# Pollard's $\rho$ Algorithm

**Lab Assignment 3 - Public Key Cryptography, UBB-CS Year 3**

**Popa Alex Ovidiu, group 936**

Problem Statement: Pollard's $\rho$ algorithm. The implicit function will be f(x) = x^2 + 1, but it will also allow the use of a function f given by the user.

Function to compute the gcd of two integers, to be used in Pollard's algorithm.

```
<<gcd>>=
def gcd(x, y):
    while y:
        r = x % y
        x = y
        y = r
    return x

@
```

Function which evaluates a function passed as a string, like "x^2+1". It is evaluated in the value x.

```
<<funcall>>=
def funcall(f, val):
    coeff = f.replace('[','').replace(']','').split(',')
    s = 0
    for i in range(len(coeff)):
        s+=int(coeff[i])*(val**i)
    return s

@
```

Pollard's $\rho$ Algorithm implementation, following the algorithm from the lecture.

```
<<PollardImpl>>=
<<gcd>>
<<funcall>>
def pollard(n, x0, f):
    x = [x0]
```

```
    j = 1
    while True:
        xj = funcall(f, x[-1]) % n
        x.append(xj)
        xj = funcall(f, x[-1]) % n
        x.append(xj)
        d = gcd(abs(x[2 * j] - x[j]), n)
        if 1 < d < n:
            return d
        elif d == n:
            return None
        j += 1
```

@

Pollard Function runner, with x0 going from 2 until a solution is found.

```
<<PollardRunner>>=
<<PollardImpl>>
def pollardRunner(n, f):
    x0 = 2
    while True:
        result = pollard(n, x0, f)
        if result is not None:
            return result
        x0 += 1
```

@

Main function, cmd line arguments are interpreted and the function is executed.

```
<<*>>=
<<PollardRunner>>
import sys

def main():
    n = 7031
    f = '[1,0,2]'
    params = sys.argv[1:]
    if len(params) == 4 and params[0] == "-n" and params[2] == "-f":
        n = int(params[1])
        f = params[3]
    elif len(params) == 2 and params[0] == "-n":
        n = int(params[1])
    print("Running Pollard with n={} and f={}".format(n, f))
    print(pollardRunner(n, f))
```

```
main()
```

```
@
```

Runner command example:

```
notangle pollard.md >pollard.py && python pollard.py -n 7031 -f [2,0,2]
```