Breaking Diffie-Hellman/ElGamal

The security of Diffie-Hellman and ElGamal are based on the difficulty of solving the discrete log problem

That is if we had a way of solving

$$a^x \equiv b \ (mod \ n)$$

then these methods of key exchange are vulnerable.

Given a primitive root a for an integer n there are methods for solving the equation

$$a^x \equiv b \ (mod \ n)$$

but these algorithms do not run much faster than $O(\sqrt{n})$ and for sufficiently large n the difference between the speed of key exchange and breaking the key exchange is large.

Baby step/Giant step method

Goal: Solve $a^x \equiv b \ (mod \ n)$.

Idea: Find $a^i \equiv ba^{-j} \ (mod \ n)$ by searching through a small enough space of possible i and j.

Fix $m = \left\lceil \sqrt{(\phi(n))} \right\rceil$ then find $c \equiv a^{-m} \ (mod \ n)$.

Next calculate a table of $a^i \ (mod \ n)$ for $0 \leq i < m$ and then calculate $bc^j \ (mod \ n)$ for $0 \leq j < m$ until you find one of these values in the table.

Solution: When we find $a^i \equiv bc^j \ (mod \ n)$ then we have $a^{i+mj} \equiv a^i c^{-j} \equiv b \ (mod \ n)$.

Example: $p = 53$ and $a = 3$. We wish to solve

$$3^x = 41 \ (mod 53).$$

- $m = \lceil \sqrt{\phi(53)} \rceil = 8$ and $3^{-8} \equiv 24 \ (mod\ 53)$.
- Now $41 \cdot 24^i \ (mod\ 53)$.

i	$3^i \ (mod\ 53)$
0	1
1	3
2	9
3	27
4	28
5	31
6	40
7	14

i	$41 \cdot 24^i \ (mod\ 53)$
0	41
1	30
2	31
3	2
4	48
5	39
6	35
7	45

- Conclusion: $3^{2 \cdot 8+5} \equiv 3^{21} \equiv 41 \ (mod\ 53)$

There are several improvements to this algorithm but which do not change the speed of algorithm wildly (i.e. it is still much harder to take a discrete log than it is to find $a^b (mod m)$).

- The Pohlig-Hellman algorithm (section 9.2) reduces the discrete logarithm problem to order $O(\sqrt{p})$ where p is the largest prime which divides $\phi(n)$. This implies that we should insure that when we choose the modulus p in the Diffie-Hellman/ElGamal key exchange, we should ensure that $\phi(p) = p - 1$ has large prime factors.

- Some other improvements to this method reduce the memory required to store values to compare and are more suitable for parallel implementation (say over the internet).

Security in Modern cryptography relies on trapdoor functions...

Multiply large primes together (easy)	$\leftrightarrow$	Factor large integers into primes (hard)
Compute $y \equiv a^b \ (mod \ \ m)$ (easy)	$\leftrightarrow$	Find x such that $a^x \equiv b \ (mod \ \ m)$ (hard)

Are there others????