Cryptanalysis: Discrete-Log
Last Thursday: RSA cryptanalysis

1. Mining \( p's \) \& \( q's \):

\[
\begin{align*}
\{ P_1, P_2, P_3, P_4 \} & \quad \text{(GCD free)} \quad P_1
\end{align*}
\]

2. Coppersmith/Infineon

"Optimized" RSA keygen:

\[
\begin{align*}
p &= k \cdot M + 65537^a \quad \% M \\
q &= l \cdot M + 65537^b \quad \% M
\end{align*}
\]

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 $h = g^x, x \in \mathbb{Z}_q$, find $x$.
- **Warmup:** Brute force search: $2^{1/2}$ expected time $O(n)$ group ops!

**Visualisation:**

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

$\uparrow$

uses baby steps,
but giant steps are also cheap.

$g^x \text{ computable w/ } \leq 2\log_2 x$ group ops!
Baby Step, Giant Step [Shanks '71]

Idea: use giant steps to precompute $\sqrt{q}$-spaced sign-posts.

$$g \rightarrow g^{\sqrt{q}} \rightarrow g^{2\sqrt{q}}$$

Map $M$:

$$g_i^{\sqrt{q}} \rightarrow i^{\sqrt{q}}$$

$$i \in \{0, 1, \ldots, \sqrt{3}\}$$

Offline phase (g):
Build $M \leftarrow O(\sqrt{q} \log q)$ time & space.

Online phase (h):
Find $j \in \{0, 1, \ldots, \sqrt{3}\} \leftarrow O(\sqrt{q} \log q)$ time
s.t. $h g^i \in M$

return $M[h g^i] - j$
BS - GS in practice:
say $q \approx 2^{80}$ (real life: $q \approx 2^{256}$)

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

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

$\Rightarrow 94$ cpu years  
Easily parallelized

$$\sqrt{q} \log_{2}\sqrt{q} \text{ g-elements}$$

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

$\Rightarrow 1.4$ PB

in a single table...

can we reduce space?
Pollard's Rho algorithm [’75]

Idea: use a random walk!

\[ u_i = g^i h^j \]

\[ u_0 = g^0 h^0 \]

\[ u_i = g^i h^j \]

\[ u = g^a h^b \]

\[ g^a h^b = g^a h^b, \ b_i \neq b_j \]

\[ a_i + x b_i = a_j + x b_j \implies x = \frac{a_j - a_i}{b_i - b_j} \]

If transition function is pseudo-random, √q steps needed...

Need:
1. a pseudo-random transition function over \((a, b)\)
2. cycle detection
Cycle-finding: Floyd's Tortoise and Hare

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 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 \( SD \cup SG \cup SH \)

\[
\text{step(}u = g^a h^b = \bigcup \bigg\{ g^e h^b \ and \ od \bigg\} \bigg\}
\]

effective when split is unrelated to group structure

Analysis (Pollard rho)

\( \mathcal{O}(\sqrt{q}) \) time

\( \mathcal{O}(1) \) space (two pointers!)
Generic Group d-log lower bound

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

\[ \rightarrow \] BS-GS & Pollard's \( \rho \) are time optimal\(^*\)

\[ \rightarrow \] Pollard's \( \rho \) is space optimal

\[ \rightarrow \] uses \underline{Generic Group Model}

\[ \rightarrow \] great proof!

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

\[ \Rightarrow \] 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 \( h = 2 + 2 + \ldots + 2 \) (x times),

find \( x \).

Solution: division!

only \( \text{poly log} \ (p) \) time

d-log in \( \mathbb{Z}_p^* \): totally broken
D-log in ℤ\(_p^\times\): Index Calculus

ℤ\(_p^\times\) = \{1, 2, 3, ..., p-1\} under \(\times \mod p\).

\(|ℤ\(_p^\times\)| = p-1.

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

Example:

ℤ\(_7^\times\) = \{1, 2, 3, 4, 5, 6, 13\}

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

\(g=2\) generates \{2, 4, 13\} size \(q\).
Understanding our goal: a sub-exponential (but not poly) attack

• define

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

$$ \Rightarrow LN(0, c) = \exp \left( c \ln \ln N \right) = \left( \ln N \right)^{poly \ ln \ N} $$

$$ \Rightarrow LN(1, c) = \exp \left( c \ln N \right) = N^c $$

$$ \Rightarrow LN(\frac{1}{2}, c) = \exp \left( c \sqrt[\ln N]{\ln \ln N} \right)^{exp \ ln \ ln N} $$

in between.

we'll build a $L_q(\frac{1}{2}, 3)$ attack...
High-Level Alg

1. Let \( B = \{2, 3, 5, \ldots, p_3\} \) 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 \( i \in \{2, 3, \ldots, 13\} \)

3. Compute \( \log gh \) from \( \{\log g p_i\} \)
   - use "random self-reducibility"
   - sample \( r \leftarrow \mathbb{Z}_g \) until \( hgr \) factors in \( B \).

\( \Rightarrow hgr = \prod_i p_i e_i \)

\( \Rightarrow \log g h + r = \sum_i e_i \log g p_i \)

\( \log g h = \sum_i e_i \log g p_i - r \)
Q: Probability a random $u \in G$ factors in $B$?

- called being "$B$-smooth"

**Fact:** there are $\approx \frac{\ln^u}{u}$ $B$-smooth numbers $\leq \beta$,

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

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

If $\beta = L_p(\frac{1}{2}, 1)$,

one can show $u^u \propto \beta$

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

- cost of a $B$-smooth check?

\[ \frac{\beta}{\ln \beta} \cdot \text{poly log } (\beta) = \tilde{O}(\beta) \]

- division time

- $\Rightarrow$ step 3 takes $\tilde{O}(\beta^2)$ time.
Step 2: How to get $\log g \pi$?

2a. Sample $r \in \mathbb{Z}$ s.t. $g^r$ factors in $B$. \(\sim \Theta(\beta^3)\) time

\[ g^r = \prod_i p_i^{e_i} \]

\[ r = \sum_i e_i \log g \pi \]

2b. Repeat $t \pi \beta$ times, for $\pi \beta$ random linear relations

\[ \text{fact?} \leq \frac{\beta}{\ln \beta} \text{ primes less than } \beta \]

\[ \Rightarrow \text{so } t = \Theta(\beta), \text{ need } \Theta(\beta) \text{ repetitions} \]

2c. Solve eqns, via guassian elimination

\( \sim \Theta(\beta^3) \) time

Total time: \( \Theta(\beta^3) \), which is \( \mathcal{L}_q \left( \frac{1}{2}, 3 \right) \)
d - log in \( \mathbb{Z}_p^* \) in practice?

best algorithms are

\[ L_p\left(\frac{1}{3}, 2\right) \text{ (better than } L_p\left(\frac{1}{2}, 3\right)! \) \\
\sim 2^{122} \text{ for } p \times 2^{2048} \]

\( \uparrow \)

\( \uparrow \) 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, $\mathbb{Z}_p^\times$ is vulnerable to sub-exponential attacks
   (RSA is too! - [Lenstra ’87])
   $\rightarrow$ need better groups

$\Rightarrow$ Next Lecture: elliptic curve groups!