Explanation of the Python Code

Tanouir AMARI

November 29, 2024

1 Introduction

This document explains the Python code used for discrete logarithm computations and B-smoothness checking. The code involves multiple steps: generating a factor base, checking for B-smoothness, finding relations, and solving a system of linear equations. The goal is to compute the discrete logarithm and perform related cryptographic operations.

2 1. Importing Libraries

The code begins by importing the necessary libraries:

- `math`: This library is used for basic mathematical operations.
- `sympy`: This is a powerful symbolic mathematics library that provides functions for primality testing, factorization, and modular arithmetic.
- `random`: This library is used for generating random numbers, especially for selecting random exponents.

3 2. Parameter Definitions

The following parameters are defined at the start of the program:

- `p`: This is the modulus used in calculations, which is a large prime number.
- `alpha`: A generator of the subgroup in the modular arithmetic. It is an element of order `p` used for exponentiation.

- `order`: The order of the subgroup generated by `alpha`, i.e., the number of distinct powers of `alpha` modulo `p`.
- `factor_base_bound`: The upper bound for generating the factor base, which is a list of primes used for factorizing numbers.

4 3. Function `generate_factor_base`

The `generate_factor_base` function generates a list of prime numbers up to the given bound. These primes will be used as the "factor base" to test whether numbers are B-smooth (i.e., can be factored into primes from the factor base).

Explanation: This function iterates through all integers up to `bound` and uses the `isprime` function from `sympy` to check for primality. All prime numbers found are added to the factor base.

5 4. Function `is_b_smooth`

The `is_b_smooth` function checks whether a number can be entirely factored into the primes in the factor base. A number is B-smooth if it can be expressed as a product of primes from the factor base.

Explanation: For each prime in the factor base, the function divides the input value by the prime as long as it is divisible. It counts the number of times each prime divides the value and stores this in a dictionary. If, after factoring out all the primes from the factor base, the value is reduced to 1, the number is considered B-smooth, and the function returns `True` along with the factorization. If the remaining value is not 1, the function returns `False`.

6 5. Function `generate_b_smooth_beta`

The `generate_b_smooth_beta` function attempts to generate a B-smooth number β by selecting random exponents k and computing $\alpha^k \mod p$. The function checks whether this resulting number is B-smooth.

Explanation: The function repeatedly selects a random exponent k between 1 and `order - 1`, computes $\beta = \alpha^k \mod p$, and checks if β is B-smooth using the `is_b_smooth` function. If a B-smooth number is found, the function returns β and its factorization. If no B-smooth number is found after a set number of attempts, it returns `None`.

7 6. Function `find_relations`

The `find_relations` function generates relations by computing powers of α modulo p and checking if these powers are B-smooth.

Explanation: The function iterates over all exponents k from 1 to `max_exponent`. For each exponent, it computes $\alpha^k \mod p$, then checks if the result is B-smooth using the `is_b_smooth` function. If the result is B-smooth, the factorization of that power is added to the list of relations, and the exponent k is stored. The function continues this process for all exponents up to `max_exponent`.

8 7. Function `solve_linear_system`

The `solve_linear_system` function solves a system of linear equations that arises from the relations found in the previous step.

Explanation: This function builds a matrix from the relations found and the corresponding exponents. The matrix represents a system of linear equations where the unknowns are the discrete logarithms of the prime factors of α. It uses modular arithmetic to solve this system and compute the discrete logarithms.

9 8. Function `calculate_discrete_log`

The `calculate_discrete_log` function computes the discrete logarithm of a number β using its prime factorization and the logarithms of the primes in the base.

Explanation: This function calculates the discrete logarithm of β by using the following relation:

$$\log_\beta = \sum(\text{exponent of prime} \times \log(\text{prime}))$$

For each prime factor of β, the function multiplies the exponent by the logarithm of the prime (from the base) and sums the results. If a logarithm for a prime factor is not available, it raises an error.