1}{d}:k=0,1,2,\cdots,d-1\right\}$。

3 区间离散对数问题求解—*kangaroos* 算法及改进

3.1 两只袋鼠算法及改进

3.1.1 经典袋鼠算法

简述经典 Pollard’s kangaroos 算法：

区间离散对数问题：给定循环群 G，g，h 是群中的元素，计算 x 使得$h=g^x(a\leq x\leq b)$。对于求解该问题的经典 Pollard’s kangaroos 算法，该算法为指数算法占用极少的存储空间。

在袋鼠算法模型中，我们定义了两个角色分别是驯化的袋鼠和野生的袋鼠。驯化的袋鼠的跳跃过程与在玩纸牌游戏中第一次翻牌过程类似，每翻开一张牌都相当于驯化的袋鼠在落点处做了一个陷阱。之后的第二次翻牌过程与野生袋鼠的跳跃过程类似，第二次翻牌过程只要有一张牌是第一次所翻过的，那么两次翻牌过程一定最终会落在同一张牌上，野生袋鼠每次跳跃中只要存在一次落在驯化袋鼠的“陷阱”中，那么就一定会发生碰撞，问题也就可解了。

两只袋鼠都在以 g 为生成元的群 G 中跳跃，均代表着群里的元素。驯化袋鼠的起点是$t_0=g^{(a+b)/2}$，野生袋鼠跳跃的起点为 $w0=h$。两个起点的指数都在区间 $[a,b]$ 中，驯化的袋鼠在区间的中点，野生袋鼠的起点在$x=log_g h$。之所以称其中的一只为野生袋鼠是因为我们不知道 x 的值是多少，也不知道其跳跃的精确位置。算法最终的目的是发现一个野生袋鼠和驯化袋鼠之间的一个碰撞，根据碰撞时两只袋鼠所在的位置和跳跃的距离信息可以推算出 x 的值。

可以将整个过程分为三部分：

Stage1：第一阶段是从两只袋鼠开始跳跃开始至跳跃前处于后面的袋鼠 B 到达跳跃前处于前面的袋鼠 F 的起点位置。如果我们定义指数 x 所处的区间长度 b — a=N 是一定的，那么 B 和 F 在开始跳跃前的平均距离为 N/4。因此，第一阶段跳跃的步数是 N/4m（m 是袋鼠跳跃的平均步长）。

Stage2：第二阶段是从第一阶段结束到 B 跳跃落在 F 跳跃所经历过的群元素上，当跳跃前处于后面的袋鼠 B 到达跳跃前处于前面的袋鼠 F 的起点位置时起，B 将进入一个 F 所走过的区域，在这个区域中的 1/m 个群元素是 F 已经走过的。因此 B 每次跳跃落在的群元素是在 F 的路线中的可能性是 1/m。因此，我们可以预测出 B 经过 m 次跳跃所到的元素是 F 经历过的。

Stage3：第三阶段是从第二阶段结束到 B 落入可区分点集合中。对于 G 中的每个元素是可区分点集合中的元素的可能性是$c\,log\,(N)/\sqrt{N}$，可以知道 B 在这一阶段会经历$\frac{1}{c\,log\,(N)/\sqrt{N}}=\sqrt{N}/c\,log\,(N)$步运算。

由此我们能够预测出这个算法需要$N/4m+m+\sqrt{N}/c\,log\,(N)$步来解决区间离散对数问题。每一步需要进行两次群运算，并且对于 m 最佳的选择是$m=\sqrt{N}/2$，由此可以得出理想的运行时间是$(2+1/c\,log\,(N))\sqrt{N}=(2+o(1))\sqrt{N}$次群运算。

3.1.2 袋鼠算法与 rho 算法

袋鼠算法也被称为 lambda 算法，但是 *Pollard’s Rho* 算法的并行化已经越来越流行，*Pollard’s Rho* 算法有时也被成为 lambda 算法。所以有时会将两种算法混淆这里我们将讨论它们的联系和区别。

通过前面的内容我们已经知道 *Pollard’s Rho* 算法在求离散对数问题时是通过计算出一个在群 G 中的序列 $\{yk\}$，通过选择一个初始项$y_0\in G$，按照规定的迭代规则 $yk+1=F(yk)$，其中 $F:G\to G$ 是一个伪随机映射。这样所定义的序列 $\{yk\}$ 最终一定是周期的。如果将整个过程画出从底部开始，并以有个循环结束，最终会形成一个希腊字母 ρ（rho）的形状。

在袋鼠算法中，如果也将两个袋鼠序列画在纸上，$\{tk\}$ 从左下角开始画，$\{wk\}$ 从右下角开始画，当碰撞发生时两个序列曲线相交，如果我们将碰撞的区域放大我们可以看到一个希腊字母λ（lambda）。

袋鼠算法和λ算法所生成的序列都按照如下迭代方式：$zk+1=zk*m$。

正如前面所讨论的 Rho 算法是基于生日悖论的思想，它的特征是实在群 G 中随机“游走”。群元素通过与 $m_i = g^{u_i}$ 相乘实现随机“游走”，这里 ui 被认为是从 $[1, ord\ g]$中随机选择，平均值为$(ord\ g)/2$。

在袋鼠算法中我们还是计算$m_i = g^{s_i}$，si 可以看作是很小的距离，大约是$\sqrt{b-a}$ $(a \le x = log_g\ h \le b)$。所以不用再整个群中随机“游走”，我们将跳跃距离看作是平均值为$\sqrt{b-a}/2$的小整数。当 a=0 且$b = ord\ g$时，说明没有关于离散对数位数的任何信息可用，使用袋鼠算法求解比使用 *Pollard's Rho* 算法规模更小。

3.2 三只袋鼠算法及改进

三只袋鼠：

这个模型定义了三种不同的袋鼠，W1，W2 和 T。三种不同的袋鼠分别有自己的群元素形式$g^p h, g^r h^{-1}$和$g^q (p, q, r \in \mathbb{N})$。当任何一对袋鼠发声碰撞，我们就可以通过$g^p h, g^r h^{-1}$和$g$ q 的信息求解出 x。

如果我们得到$g^p h$=g^q，那么 x=p − q。

如果我们得到$g^r h^{-1}$=g^q，那么 x=r − q。

如果我们得到$g^p h$=$g^q h^{-1}$,那么$x = 2^{-1}(q-p)\ mod\ (|g|)$。

因此任何一对袋鼠只要其发生碰撞就能够求解区间离散对数问题。

定义 W1 的跳跃：

W1 开始跳跃的点处在群元素$W_{1,\ 0} = g^{-N/2} h$。

W1 跳跃的迭代过程是$W_{1,i+1} = W_{1,i} g^{H(W_{1,i})}$。

定义 W2 的跳跃：

$W2$ 开始跳跃的点处在群元素$W_{2,\ 0} = g^{N/2} h^{-1}$。

$W2$ 跳跃的迭代过程是$W_{2,i+1} = W_{2,i} g^{H(W_{2,i})}$。

定义 T 的跳跃：

T 开始跳跃的点处在群元素$T_0 = g^{3N/10}$。

T 跳跃的迭代过程是$T_{i+1} = T_i g^{H(T_i)}$。

根据以上的定义我们可以看出 $W1$、$W2$、T 的起始点的指数分别为 $x - N/2$、$N/2 - x$、$3N/10$ 。我们可以定义 T 和 $W1$ 之间的距离为 $dT,w1$，定义 T 和 $W2$ 之间的距离为 $dT,w2$，定义 $W2$ 和 $W1$ 之间的距离为 $dw2,w1$。

由此可得：

$$d_{T,w_1}(x) = |(x - N/2) - (3N/10)| = |x - 4N/5|$$
$$d_{T,w_2}(x) = |N/5 - x|$$
$$d_{w_2,w_1}(x) = |2x - N|$$

再定$d_C(x) = min\{d_{T,w_1}(x), d_{T,w_2}(x), d_{w_2,w_1}(x)\}$。

在这个模型中袋鼠的起始位置是所有最近的袋鼠的平均距离最小值。

3.3 四只袋鼠算法及改进

四只袋鼠算法很简单，是在三只袋鼠算法的基础上进行了巧妙地扩充。所以说在研究四只袋鼠算法时，我们还是先从三只袋鼠算法开始。就像三只袋鼠算法一样，先从三只袋鼠 T1，W1，W2 开始，它们跳跃前的位置分别为 3N/10, x — N/2 — x。在这基础上，添加一个额外的驯服袋鼠 T2，定义它的起始位置为（3N/10）+10。

很容易看出 W1，W2 的起始点具有相同的奇偶性，并且 T1，T2 中一定存在一个与 W1，W2 的奇偶性相同。因此，当跳跃的步长确定时（即袋鼠所处位置到下一步所跳跃的步长的函数 F 确定），在整个跳跃过程中，两只野生袋鼠和某一只驯化袋鼠将能够发生碰撞，也说明另外一只驯化袋鼠不能和其他任何一只袋鼠发生碰撞。因此，能够碰撞的三只袋鼠是与三只袋鼠算法完全相同的。

在三只袋鼠算法中，发生一次碰撞也就是解决区间离散对数问题需要$(1.818 + o(1))\sqrt{N}$次群运算。然而，在四只袋鼠算法中，因为有一个“无用”的驯化袋鼠和另外三个“有用”的袋鼠需要同样多的群运算，所以使用四只袋鼠算法解决区间离散对数问题需要群运算数为$(1+3)/3\ (1.818 + o(1))\sqrt{N} = (1.714 + o(1))\sqrt{N}$次。

3.4 三种袋鼠算法对比

在以上研究的基础上，以如下例子为衡量标准，对比三种算法的效率及实际运行速度与理论运行速度对比并进行分析。

群信息如下：

p=1147037487492527565811666350723216140208665025845389627453499167689899926264158151910107474064236984823329423985151921234184433734711989987439145632978562 3

q=335062023296420808191071248367701059461

g=6229523353339612969781592660847410858898813587384599399782901799360636355667402585551677830090585673979634661031400826474866116573508115606305870131833 57

y=9388897478013399550694114614498790691034187453089355259602614074132918843899833277397448144245883225611726912025846772975325932794909655215329941809013733

求解出相应的指数 x。

分别用三种袋鼠算法进行求解，将对比汇总如表 1 所示。

三种算法的理论运行时间比值为 2 ： 1.818 ： 1.714，但是在实际运行中运行时间的比值为 2 ： 1.834 ： 1.746，可以看出比值变小了。分析可能是因为随着袋鼠数量的增加，每次跳跃后附加的，判断是否加入可区分集和可区分集中的值变多，增加了查找次数，使运算时间增加。也可能是由于算法使用 Python 语言编写，运行效率较 C 语言和汇编语言相比更慢一些，也会影响到算法效率。但是相对于两只袋鼠算法效率还是有很大提升。

表 1 对比汇总表

算法类型	两只袋鼠	三只袋鼠	四只袋鼠
袋鼠类型	T, W	T, W_1, W_2	$T_1, T_2\ W_1, W_2$
起始点位置	$T_0 = g^{N/2}$ $W_0 = h$	$W_{1,0} = g^{-N/2}h\ W_{2,0}=g^{N/2}h^{-1}\ T_0 = g^{3N/10}$	$W_{1,0}=g^{-N/2}h\ W_{2,0} = g^{N/2}h^{-1}\ T_{1,0} = g^{3N/10}$ $T_{2,0} = (3N/10) + 1$
理论运行时间 (单位 : 次群运算)	$(2 + o(1))\sqrt{N}$	$(1.818 + o(1))\sqrt{N}$	$(1.714 + o(1))\sqrt{N}$
实际运行时间 (单位 : 秒)	3471	3182	2995

参考文献

[1] 王瑶,吕克伟.关于区间上离散对数问题的改进算法[J].密码学报,2015,2(6):570-582.

[2] Diffie W, Hellman M E. New directions in cryptography[J]. IEEE Transactions on Information Theory, 1976, 22(6):644-654.

[3] Gennaro R. An Improved Pseudo-random Generator Based on Discrete Log.[C]// International Cryptology Conference on Advances in Cryptology, Springer-Verlag, 2000:469-481.