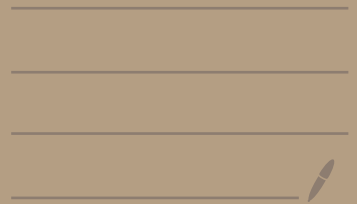
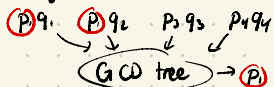


Attacking Discrete-Log



Last Time:

- Mining P's & Q's



- Coppersmith's attack: Infixion edition

optimized keygen:

$$p \leftarrow k \cdot M + 65537^a \% M$$

$$q \leftarrow l \cdot M + 65537^b \% M$$

Bad Randomness \Rightarrow Broken Crypto

Today: Dimer attacks on discrete log

- Generic group attacks

- Baby-step, giant-step

- Pollard's rho

- Shoup's lower bound

matching!

- \mathbb{Z}_p^* attack:

- index calculus

Generic Discrete-log Attack

- Fix group G , order q , generator g .

- given: $h = g^x$, $x \in \mathbb{Z}_q$

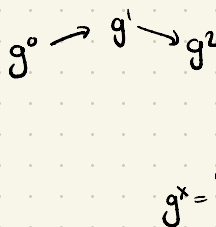
- find: x .

Warm-up Attack: Naive

Brute-force search:

for $i \in \{0, \dots, q-1\}$:

if $g^i = h$: return i



uses "baby steps": $g^i \rightarrow g^{i+1}$ but are

"giant steps": $g \rightarrow g^n$

cheap too: $\leq \log_2 n$ group-ops!

Baby-Step, Giant-Step [Shanks'71]

Idea: precompute \sqrt{q} -spaced signposts

Offline phase (G, g, q):

build M

$\rightarrow O(\sqrt{q} \log q)$ g-ops

$\rightarrow O(\sqrt{q} \log q)$ bits in M what are we assuming?

Online phase (h):

find $j \in \{0, \dots, \sqrt{q}\}$

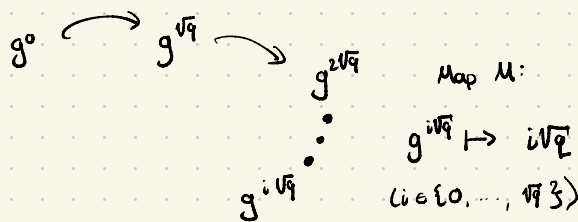
s.t. $hg^j \in M$

and return $M[hg^j] - j$

$$M[hg^j] - j = \log_g(hg^j) - j = \log_g h + j \log_g g - j = \log_g h \quad \checkmark$$

Time: $O(\sqrt{q} \log q)$

\Rightarrow b/c table lookup (w/ tree)



Space is bottleneck

Take $q = 2^{80}$ (real: $q = 2^{256}$)

time: $\frac{3}{2} \sqrt{q} \log q$ g-ops = $\frac{3}{2} 2^{40} \cdot 40$ g-ops = 94 cpu-years

space: \sqrt{q} g-elements = $2^{40} \cdot 82 B \cdot 2 \approx 70 TB$ (in ea table)

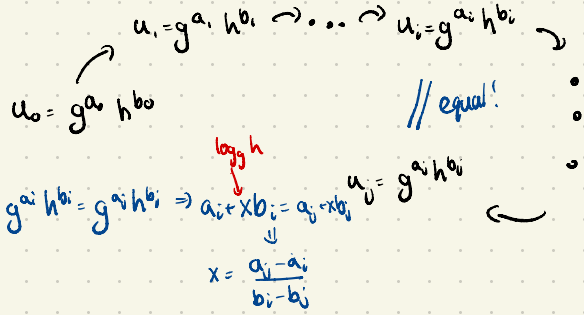
can we reduce space?

parallelize!

(cann 25519-dalek: 45 μs /g-op)

Pollard's Rho Algorithm

Idea: random walk + cycle finding!



For a random transition $O(\sqrt{q})$ steps give ν cycle w/ all but negl. prob.

Need:

1. a pseudo-random transition fn
2. cycle finding.

1. Pseudo-random steps

- can't hash $H(g^a h^b) \rightarrow g^a$
- lose $d \log$ information for g !
- simple alternative

• split G into random $G = S_0 \cup S_1 \cup S_2$

$$\text{step}(u = g^a h^b) = \begin{cases} g^{2a} h^b & u \in S_0 \\ g^{a+1} h^b & u \in S_1 \\ g^a h^{b+1} & u \in S_2 \end{cases}$$

effective when split is unrelated to G 's structure

2. Cycle Finding: Floyd's tortoise & hare [Knuth '69]

Problem:

Given an infinite sequence x_1, x_2, \dots that eventually cycles.

find $i > j$ s.t. $x_i = x_j$

Solution:

two pointers: $t_i = x_i$ ← tortoise

$h_i = x_{2i}$ ← hare

distance between them: i

for cycle of length l , $t_j = h_j$ for first j divisible by l after t_i enters the cycle!

runtime: $O(\text{distance to cycle closure})$

3. Analysis

$O(\sqrt{q})$ time $\leftarrow \Rightarrow 1 - \text{negl}(l)$

overwhelming success probability

$O(1)$ space

\Rightarrow 2 pointers!

Can we do better?

Shoup's Lower Bound [Shoup '97]

Any group-generic $d \log$ attack requires $\Omega(\sqrt{q})$ g -ops to attain non-negl. success probability.

\rightarrow Baby-step, Giant-step & Pollard's Rho are time-optimal ($\ast \log$ factors)

\rightarrow Pollard's Rho is space-optimal why? $\ddot{\smile}$

\rightarrow uses Generic Group Model

\rightarrow Really nice proof. See Thm of today's reading.

Next: non-generic attacks!

Part II: Non-generic attacks

Warmup: $(\mathbb{Z}_p, +)$

Consider group $(\mathbb{Z}_p, +)$ ^{prime}

- elements in $\mathbb{Z}_p = \{0, 1, \dots, p-1\}$
- group operation: $a, b \mapsto (a+b) \% p$
- d-log problem:

given $g, h \in \mathbb{Z}_p$, find x such that $\underbrace{g + \dots + g}_x = h$

- easy solution: division!
if $h/g = k$, then $\underbrace{g + \dots + g}_k = h$ (in \mathbb{Z}_p)
*in \mathbb{Z}_p^**

Index calculus (\mathbb{Z}_p, \times) (aka \mathbb{Z}_p^*)

- \mathbb{Z}_p^* :
- elements from $\{1, 2, \dots, p-1\}$
 - under $\times \pmod p$
 - assume (today) that $q = \frac{p-1}{2}$ is prime ^{"safe prime"}
 - and that g generates an order q multiplicative subgroup $(\{g, g^2, g^3, \dots, g^{q-1}\} = \mathbb{Z}_p^*)$
 - example: \mathbb{Z}_7^*
 $p=7, q=3, g=2$ generates $\{2, 4, 1\}$

Our goal:

- a sub-exponential (but super-poly) attack
- define $L_N(\alpha, c) = \exp(c \log^{\alpha} N (\log \log N)^{-c})$, so
 - $L_N(0, c) = (\log N)^c$ poly $(\log N)$ = poly $(N$'s size)
 - $L_N(1, c) = N^c$ exp $(\log N)$
 - $L_N(1/2, c) = \exp(c \sqrt{\log N} \cdot \log \log N)$ "sub-exponential"
- We'll build a $L_2(1/2, 3)$ attack

Plan of attack:

- Let $B = \{2, 3, 5, \dots, p_6\}$ contain all primes $\leq \beta$ ← to be set later
a "factorization basis"
ex: $120 = 2^3 \cdot 3 \cdot 5$ factors in $\{2, 3, 5\}$
 $140 = 2^2 \cdot 5 \cdot 7$ does not

2. Compute $\log_g p_i$ for $i \in [t]$

↳ we'll explain this next

3. Compute $\log_g h$ from $\{\log_g p_i\}$

- use "random self-reducibility"
- sample $r \in \mathbb{Z}_2$ until hg^r factors in B . That is, $gh^r = \prod p_i^{e_i}$
→ $\log_g(hg^r) = \sum e_i \log_g p_i$ → $\log_g h = -r + \sum e_i \log_g p_i$
- How many r do we need to sample?
Math fact 2 implies: $\Pr[\beta\text{-smooth}] \approx 1/u^u$ $u = \frac{\log q}{\log \beta}$
w/ $\beta = L_2(1/2, 1)$, you can show $\Pr[\beta\text{-smooth}] \geq \beta$
⇒ expected # of r -values $\leq \beta$.

How much work per r -value?

size of $B \rightarrow \frac{\beta}{\log \beta} \cdot \text{poly log } \beta = \tilde{O}(\beta)$
↳ division time
⇒ total runtime $\tilde{O}(\beta^2)$

Math Facts

- There are (asymptotically) $\frac{\beta}{\log \beta}$ primes in the first β integer
- A number is " β -smooth" if its prime factors are all $\leq \beta$. There are $\frac{N}{u^u}$ β -smooth $\#s \leq N$. For $u = \frac{\log N}{\log \beta}$

How? I'm glad you asked.

$$\beta = L_2(1/2, 1)$$

$$\log \beta = \sqrt{\log 2} \cdot \log \log 2$$

$$u = \frac{\log q}{\log \beta} = \frac{\log q}{\sqrt{\log 2} \cdot \log \log 2}$$

$$u^u = \exp(u \log u) = \exp\left(\frac{\log q}{\sqrt{\log 2} \cdot \log \log 2} \cdot \log\left(\frac{\log q}{\sqrt{\log 2} \cdot \log \log 2}\right)\right)$$

$$\leq \exp\left(\frac{\log q}{\sqrt{\log 2} \cdot \log \log 2} \cdot \left[\log \log 2 - \log \log \log 2\right]\right)$$

$$\leq \exp\left(\frac{1}{2} \sqrt{\log 2} \cdot \log \log 2\right)$$

$$\leq \exp\left(\sqrt{\log 2} \cdot \log \log 2\right) = \beta$$

notice the stack better analysis? Yes!

Step 2: Getting the $\{\log_g p_i\}$

Sample r s.t. g^r factors in β .

$$g^r = \prod p_i^{e_i}$$

$\Rightarrow r = \sum e_i \log_g p_i \leftarrow$ linear eqn in $\log_g p_i$

$\leftarrow \tilde{O}(\beta^2)$ time to find r .

Repeat $O(|\beta|) = \tilde{O}(\beta)$ times to get a solvable linear system

• solve for all $\log_g p_i$

$\rightarrow O(\beta^3)$ time

Total runtime: $\tilde{O}(\beta^3) = L_2(1/2, 3)$

Best known attack?

$$L_2(1/3, 2)$$

for $q \approx 2^{2048}$ $\approx 2^{122}$ \leftarrow close to 128

\Rightarrow source of thousand-bit moduli for \mathbb{Z}_p^*

RSA?

Similar attacks [Lenstra '87] [Lenstra '93]

Conclusion

We need better d -log groups!

Next time: elliptic curve groups!