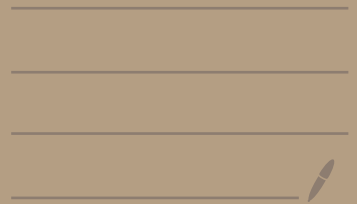
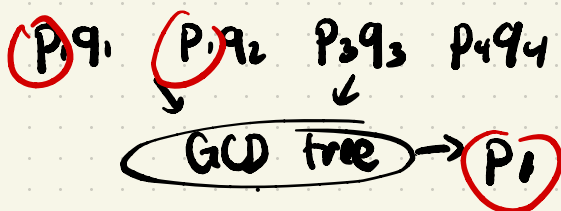


Cryptanalysis: Discrete-Log



Last Thursday: RSA cryptanalysis

↳ Mining p 's & q 's



bad
randomness

2. Coppersmith / Iafineon

"optimized" RSA keygen:

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

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

Today:

Direct attacks on d -log

1. Generic Group Attacks

a. "Baby-step giant-step"

b. "Pollard's rho"

c. Shoup's lower bound

2. \mathbb{Z}_q^* attack: index calculus.

Generic Group D-log:

- G a group: order q , generator g .
- given $h \in G^x$, $x \in \mathbb{Z}_q$, find x .
- Warmup: Brute force search:
 - $\approx 1/2$ expected time \rightarrow group ops!
 - $O(1)$ space

Visualisation:

$$g^0 \rightarrow g^1 \rightarrow g^2 \dots$$

\uparrow
uses baby steps,

but giant steps are

also cheap.

$$g^x = h^{\downarrow}$$

g^x computable w/ $\approx 2 \log_2 x$
group ops!

Baby Step, Giant Step [Shanks '71]

Idea: use giant steps to precompute

$$\sqrt{q} \text{ - spaced } \quad \text{sign - posts} \quad g^0 \longrightarrow g^{\sqrt{q}}$$

Map M :

$$g^{i\sqrt{q}} \mapsto i\sqrt{q}$$

$i \in \{0, 1, \dots, \sqrt{q}\}$

$$g^{i\sqrt{q}}$$

Offline phase (g):

build $M \leftarrow O(\sqrt{q} \log q)$ time & space

Online phase (h):

find $j \in \{0, 1, \dots, \sqrt{q}\} \leftarrow O(\sqrt{q} \log q)$ time

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

return $M[hg^j] - j$

BS - GAS in practice:

Say $q \approx 2^{80}$ (real life: $q \approx 2^{256}$)

time

$$\frac{3}{2} \sqrt{q} \log_2 \sqrt{q} \text{ g-ops}$$

curve 25519
-dalek:

↙ $\approx 45 \mu\text{s / op}$

$$\frac{3}{2} 2^{40} \cdot 40 \text{ g-ops}$$

$\Rightarrow 94 \text{ cpu years}$

\uparrow easily parallelized

space

$$\sqrt{q} \log_2 \sqrt{q} \text{ g-elements}$$

$$2^{40} \cdot 40 \cdot 32 \text{ bytes}$$

$$\Rightarrow 1.4 \text{ PB}$$

in a single table...

can we reduce space?

Pollard's Rho algorithm [75]

Idea: use a random walk!

$$\begin{aligned} u_0 &= g^{a_0} h^{b_0} \\ u_1 &= g^{a_1} h^{b_1} \dots \rightarrow u_i = g^{a_i} h^{b_i} \\ &\parallel \text{equal!} \\ u_j &= g^{a_j} h^{b_j} \end{aligned}$$

$$\Rightarrow g^{a_i} h^{b_i} = g^{a_j} h^{b_j}, \quad b_i \neq b_j$$

$$a_i + x b_i = a_j + x b_j \Rightarrow x = \frac{a_j - a_i}{b_i - b_j}$$

If transition function is pseudo-random, \sqrt{q} steps needed...

Need:

1. a pseudo-random transition function over (a, b) .

2. cycle-detection

Cycle-finding: Floyd's Tortoise + Hare

[Knuth '68]

An infinite sequence

x_1, x_2, x_3, \dots

which eventually cycles

→ two pointers, one at double speed

$$t_i = x_i$$

$$h_i = x_{2i}$$

→ distance between them: i

→ if cycle has length l ,

$t_j = h_j$ for first j divisible
by l after t_i enters
the cycle

Pseudo-random steps

- can't hash $H(g^a h^b) \rightarrow g'$
(wouldn't know exponents for g')
- simple alternative:

split G into $S_D \cup S_G \cup S_H$

$$\text{Step}(u = g^a h^b) = \begin{cases} g^{2a} h^{2b} & u \in S_D \\ g^{a+1} h^b & u \in S_G \\ g^a h^{b+1} & u \in S_H \end{cases}$$

effective when split is
unrelated to group structure

Analysis (Pollard rho)

$O(\sqrt{q})$ time

$O(1)$ space (two pointers!)

Generic Group d-log lower bound

◦ [Shoup '97] shows any group-generic d-log algorithm requires $\Omega(\sqrt{q})$ group operations.

→ BS-GS & Pollard's ρ are time optimal* * log-factors.

→ Pollard's ρ is space optimal

→ uses Generic Group Model

→ great proof!

Thm 1 of "Lower bounds for the Discrete Log Problem and other Problems"

⇒ Next: non-generic attacks.

Warm-up: d-log in \mathbb{Z}_p (aka $(\mathbb{Z}_p, +)$)

\mathbb{Z}_p is $\{0, 1, 2, \dots, p-1\}$
under addition mod p .

d-log problem in \mathbb{Z}_p .

given a generator (e.g. 2)

and a $h = \underbrace{2 + 2^2 + \dots + 2^x}_{x \text{ times}}$

find x .

Solution: division!

only poly log(p) time

d-log in $(\mathbb{Z}_p, +)$: totally broken

D-log in \mathbb{Z}_p^* : Index Calculus

\mathbb{Z}_p^* : $\{1, 2, 3, \dots, p-1\}$ under
 $\times \pmod p$.

$$|\mathbb{Z}_p^*| = p-1. \leftarrow \text{even.}$$

for today, let $\frac{p-1}{2}$ be
a prime q , and
 g be a generator of
an order q subgroup...

Example:

$$\mathbb{Z}_7^* = \{1, 2, 3, 4, 5, 6\}$$

$$p=7, \quad q=3 \quad (\text{prime!})$$

$g=2$ generates $\{2, 4, 1\}$
size q . \uparrow

Understanding our goal:
a sub-exponential (but not poly)
attack

• define $L_N(\alpha, c) =$

$$\exp\left(c \ln^\alpha N (\ln \ln N)^{1-\alpha}\right)$$

$$\Rightarrow L_N(0, c) = \exp(c \ln \ln N) = (\ln N)^c$$

poly in $\ln N$

$$\Rightarrow L_N(1, c) = \exp(c \ln N) = N^c$$

exp in $\ln N$

$$\Rightarrow L_N\left(\frac{1}{2}, c\right) = \exp\left(c \sqrt{\ln N} \cdot \ln \ln N\right)$$

in between

we'll build a $L_q\left(\frac{1}{2}, 3\right)$ attack...

High-Level Alg ← factorization basis

1. Let $B = \{2, 3, 5, \dots, p_t\}$
be the primes $\leq \beta$ ← set later

⇒ 120 factors in $\{2, 3, 5\}$

⇒ 140 does not

2. Compute $\log_g p_i$ for $i \in \{1, 2, \dots, t\}$

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

• use "random self-reducibility"

• sample $r \leftarrow \mathbb{Z}_q$ until hg^r
factors in B .

$$\rightarrow hg^r = \prod_i p_i^{e_i}$$

$$\Rightarrow \log_g hg^r = \sum_i e_i \log_g p_i$$

$$\log_g h = \sum_i e_i \log_g p_i - r$$

Q: Probability a random $u \in G$ factors in B ?

\Rightarrow called being " β -smooth"

fact: there are $\approx \frac{P}{u^u}$
 β -smooth numbers $\leq P$,

where $u = \frac{\ln P}{\ln \beta}$

$\Rightarrow \Pr[\beta\text{-smooth}] \approx \frac{1}{u^u}$

w/ $\beta = L_p(1/2, 1)$,

one can show $u^u \approx \beta$.

$\Rightarrow O(\beta)$ samples to find
a β -smooth #.

\Rightarrow cost of a β -smooth check?

$\frac{\beta}{\ln \beta} \cdot \text{polylog}(\beta) = \tilde{O}(\beta)$

$|B| \uparrow$ \uparrow division time

\Rightarrow step 3 takes $\tilde{O}(\beta^2)$ time.

Step 2? How to get $\log_g p_i$?

2a. Sample $r \in \mathbb{Z}_q$ s.t. g^r factors
in B . $\leftarrow \tilde{O}(\beta)$ time

$$\Rightarrow g^r = \prod_i p_i^{e_i}$$

$$\Rightarrow r = \sum_i e_i \log_g p_i$$

\leftarrow linear equation in $\{\log_g p_i\}$

2b. repeat $\approx t$ times, for $\approx t$
random linear relations

\Rightarrow fact? $\approx \frac{\beta}{\ln \beta}$ primes less than β .

\Rightarrow so $t = \tilde{O}(\beta)$, need $\tilde{O}(\beta)$ repetitions

2c. solve eqns, via gaussian
elimination

$\rightarrow \tilde{O}(\beta^3)$ time

total time: $\tilde{O}(\beta^3)$, which

is $L_q(1/2, 3)$

d-log in \mathbb{Z}_p^* in practice?

best algorithms are

$L_p(1/3, 2)$ (better than $L_p(1/2, 3)$!)

$\approx 2^{122}$ for $p \approx 2^{2048}$

↑ source of thousand-bit security requirements for \mathbb{Z}_p^* .

RSA is similar...

Recap

1. Pollard's ρ alg. [75] breaks generic d-log in $O(\sqrt{q})$ time and $O(1)$ space
2. [Shoup '97] shows this is optimal for generic d-log attacks
3. However \mathbb{Z}_p^* is vulnerable to sub-exponential attacks
(**RSA** is too! - [Lenstra '87])
[Lenstra '93]
→ need better groups
⇒ Next Lecture: elliptic curve groups!