MysteryTwister C3

THE CRYPTO CHALLENGE CONTEST

Trendy Messenger

Author: Matthias Minihold, ECRYPT-NET

February 2018

Introduction

Let $pk = (p, \alpha, \beta = \alpha^x)$ be a public key of Textbook-ElGamal.

- Routinely sniffing on the WiFi, you get hold of an encryption $(\gamma, \delta) = (\alpha^r, m\beta^r) \in \mathbb{Z}_p^* \times \mathbb{Z}_p, r < p$, of a message m.
- You swipe to the latest version of TrendyMessenger (TM)'s open-source implementation on gitHoop and spot that the randomly generated r fits into 32 bit – you tweet: "LoL"!
- Implement an algorithm (e.g. in SageMath) to compute $m \in \mathbb{Z}_p$.

Challenge

Listing 1: Problem Instance

```
p = 246139997181309245630998777670253383692585291853967868751857421718171672049
10996312592164843173430891428993120199268223075414980740128871658667877466035939

alpha = 2

beta = 22795251975381661023656501765196259706412705817339345965751992984825289818
45890363349702188333585253476929213560021288361910432009371841113589260601895867

gamma =15229823750231334383800703430908579413110933106706766565240790862363380478
880928619819996978089232014142969448697641680441970715052469504872391555072540519

delta = 9117941499511792269515644754884178477935882887815606663067541248305514716
670707212785905268344517972740585176203248160118298898711714791666985451763180745
```

The solution consists of the message m. Please enter the solution without any spaces.

Hint: Take this code or leave it. Binary Search.

```
# To use bisect.bisect_right first import  bisect
# or use this version of binary search:
NOT_FOUND = -1 # redefine if needed
def bsearch(L, value):
        lo, hi = 0, len(L)-1
        while lo <= hi:
                mid = (lo + hi) // 2  # x//2 computes floor(x/2)
                if L[mid] < value:
                        lo = mid + 1
                elif value < L[mid]:
                        hi = mid - 1
                else:
                        return mid
        return NOT_FOUND

def bsearchTest(n=11):
        L = range(n)
        for elt in L:
                assert bsearch(L, elt) == elt

        assert bsearch(L, n) == NOT_FOUND
        return "bsearchTest passed!"

bsearchTest()
```