Plaid CTF 2014: graphs

Category: Crypto Points: 200 Description:

> In this era, block ciphers hadn't even been invented. The Plague created this system based on problems he knew to be NP hard, but there must be something you can do to decode his messages.

Write-up

So, first of all, what are our files?

- Ciphertext - encoded array of 1024 big numbers.
- Plaintext - the sum of 64 numbers from the cleartext.
- Public key - encoded graph (list of lists).
- Private key - indexes of numbers in the cleartext, which generate a cleartext.
- A `genkey.py` script that performs the encryption.

So we have the public key with 1024 vertices, but we need only 64 of them.

Let's take a closer look at how the key generation works:

```
vertices = range(self.keylen)
privkey = random.sample(vertices,self.keylen>>4)
#privkey = random 64 vertices

tocover = set(vertices).difference(set(privkey))
#tocover = all others vertices

G = [0]*self.keylen
for v in vertices:
  G[v] = []
#just graph init

#!!! the most important part !!!
while len(tocover) > 0:
  src = random.choice(privkey)
  dst = random.choice(list(tocover))
  G[src].append(dst)
  G[dst].append(src)
  tocover = tocover.difference(set([dst]))
```

So `len(tocover) = 1024 - 64 = 960`. Let's take 2 random vertices, 1 from the private key, 1 from the `tocover` set, and we 'connect' them. We can then compute lengths of private keys vertices (count of connected element) = 960 / 64 = 15 (will be same for all `keylen`). Now we know, that our `privkey` vertices have 15 elements, but what with others?

```
others = list(set(vertices).difference(set(privkey)))
for o in others:
  for n in others:
    if random.getrandbits(5) == 0:
      if o not in G[n]:
        G[n].append(o)
        G[o].append(n)
```

We connect two vertices `o` and `n` only in 1/32 of all cases (`if random.getrandbits(5) == 0`). But our graph is symmetric, so actually it's in 1/16 of all cases. Okay, so for every vertice `o` we have about `960 / 16 = 60` connected elements versus 15 elements in private key.

Now we need only get indexes of 64 vertices with the smallest count of connected elements. I did it this way:

```
privkey = []
for index, el in enumerate(self.pubkey):
  if len(el) <= 30:
    privkey.append(index)
self.privkey = privkey
```

Let's decrypt the ciphertext using that key `22756295994291953255513854050290361717820460851310522145563405409616 62` . The cleartext is:

```
The flag is: 3veryb0dy_poops~
```

The flag is `3veryb0dy_poops~` .

Other write-ups and resources

- https://fail0verflow.com/blog/2014/plaidctf2014-crypto200-graphs.html
- Source code for this challenge, released after the CTF
- Russian
- http://piggybird.net/2014/04/plaidctf-2014-reversing-300-paris/