密码学二级

分级通关系列教程

素性判断

- 561是不是素数?
- 1000000000000000000000000000000000000000000000000000000000000000000000000000000000000000028900000002670000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000077163
- 95887882817631642349993491222159715578101423142394742573793693888302183209364650259690575352235369168922772321655833193176144076494109674234144894625737783987445672891333584844250311646967000687965463
- 费马小定理: p为素数，(a,p)=1,a^(p-1)=1(modp)
- Miller-Rabin
- AKS

离散对数

- 素数$p = 17$,
- $2^i(modp)(i = 0,1, \dots, 15)$: $[1, 2, 4, 8, 16, 15, 13, 9, 1, 2, 4, 8, 16, 15, 13, 9]$
- $3^i(modp)(i = 0,1, \dots, 15)$: $[1, 3, 9, 10, 13, 5, 15, 11, 16, 14, 8, 7, 4, 12, 2, 6]$
- 选取 g =3
- 已知a $= 15$, 求i 满足$g^i \equiv a(modp)$(离散对数问题)
- 群G=$< g >$=$\{g^0, g^1, \dots, g^{n-1}\}$
- n很大

DH协议

- 大素数p,本原元g
  - *Alice*→*Bob* ,随机选择$0 < x < p$,发送$g^x (mod p)$
  - *Bob*→*Alice* ,随机选择$0 < y < p$,发送$g^y (mod p)$
  - *Alice,Bob*分别计算$(g^y)^x (mod p), (g^x)^y (mod p)$
- 两者计算的值都为g^{xy} $(mod p)$,所以相等，为共享密钥
- 第三者从$g^x(mod p)$不能计算x ，从$g^y(mod p)$不能计算y （离散对数难解性)

小规模DLP

- Shanks（BSGS-Baby steps giant steps）
- p= 39859248359303,y= 324987538,z= 3498594389434，求满足下式的最小正整数x.

求解关于x的方程

$$y^x = z(mod\,p)$$

其中$(y,p)=1$

做法并不难，我们把x写成一个$am-b$的形式

那么，原式变成了

$y^{am} = zy^b(mod\,p)$

我们求出所有b可能的取值(0~m-1)，并且计算右边的值

同时用哈希或者map之类的东西存起来，方便查询

对于左边，我们可以枚举所有可能的a，然后直接查右边的值有没有相等的即可

复杂度是$O(max(m,p/m))$

不难证明$m=\sqrt(p)$时复杂度最优

所以$bsgs$算法的复杂度是$O(\sqrt(p))$

小规模DLP

- Pollard-ρ

设 $<\alpha>$ 是由 α 生成的循环群，阶为 n， $\beta\in<\alpha>$，求 $ind_{\alpha}\beta$，即 $0\le i<n$，使得 $0\le i<n,\beta=\alpha^{i}$。求解思路是构造一个随机序列 $x_1,x_2,\ldots,x_m$，其中每个元素 $x_i=\alpha^{a_i}\beta^{b_i}$，若存在两个元素 x_i,x_j 使得 $x_i=x_j,i\neq j$，那么 $\alpha^{a_i}\beta^{b_i}=\alpha^{a_j}\beta^{b_j}$，即 $\beta^{b_i-b_j}=\alpha^{a_j-a_i}$，进一步得到 $ind_{\alpha}\beta\equiv(b_i-b_j)^{-1}(a_j-a_i)(\bmod n)$。

小规模DLP

- Pollard-ρ

首先将 $<\alpha>$ 分成元素个数大致相等的三个两两互不相交的集合，即 $<\alpha> = S_1 \cup S_2 \cup S_3$

构造函数：$f(x,a,b)=\begin{cases}(\beta x, a, b+1), x\in S_1\\(x^2, 2a, 2b), x\in S_2\\(\alpha x, a+1, b), x\in S_3\end{cases}$，

根据 f 函数的构造，当自变量 (x,a,b) 满足 $x=\alpha^a\beta^b$ 时，其函数值也满足这个关系。

由 Pollard ρ方法，我们选取初始值 (x_1,a,b) 满足 $x_1=\alpha^a\beta^b$，例如 $(1,0,0)$，迭代调用 f 函数，直到找到 $x_i = x_{2i}, i\geq 1$，根据上面的讨论，此时 $ind_\alpha\beta \equiv (b_i - b_{2i})^{-1}(a_{2i}-a_i)(\bmod n)$。

实用工具

- CADO-NFS：参考：http://cado-nfs.gforge.inria.fr/
- yafu:https://sourceforge.net/p/yafu/wiki/Home/

HASHCAT https://blog.csdn.net/SHIGUANGTUJING/article/details/90074614

- -a 指定要使用的破解模式，其值参考后面对参数。“-a 0”字典攻击，“-a 1” 组合攻击；“-a 3”掩码攻击。
- -m 指定要破解的hash类型，如果不指定类型，则默认是MD5
- -o 指定破解成功后的hash及所对应的明文密码的存放位置,可以用它把破解成功的hash写到指定的文件中
- --force 忽略破解过程中的警告信息,跑单条hash可能需要加上此选项
- --show 显示已经破解的hash及该hash所对应的明文
- --increment 启用增量破解模式,你可以利用此模式让hashcat在指定的密码长度范围内执行破解过程
- --increment-min 密码最小长度,后面直接等于一个整数即可,配置increment模式一起使用
- --increment-max 密码最大长度,同上
- --outfile-format 指定破解结果的输出格式id,默认是3
- --username 忽略hash文件中的指定的用户名,在破解linux系统用户密码hash可能会用到
- --remove 删除已被破解成功的hash
- -r 使用自定义破解规则

HASHCAT

破译8位数字SHA1:
d6cfe5e76c8347bc803168fe861f69fcc69cc79c
破译8位小写字母加数字SHA1:
eb1d44e685e37f25e877d11f2c557ddc76ae9269