Cryptography

PROF. DR. JOACHIM VON ZUR GATHEN, KONSTANTIN ZIEGLER

7. Assignment: Index calculus and hash functions

(Due: Wednesday, 17 December 2008, 13^{40}, b-it bitmax)

Exercise 7.1 (smooth numbers and index calculus). (12 points)

In the first part of this exercise, consider the factor base $\mathcal{B} = \{2, 3, 5, 7, 11\}$ consisting of the first five prime numbers.

We want to get a feeling for the probability that a randomly chosen number in the range from 1 to 1000 factors over $\mathcal{B}$, i.e. is B-*smooth*.

(i) In a loop, draw random integers between 1 and 1000. Test whether they factor over $\mathcal{B}$ (note that no complete factorization is required for this). Repeat until you have found 20 B-smooth numbers. The fraction $20/i$, where i is the total number of performed iterations, is an estimate for the fraction of B-smooth numbers among the integers between 1 and 1000. What is yours? [3]

In the second part, we want to see the index calculus in action. We are interested in the multiplicative group $G = \mathbb{Z}_p^\times$ with $p = 227$ and generator $g = 2$. We choose as factor base $\mathcal{B} = \{2, 3, 5, 7, 11\}$ with all primes up to the bound $B = 11$.

In the preprocessing step we compute the discrete logarithms of all elements in the factor base $\mathcal{B}$.

(ii) Instead of randomly choosing exponents e and testing, whether g^e rem p factors over $\mathcal{B}$, we have already prepared a list with suitable exponents for you. Let e take values from $\{40, 59, 66\}$, give the factorization of g^e rem p over $\mathcal{B}$ and the corresponding linear congruence modulo $(p-1)$ involving the discrete logarithms of the elements in $\mathcal{B}$. [3]

(iii) The discrete logarithm of the generator $g = 2$ is obviously 1, but even with this information, the three linear relations from (ii) are not enough to determine the remaining four unknown discrete logarithms. Find one additional linear congruence from an exponent $e > 10$ yourself. [2]

(iv) Assuming that your additional congruence is linearly independent from the three previous ones, solve the system of congruences for the discrete logarithms of the base elements. (If you do this by hand, note that division by 2 is impossible modulo $(p-1)$. If you use a computer algebra system, note that those are aware of this problem and have special commands to solve systems of congruences with a given modul, e.g. `msolve` in MAPLE and `solve_mod` in SAGE.) [2]

Once we have found the discrete logarithms for the elements in the factor base, we can finally compute the discrete logarithm of any element x in the group with the following method:

- Choose random exponents e until $xg^e \mod p$ factors over $\mathcal{B}$, say $xg^e \equiv p_1^{\beta_1} p_2^{\beta_2} \cdots p_h^{\beta_h} \mod p$.
- The corresponding linear relation reads

$$\operatorname{dlog}_g x + e = \beta_1 \operatorname{dlog}_g p_1 + \beta_2 \operatorname{dlog}_g p_2 + \cdots + \beta_h \operatorname{dlog}_g p_h \mod (p-1)$$

- Since all the $\operatorname{dlog}_g p_i$ have already been determined in the preprocessing step, you can solve this equation modulo $(p-1)$ for $\operatorname{dlog}_g x$.

(v) Apply this procedure to compute $\operatorname{dlog}_2 224$ in $\mathbb{Z}_{227}^\times$. [2]

Exercise 7.2 (a discrete log hash function). (5 points)

A prime number q so that $p = 2q + 1$ is also prime, is called a *Sophie Germain prime*. We choose $q = 7541$ and $p = 2 \cdot 7541 + 1$ both prime and want to define a hash function on the set $\mathbb{Z}_q \times \mathbb{Z}_q$.

(i) Let $\alpha = 604$ and $\beta = 3791$. Prove that $\operatorname{ord}(\alpha) = \operatorname{ord}(\beta) = q$. [2]

The elements α and β actually generate the same subgroup of $\mathbb{Z}_p^\times$, i.e. $\langle\alpha\rangle = \langle\beta\rangle$. Call this subgroup G.

(ii) Now, we can define a hash function [1]

$$h : \mathbb{Z}_q \times \mathbb{Z}_q \to G, (x_1, x_2) \mapsto \alpha^{x_1} \beta^{x_2}.$$

Compute $h(7431, 5564)$ and $h(1459, 954)$ and compare them.

(iii) In (ii) you found a collision for the hash function h. This enables you to compute the discrete logarithm $\operatorname{dlog}_\alpha \beta$. Do it. [2]