# The Pohlig-Hellman algorithm

- Given some non-zero $b \in F_q$ and generator $g$, how can we find $x$ such that $b = g^x$?
- Write $q - 1 = p_1^{k_1} \dots p_m^{k_m}$ and let $\ell^k$ be $p_i^{k_i}$ for some $i$.
- Now we can generate all powers

$$g^{m\frac{q-1}{\ell}} \text{ for } m = 0 \dots \ell - 1.$$

- Computed the sequence $x_0, x_1, \dots, x_{i-1}$ and a sequence of elements of $F_q$, $b_0 = b, b_1, \dots, b_{i-1}$ so that for $0 < i < k$,

$$b_i = b_{i-1} g^{-x_{i-1}\ell^{i-1}} \text{ and } b_i^{\frac{q-1}{\ell^{i+1}}} = g^{x_i \frac{q-1}{\ell}}.$$

- Determine $x_i$ by comparing to our list of powers of $g$ and finally determine $x$ by the Chinese Remainder Theorem.

# Analysis of Pohlig-Hellman

- This attack is reasonably effective assuming that you can factor $q-1$.
- It runs in time proportional to the size of the largest prime divisor of $q-1$.
- As with the Pollard $p-1$ algorithm, the take-away here is to make sure that $q-1$ has some large prime factor.
- For instance, a Mersennes prime is a prime of the form $2^n-1$. Since there are fields of size $2^n$ for all $n$, any $n$ such that $2^n-1$ is prime would be a good choice. There are not known to be infinitely many such primes but there are such with millions of digits.

# Baby step - giant step algorithm

- Again we try to find $x$ from $b$ given a generator $g$ and $b = g^x$. Let $N = [\sqrt{q-1}] + 1$.
- We make two lists:

$$\begin{array}{lr} \underline{\text{Baby step}} & \underline{\text{Giant step}} \\ g^0 & b \\ g^1 & bg^{-N} \\ g^2 & bg^{-2N} \\ \vdots & \vdots \\ g^{N-1} & bg^{-(N-1)N} \end{array}$$

- We look for a match between the two lists and if we find one, say

$$g^i = bg^{-kN} \text{ then } b = g^{i+kN}$$

and we have found $x$.

# You can always find $x$

- Note $0 \leq x < q - 1 \leq N^2$ so $x = x_0 + x_1 N$ for some $x_0, x_1 \leq N$.
- This means
$$b = g^x = g^{x_0} \cdot g^{x_1 N}$$
and so
$$g^{x_0} = b g^{-x_1 N}.$$
- This algorithm takes on the order of $\sqrt{q}$ many steps.

# Index calculus attack

- Here is another attack on discrete logs. It is similar to the quadratic sieve method and I will only describe it for fields $F_p$ where $p$ is a prime. It can be done in general for any finite field with a little more effort.
- Now everything is a number: $g$ is a generator of $F_p$ and $b$ is a non-zero element of $F_p$ and we want to find $x$ such that $b = g^x$.
- We fix some primes $p_1, p_2, \ldots, p_m$ and suppose that for some $k$

  $$g^k \equiv p_1^{\alpha_1} p_2^{\alpha_2} \ldots p_m^{\alpha_m} \bmod p.$$

  Then

  $$k = \alpha_1 L_g(p_1) + \alpha_2 L_g(p_2) + \ldots + \alpha_m L_g(p_m) \bmod p - 1.$$

- If we do this for sufficiently many $k$ then we will learn the value of $L_g(p_i)$ for all $i$ just by solving these linear equations.

# Index calculus attack

- Now if we can find some $r$ such that

$$bg^r \equiv p_1^{\beta_1} p_2^{\beta_2} \dots p_m^{\beta_m} \bmod p$$

then

$$L_g(b) = -r + \beta_1 L_g(p_1) + \dots + \beta_m L_g(p_m) \bmod p - 1.$$

- How do we find $r$? Pick $r$'s randomly between 0 and $p$. By the birthday problem argument, if this is going to work it will work quickly. The issue is choosing enough primes $p_i$ so that one can generate enough $g^k$'s.