# Cryptanalysis: Discrete-Log

# Last Thursday: RSA cryplanalysis

1. Mining p's & q's

$p_1q_1$ $p_1q_2$ $p_3q_3$ $p_4q_4$

$\xrightarrow{}$ GCD tree $\rightarrow p_1$

bad randomness

2. Coppersmith / Infineon
"optimized" RSA keygen:

$$p = k\cdot M + 65537^a \,\%\, M$$
$$q = \ell\cdot M + 65537^b \,\%\, M$$

# 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 $h \leftarrow g^x$, $x \xleftarrow{\$} \mathbb{Z}_q$, find $x$.
- Warmup: Brute force search:

  $q/2$ expected time

  group ops!

  $O(1)$ space

Visualisation!

$$g^0 \to g^1 \to g^2 \cdots$$

uses baby steps,

but giant steps are also cheap.

$$g^x = h$$

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

# Baby Step, Giant Step [Shanks '71]

Idea: use giant steps to precompute

$\sqrt{q}$-spaced sign-posts

$$g^0 \rightarrow g^{\sqrt{q}} \rightarrow g^{2\sqrt{q}} \quad \cdots \quad g^{i\sqrt{q}}$$

Map $M$:

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

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

Offline phase $(g)$:

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

Online phase $(h)$.

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

s.t. $h g^j \in M$.

return $M[hg^j] - j$

# BS-GS in practice:

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

time

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

curve 25519-dalek: $\approx 45\mu s$ / op

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

$\Rightarrow$ 94 cpu years

easily parallelized

space

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

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

$\Rightarrow$ 1.4 PB

in a <u>single</u> table...

can we reduce space?

# Pollard's Rho algorithm ['75]

Idea: use a random walk!

$u_0 = g^{a_0} h^{b_0} \rightarrow u_1 = g^{a_1} h^{b_1} \rightarrow \cdots \rightarrow u_i = g^{a_i} h^{b_i} \rightarrow \cdots \rightarrow u_j = g^{a_j} h^{b_j}$

// equal!

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

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

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

$x_1, x_2, x_3, \ldots$

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 $\ell$,

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

# Pseudo-random steps

- can't hash $H(g^a h^b) \to g'$
  (wouldn't know exponents for $g'$!)
- simple alternative:
  split $G$ into $S_D \cup S_G \cup S_H$

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

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 $\rho$ are time optimal*
    * log-factors...
  - → Pollard's $\rho$ 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 $\{0, 1, 2, \ldots, p-1\}$

under addition mod $p$.

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

given a generator (e.g. 2)

and a $h = \underbrace{2+2+\ldots+2}_{x \text{ times}}$,

find $x$.

Solution: division!

only polylog($p$) time

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

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

$\mathbb{Z}_p^*$ : $\{1, 2, 3, \ldots, p-1\}$ under

$\times$ mod $p$.

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

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

example:

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

$p=7$, $q=3$ (prime!)

$g=2$ generates $\{2, 4, 1\}$

size $q$.

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 $\ln N$

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

in between

we'll build a $L_q(1/2, 3)$ attack...

# High-Level Alg

factorization basis

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

$\Rightarrow 120$ factors in $\{2, 3, 5\}$
$\Rightarrow 140$ does not

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

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 $B$.

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

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

$$\boxed{\log_g h = \sum_i e_i \log_g p_i - r}$$

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

$\Rightarrow$ called being "$\beta$-smooth"

fact: there are $\approx P/u^u$ $\beta$-smooth numbers $\le P$, where $u = \dfrac{\ln p}{\ln \beta}$

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

w/ $\beta = L_p(1/2, 1)$, one can show $u^u \approx \beta$.

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

$\Rightarrow$ cost of a $\beta$-smooth check?

$$\underbrace{\frac{\beta}{\ln \beta}}_{|B|} \cdot \underbrace{\text{polylog}(\beta)}_{\text{division time}} = \tilde{O}(\beta)$$

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

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

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

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

$\Rightarrow r = \sum_i e_i \log_g p_i$ ↙ 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 $\beta$.

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

2c. Solve eqns, via guassian 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 $\rho$ 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: eliptic curve groups!