Public Key Cryptography – Lab4

The Classical Trial-Division Algorithm

Input: $n \in \mathbb{N}, n \geq 3$ odd, composite
Output: a non-trivial factor d of n

```
Algorithm classic(n)
    for i ∈ {2, …, ⌊√n⌋} do
        if n % i = 0 then
            return i
        endif
    done
    return n
```

Input: $n \in \mathbb{N}, n \geq 3$ odd, composite
Output: the factors of n

```
Algorithm factorize(n)
    let F = ∅ be the list of factors
    while n ≠ 1 do
        let d := classic(n)
        F := F ∪ {d}
        n := ⌊n/d⌋
    done
```

The Pollard p-1 algorithm

Input: $n \in \mathbb{N}, n \geq 3$ odd, composite, B – a bound
Output: a non-trivial factor d of n

```
Algorithm classic(n)
    let k := lcm{1, …, B}
    let a be chosen randomly from {2, …, n − 2}
    a := a^k mod n
    let d := gcd(a − 1, n)
    if d = 1 or d = n then
        return n
    endif
    return "failure"
```

The Running-Time Analysis

Input n	Pollard $p-1$	Classical
19 122 127 019	**0.511000**	**0.128000**
673 542 175 381	**0.531000**	**0.124000**
3 172 441 039	**0.511000**	**0.100000**
2 534 774 595	**0.530000**	**0.101000**
21 518 399 815 424 693	**0.545000**	**0.218000**
988 053 892 081	**0.549000***	**0.117000**
17 450 859 840 493	**0.543000***	**0.127000**
338 694 389 826 206 941	**0.529000**	**0.168000**
20 660 357 779 398 623 401	**0.562000**	**1.355000**
2 086 696 135 791 260 963 501	**0.552000**	**1.136300**

The running time of *Pollard's* $p-1$ *algorithm* seems to be higher in comparison to the running time for the *classical* algorithm for "small" (10-12 digits) numbers. It is probably caused by the difference in implementations of the two algorithms, for *pollard*, *gmplib* is used, and for *classical*, C math library is used.
For larger numbers, the running time of the *classical algorithm* is increasing, while the running time for *pollard's* $p-1$ *algorithm* stays at the same level.

Vlad Cîncean / 933
cvie1883@scs.ubbcluj.ro