Crypto (short)

EASYDSA

In the code there is ($k=gen^{u*m} \pmod q$), so if we can get ($k=1$) decoding other stuff will be much easier. But u is random and there is assert ($m\%(q-1)!=0$) so we can't use Fermat's little theorem that ($a^{q-1}=1 \pmod q$).
But maybe there is other exponent that gives ($q^{exp}=1$)?
If so then exp divides (q-1) so we check divisors of (q-1). (q-1)/2 was first thing I checked and it works.
With that decoding message is straightforward.

open-gyckel-krypto

Let's say ($mod=10^{250}$, $a=p\%mod$, $b=q\%mod$).
Then ($p=mod*b+a$, $q=mod*a+b$).
Then ($n=mod^2*a*b + mod*(a*a+b*b) + a*b$).
(a*b) is not longer than 500 digits, (a*a+b*b) is no longer than 501 digits (and first can only be 1 if 501 digits).
So ($n\%mod=(a*b)\%mod$, $n/mod^3=(a*b)/mod+(+1)$)
So for both possibilities we do following:
From (a*b)%mod and (a*b)/mod we know a*b.
Also we can calculate mod*(a*a+b*b) as (n-a*b*(1+mod^2)).
So we can get (a*a+2*a*b+b*b) what is (a+b)^2.
At this point we can check if it's square and it is in second possibility, so from now on we calculate only this.
In the same manner we calculate (a-b)^2 and from that we get a and b. That gives us p and q so we can decode message.

Tulpan257

k is length of flag + 1, all polynomials are over Zmod257.
We are given random polynomial of degree k+1 and 107 values a_i. Let's call m(x) - polynomial masked. We know that with about 60% chance a_i is m(i) otherwise it's some random number.

If we have 26 correct values for m(i) we can interpolate it and that's our goal.
So I wrote a program that takes 26 random indices and interpolates m(x) assuming those values are correct.
To check if it's correct m(x) I check for each a_i if it fits. Bad guesses should not match for more than 30 indices and correct guess should match for about 60. So each time guess is better than all previous I print it.
The probability that all choose indices is about (0.6**26) so about one in a million (not really but kind of). With my not so efficient c++ implementation it took about 9min with about 3.5 million guesses.
Here's the polynomial:
138 + 65*x + 77*x^2 + 143*x^3 + 200*x^4 + 174*x^5 + 177*x^6 + 59*x^7 + 122*x^8 + 87*x^9 + 28*x^10 + 150*x^11 + 123*x^12 + 53*x^13 + 46*x^14 + 105*x^15 + 199*x^16 + 133*x^17 + 76*x^18 + 235*x^19 + 95*x^20 + 215*x^21 + 233*x^22 + 158*x^23 + 181*x^24 + 136*x^25.

So know we have to get flag from m(x). We know flag was converted to polynomial of degree 25 and we are given r(x) of degree 26. We know that (m(x)=(r(x)*flag(x))%(x^26+1)).
If we say that flag(x)=sum(b_ix^i) we get 26 equations for b_i. We also know that (b_{25}=0). Then we solve this system of equations and get the flag.

pgp-com

First I read A LOT about pgp format from link.
So we are given private key A, and three messages. First and third are encoded with public keys A, B, C. Second has only keys B, C so we don't know how to decoded.
But in third message there is:

```
We have received some indications that our PGP implementation has problems with randomness.The dev team is currently working on fixing the issue.
```

From now on our goal is to decode first and third session key.
I couldn't find any useful tools, so I did is mostly manually. My steps were:

- export private key to pem,
- using openssl get p, q etc.
- using gpg --list-packets --verbose get data with encrypted session keys,
- decrypt session keys using private RSA key.

Then we see that:

```
```

```
key1 = '0000000000000000000000000000000000000000000000000000000000001336'key3 =
'0000000000000000000000000000000000000000000000000000000000001338'
```

(keys in hex).

So the guess is that: `key2 = '0000000000000000000000000000000000000000000000000000000000001337'`
I didn't want to decode AES and everything else manually so I did following (#nr is reffering to linked document):

- constructed message as in #7.2.1,
- encrypted it with given private RSA key,
- constructed Public-Key Encrypted Session Key Packet as in #5.1
- in message2 swapped one of the said packets with my packet,
- decoded it using gpg.

That gives us message containing flag.

```
From: Midnight Sun CTF Admin <admin@midnightsunctf.se>To: Midnight Sun CTF Devteam maillist
<devs@midnightsunctf.se>
Subject:
Date: 2019-04-02 17:27:07
Hi,
How could you implement a system with such bad session key generation!?
Please remeber that midnight{sequential_session_is_bad_session} in the future...
Best regards,
CTF Admin
```

Revision #4
Created Mon, Apr 8, 2019 12:38 PM by miszcz2137
Updated Sun, Jul 14, 2019 2:57 PM by Lorak_