Kryptoanaliza w .rb

# Na poczatku był cyphertext...

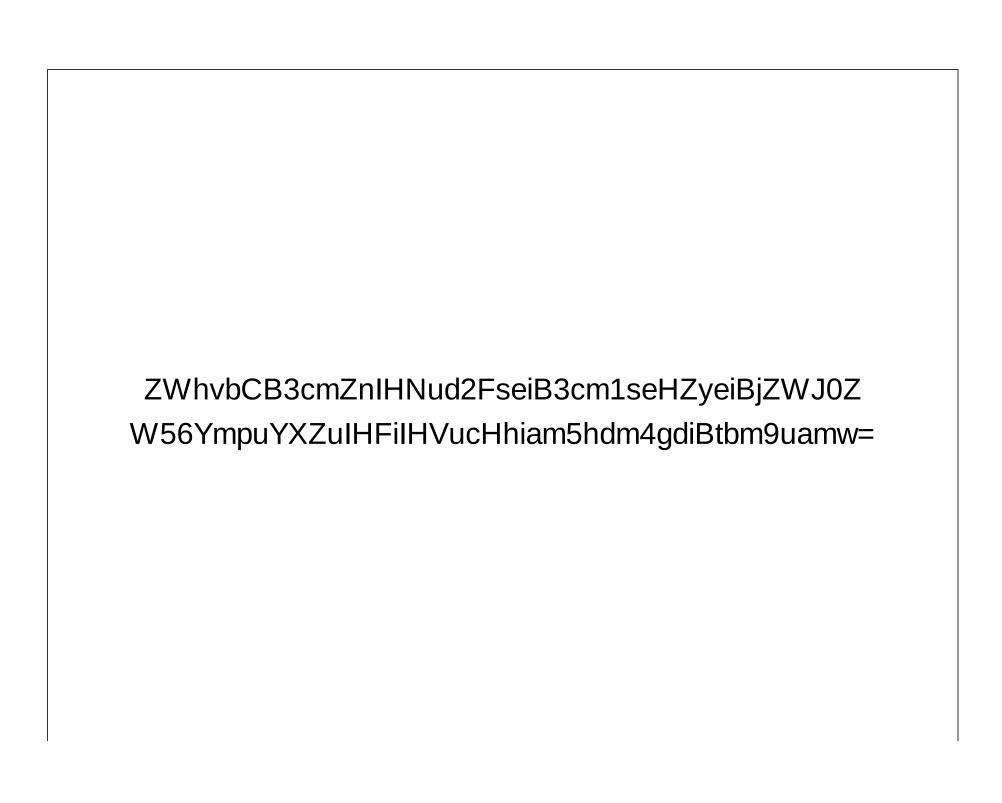


# Podobny do base64...

```
[plain] => [base64]
    'r' => 'cg=='
    'ru' => 'cnU='
    'rub' => 'cnVi'
```

```
b64_regexp = /^([A-Za-z0-9=]){4,}$/
if b64_regexp =~ "cnVieSBqZXN0IGZham55IQ=="
    p "Works!"
end
# => "Works!"
```

```
require 'base64'
Base64.decode64 "cnVieSBqZXN0IGZham55IQ=="
# => "ruby jest fajny!"
```



Wyglada jak szyfr podstawieniowy (jak cezara) Analiza jezyka Czyli częstość występowania liter w słowniku

```
def statistic(file)

result = {}
result.default = 0

file = File.new(file, "r")
file.each_line do |line|
  line.chomp!
  line.each_char do |chr|
   result[chr] += 1
  end
end
```

#### Statytyka języka polskiego

### Kolejność liter w języku polskim

```
["a", "i", "o", "n", "e", "z", "w", "r", "c", "y", "s", "k", "p", "m", "t", "l", "ł", "u", "d", "b", "j", "g", "ą", "ś", "h", "ę", "ż", "f", "ó", "ń", "ź", "ć", "c", "v", "x", "q"]
```

#### Statystyka cypher'a

{"e"=>3, "h"=>1, "o"=>2, "l"=>3, " "=>8, "w"=>3, "r"=>3, "f"=>1, "g"=>1, "s"=>1, "n"=>9, "a"=>3, "z"=>3, "m"=>2, "x"=>2, "v"=>4, "c"=>1, "b"=>4, "t"=>1, "j"=>3, "q"=>1, "u"=>1, "p"=>1}

## Kolejność cyphera

["n", " ", "v", "b", "w", "l", "r", "j", "z", "e", "a", "o", "x", "m", "s", "f", "c", "g", "t", "h", "q", "u", "p"]

"r"=>"c", "c"=>"e", "y"=>"a", "s"=>"h", "k"=>"p", "p"=>"t", "m"=>"x",

"t"=>"b", "l"=>"z", "\text{\text{\text{"}}}"=>"l", "u"=>"n", "d"=>"q", "b"=>"o", "\text{\text{\text{"}}}"=>"v",

"q"=>"f", "a"=>"w"}

## Wynik!

h[?]zs[?]au[?]q[?]mcajsf[?]aurs[?][?]uf[?]whxehcfxbcj[?]c[?][?]x[?]ocv[?]xbcj[?]c[?][?][?][?]rczcbs

LOL

```
require 'rot13'
cypher.rot13

# => ruby jest fajnym jezykiem
# programowania do hackowania i zabawy
```

Jesteś administratorem.

Za pomocą RSA zakodowano kod PIN maszyny do kawy i przesłano do pracowników.

Każdy pracownik ma swój klucz publiczny.

Ty nie dostałeś swojego PINu -- co robić? (maszyna blokuje sie po 3 niepoprawnych kodach)

Przechwytujesz szyfrogramy

I oczywiście kryptoanaliza!

```
# public key
pk = { :n => 3233, :e => 17 }
# private key
prk = { :n => 3233, :d => 2753 }

# pin-code
pin = 2563
cypher = 2563**(pk[:e])%pk[:n]
# => 2258
plain = cypher**(prk[:d])%prk[:n]
# => 2563
```

```
pk = { :n => 3233, :e => 17 }
# cypher == 2258

9999.times do |pin|
  puts pin if pin**(pk[:e])%pk[:n] == 2258
end
```

```
pk = { :n => 3233, :e => 17 }
# cypher == 2258

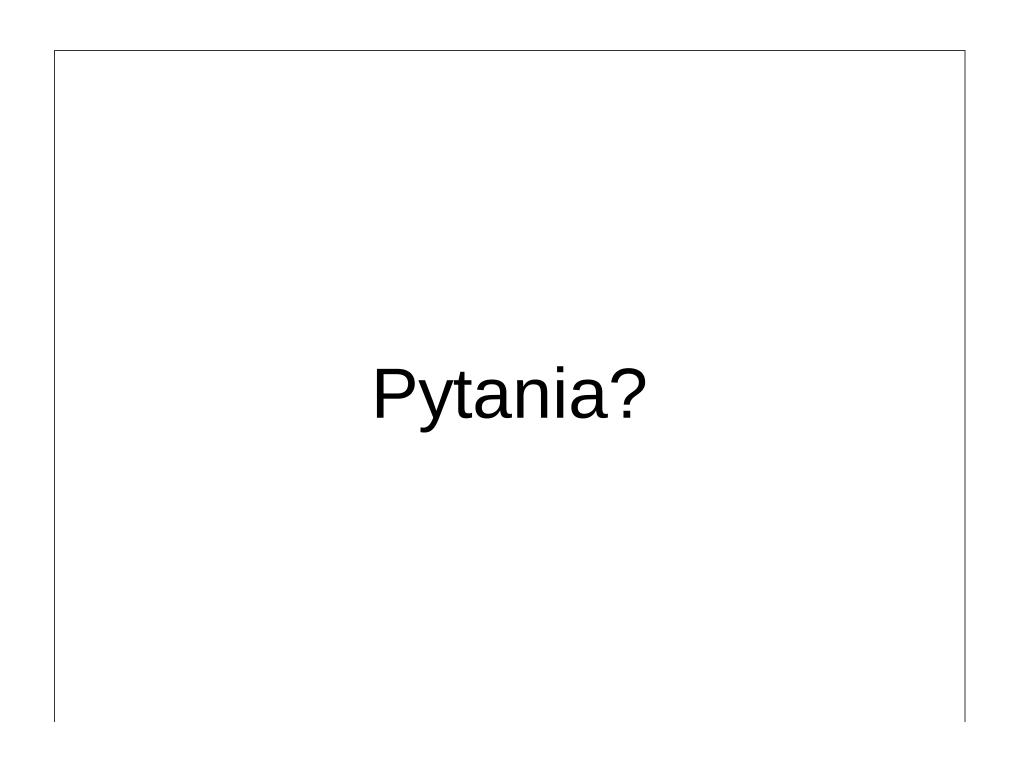
9999.times do |pin|
  puts pin if pin**(pk[:e])%pk[:n] == 2258
end

# => 2563, 5796, 9029
# MUHAHAHAHAHA! darmowa kawa ;p
```

## Pomysł

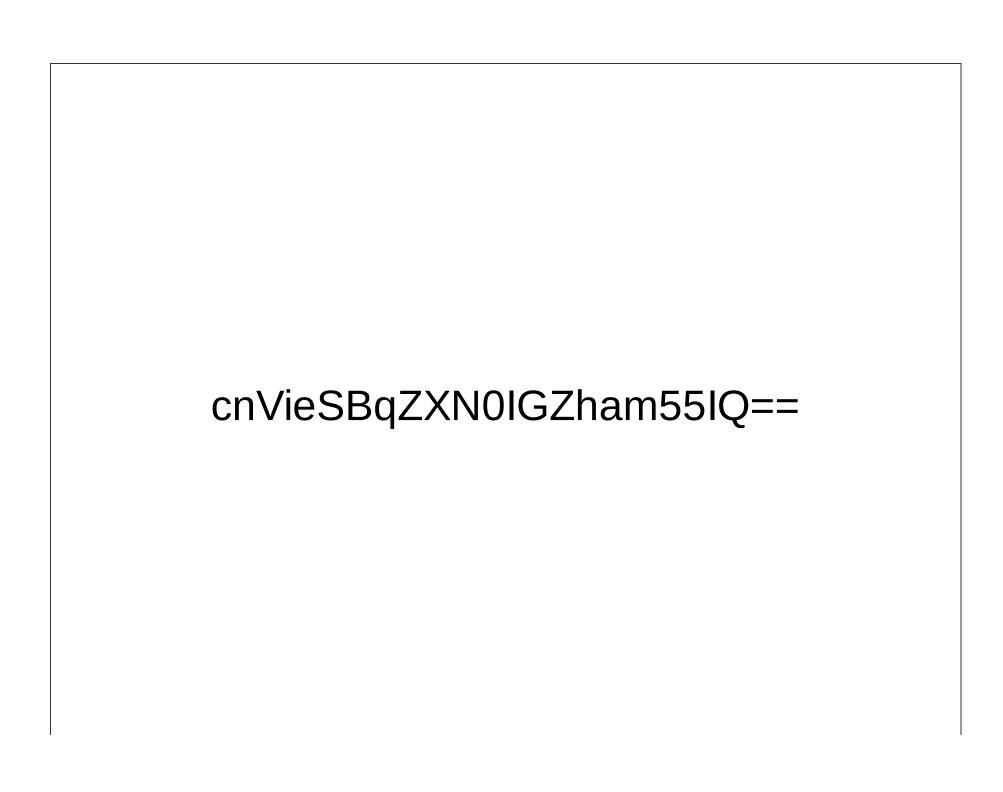
Stworzenie prostego i używalnego narzędzia do kryptoanalizy w ruby na kolejnym hackatonie

7 maj



Kryptoanaliza w .rb

# Na poczatku był cyphertext...



## Podobny do base64...

'ru' => 'cnU='

'rub' => 'cnVi'

'ruby' => 'cnVieQ=='

```
b64_regexp = /^([A-Za-z0-9=]){4,}$/
if b64_regexp =~ "cnVieSBqZXN0IGZham55IQ=="
    p "Works!"
end
# => "Works!"
```

```
require 'base64'
Base64.decode64 "cnVieSBqZXN0IGZham55IQ=="
# => "ruby jest fajny!"
```

ZWhvbCB3cmZnIHNud2FseiB3cm1seHZyeiBjZWJ0Z
W56YmpuYXZuIHFiIHVu\ncHhiam5hdm4gdiBtbm9uamw=

```
require 'base64'
cypher = [z-poprzedniego-slajdu]
if cypher =~ /^([A-Za-z0-9=]){4,}$/
   p Base64.decode64 cypher
end

# => "ehol wrfg snwalz wrmlxvrz
# cebtenzbjnavn qb unpxbjnavn vmnonjl"
```

```
def statistic(file)

result = {}
result.default = 0

file = File.new(file, "r")
file.each_line do |line|
   line.chomp!
   (line.size - 1).times do |time|
     result[line[time].chr] += 1
   end
end
```

#### Statytyka języka polskiego

#### Kolejność liter w języku polskim

```
["a", "i", "o", "n", "e", "z", "w", "r", "c", "y", "s", "k", "p", "m", "t", "l", "ł", "u", "d", "b", "j", "g", "ą", "ś", "h", "ę", "ż", "f", "ó", "ń", "ź", "ć", "c", "v", "x", "q"]
```

#### Statystyka cypher'a

{"e"=>3, "h"=>1, "o"=>2, "l"=>3, " "=>8, "w"=>3, "r"=>3, "f"=>1, "g"=>1, "s"=>1, "n"=>9, "a"=>3, "z"=>3, "m"=>2, "x"=>2, "v"=>4, "c"=>1, "b"=>4, "t"=>1, "j"=>3, "q"=>1, "u"=>1, "p"=>1}

#### Kolejność cyphera

["n", " ", "v", "b", "w", "l", "r", "j", "z", "e", "a", "o", "x", "m", "s", "f", "c", "g", "t", "h", "q", "u", "p"]

#### Odzyskane podstawienie ;-)

### Wynik!

h[?]zs[?]au[?]q[?]mcajsf[?]aurs[?][?]uf[?]whxehcfxbcj[?]c[?][?]x[?]ocv[?]xbcj[?]c[?][?][?] ]rczcbs

LOL

```
require 'rot13'
cypher.rot13

# => ruby jest fajnym jezykiem
# programowania do hackowania i zabawy
```

Jesteś administratorem. Za pomocą RSA zakodowano kod PIN do maszyny do kawy (każdy pracownik ma swój klucz publiczny). Ty go nie dostałeś -- co robić?

Przechwytujesz szyfrogramy

I oczywiście kryptoanaliza!