

MA080G Cryptography Summary Block 3

Viktor Rosvall

May 13, 2019

Discrete Logarithm problem [1]

The discrete logarithm problem is potential solution to the problem of finding the private exponent d , such that $x \equiv y^d \pmod{n}$ in the RSA cryptosystem.

Definition: given x, y and a prime p such that:

$$y \equiv x^e \pmod{p}$$

find e .

This problem however is believed to be as hard as factorization and not yet proven to be NP-complete. The order of x should be as large as possible to avoid it being broken by an exhaustive search. So x should be chosen as a primitive root mod p , which is an element of order $\lambda(p) = p - 1$

Knapsack problem [2]

Let's say we have a *knapsack* with a volume of b units, and a list of items (a_1, a_2, \dots, a_k) . We want to know if we can fill the knapsack with *some* of the items.

We want to find a tuple e of length k , where $e \in \{0, 1\}$, and

$$\sum_{i=1}^k e_i a_i = b$$

where b is the ciphertext.

The knapsack problem is NP since we can easily check if a solution is correct. Finding this solution is hard. We have in the worst-case 2^k possible e tuples to check.

In the case of a *super-increasing* data series a , the knapsack-problem degrades into an *easy* problem, so it's not always NP-complete. But it's considered *hard* since we classify problems of it's worst-case behavior.

A **super-increasing** sequence is defined as a series of positives integers where each term is greater than the sum of it's predecessors,

$$\sum_{j=1}^{i-1} a_j < a_i$$

For example 1, 2, 4, 8 is a super-increasing sequence.

Merkle-Hellman knapsack cipher [3]

To encrypt using a Merkle-Hellman knapsack cipher we need to create a super-increasing sequence (a_1, a_2, \dots, a_k) which will be our *private key* component. To create a public key component we need to *disguise* the sequence so it can't be broken using the greedy-algorithm.

To do this we need to choose an integer n greater than the sum of the a_i sequence and an integer u such that $\gcd(n, u) = 1$, then compute:

$$a_i^* = ua_i \text{ MOD } n$$

for each a , creating a new sequence $(a_1^*, a_2^*, \dots, a_k^*)$ which will be our *public key*.

ElGamal cryptosystem

Sophie-Germain primes [4]

Using the properties of some special primes, we can easily find a primitive root. A prime number pair (q, p) is called a *Sophie-Germain* pair if:

$$p = 2q + 1$$

Proposition: let (q, p) be a *Sophie-Germain* pair. Suppose that $1 < x < p - 2$. Then x is a primitive root mod p if and only if:

$$x^q \equiv -1 \pmod{p}$$

References

- [1] P. J. Cameron, *Notes on cryptography*.
<http://www.maths.qmul.ac.uk/~pjc/notes/crypt.pdf> Page 78-80
- [2] P. J. Cameron, *Notes on cryptography*.
<http://www.maths.qmul.ac.uk/~pjc/notes/crypt.pdf> Page 78-80
- [3] P. J. Cameron, *Notes on cryptography*.
<http://www.maths.qmul.ac.uk/~pjc/notes/crypt.pdf> Page 80-82
- [4] P. J. Cameron, *Notes on cryptography*.
<http://www.maths.qmul.ac.uk/~pjc/notes/crypt.pdf> Page 108-109