Cryptography I, homework sheet 10

Due: 16 December 2011, 10:45

Both exercises can be done with the help of a computer but you should submit your programs as part of the homework solution. The program can be based on any computer algebra system, in particular for computing in $\mathbb{F}_{1013}^*$. Make sure that your programs compile and run correctly; my students will not debug your programs. The programs should be humanly readable.

1. $3 \in \mathbb{F}_{1013}^*$ generates a group of order $1012 = 4 \cdot 11 \cdot 23$. Solve the discrete logarithm problem $g = 3, h = 321$ by using the Pohlig-Hellman attack, i.e. find an integer $0 < k < 1012$ such that $h = g^k$ by computing first k modulo $2, 4, 11$, and 23 and then computing k using the Chinese Remainder Theorem.

2. $3 \in \mathbb{F}_{1013}^*$ generates a group of order 1012, so it generated the whole multiplicative group of the finite field. Solve the discrete logarithm problem $g = 3, h = 224$ using the Baby-Step Giant-Step algorithm (see below).

The *Pohlig-Hellman attack* attack works in any group and is a way to reduce the reduce the hardness of the DLP to the hardness of the DLP in subgroups of prime order. In particular you'll see in the exercise that it works against the DLP in $\mathbb{F}_{1013}^*$ by solving DLPs in groups of size 2, 11, and 23. Here is the general description:
This attack is called the *Pohlig-Hellman attack* and breaks the DLP by breaking it in subgroups of prime order. So the DLP in the full group is no harder than the DLP in the biggest prime-order subgroup. The two numerical examples were using a table to solve the smaller DLPs; usually the factors are too large for that and BSGS or Pollard rho are used as subroutines.
Let G be a cyclic group generated by g and let the challenge be to find $\log_g h = k$. Let the group order n factor as $n = \prod_{i=1}^{r} p_i^{e_i}$ where $p_i \neq p_j$ for $i \neq j$. Then k can be computed from the information

$$\begin{aligned} k &\equiv k_1 \bmod p_1^{e_1} \\ k &\equiv k_2 \bmod p_2^{e_2} \\ k &\equiv k_3 \bmod p_3^{e_3} \\ &\vdots \\ k &\equiv k_r \bmod p_r^{e_r} \end{aligned}$$

by using the Chinese remainder theorem. This is because the $p_i^{e_i}$ are coprime and their product is n. So, if one can find the DL modulo all $p_i^{e_i}$ one can compute the entire DL.
Put $n_i = n/p_i^{e_i}$. Since g has order n the element $g_i = g^{n_i}$ has order $p_i^{e_i}$. The element $h_i = h^{n_i}$ is in the subgroup generated by g_i and it holds that $h_i = g_i^{k_i}$, where $k_i \equiv k \bmod p_i^{e_i}$.
E.g. $\mathbb{F}_{16}^* = \langle g \rangle$ has 15 elements, so one can first solve the DLP $h = g^k$ modulo 3 and then modulo 5. For such small numbers one can simply compute h^5 and compare it to $1, g^5$, and g^{10} to find whether k is equivalent to $0, 1$, or 2 modulo 3. Then one compares h^3 to $1, g^3, g^6, g^9$, and g^{12} to see whether k is congruent to $0, 1, 2, 3$, or 4 modulo 5.

The same approach works also for $\mathbb{F}_{17}^*$ which has $16 = 2^4$ elements – but here one can do much better! Write $k = k_0 + k_1 2 + k_2 2^2 + k_3 2^3$. Then h^8 is either equal to 1 or to $-1 = g^8$ depending on whether k_0 is 0 or 1. Once that result is known we can compare $(h/g^{k_0})^4$ with 1 and -1 to find k_1 etc. So we can solve a much smaller DLP. Instead of going for k modulo $p_i^{e_i}$ at once we can first obtain k modulo p_i, then modulo p_i^2, then modulo p_i^3, etc. till $p_i^{e_i}$ by each time solving a DLP in a group of size p_i.
Numerical examples:

$\mathbb{F}_{11}^* = \langle 2 \rangle$, find k so that $3 = 2^k$. So $g = 2$ and $h = 3$. Compute $n_1 = 10/2 = 5$, $g^{n_1} = 2^5 = -1$, and $h^{n_1} = 3^5 = 1$ to see that $k \equiv 0 \bmod 2$. Then compute $n_2 = 10/5 = 2$, $g^{n_1} = 2^2 = 4, g^{2n_1} = 2^4 = 5, g^{3n_1} = 2^6 = 9$, and $g^{4n_1} = 2^8 = 3$ and compare that to $h^{n_2} = 3^2 = 9$ to see that $k \equiv 3 \bmod 5$. These two congruences imply that $k = 8$ and indeed $g^8 = h$.
$\mathbb{F}_{17}^* = \langle 3 \rangle$, find k so that $7 = 3^k$. So $g = 3$ and $h = 7$. In this example we will obtain k one bit at a time. First compare $h^8 = 7^8 = -1$ to 1 and -1 to see that $k \equiv 1 \bmod 2$. Then compute $h/g = 8$ and then $(h/g)^4 = -1$, so also the next bit is 1 and we see $k \equiv 3 \bmod 4$. Then compute $h/g^3 = 16$ and then $(h/g^3)^2 = 1$ to see that the next bit is 0, so $k \equiv 3 \bmod 8$. Finally, since $h/g^3 = 16 = -1$ we see that the highest bit is 1, so $k \equiv 11 \bmod 16$ and indeed $3^{11} = 7$. This solved the DLP in $\mathbb{F}_{17}^*$ with just 4 very easy computations and comparisons. So computing DLs in fields $\mathbb{F}_p$ with $p = 2^r + 1$ is easy.

The *Baby-Step Giant-Step (BSGS) method* works in any cyclic group, so it can be used as a subroutine to the Pohlig-Hellman attack. Let ℓ be the group order and put $m = \lfloor \sqrt{\ell} \rfloor$. Then the discrete logarithm k can be written as $k = k_0 + k_1 m$ with $k_0 \in [0, m-1]$. The BSGS algorithm computes all powers g^i for integers $i \in [0, m-1]$ and then iteratively computes $h/(g^{jm})$ for j and checks whether this value is among the initially computes powers of g. The small powers of g are the baby steps, the powers $h/(g^{jm})$ are the giant steps. There must exist a match because $h/(g^{k_1 m}) = g^{k_0 + k_1 m - k_1 m} = g_0^k$ is among the precomputed values and will be found for $j = k_1 \leq \lceil \sqrt{\ell} \rceil$.
To make your computations more efficient you should sort the results from the baby steps (but remember which i belongs to which value) and compute the $h/(g^{jm})$ by first computing $d = g^{-m}$ (using 1 multiplication and one inversion, starting from the last of the baby steps) and then checking $h, h' = h \cdot d, h' = h' \cdot d, \ldots$ in succession. In summary the attack takes at most $2m$ multiplications and 1 inversion.