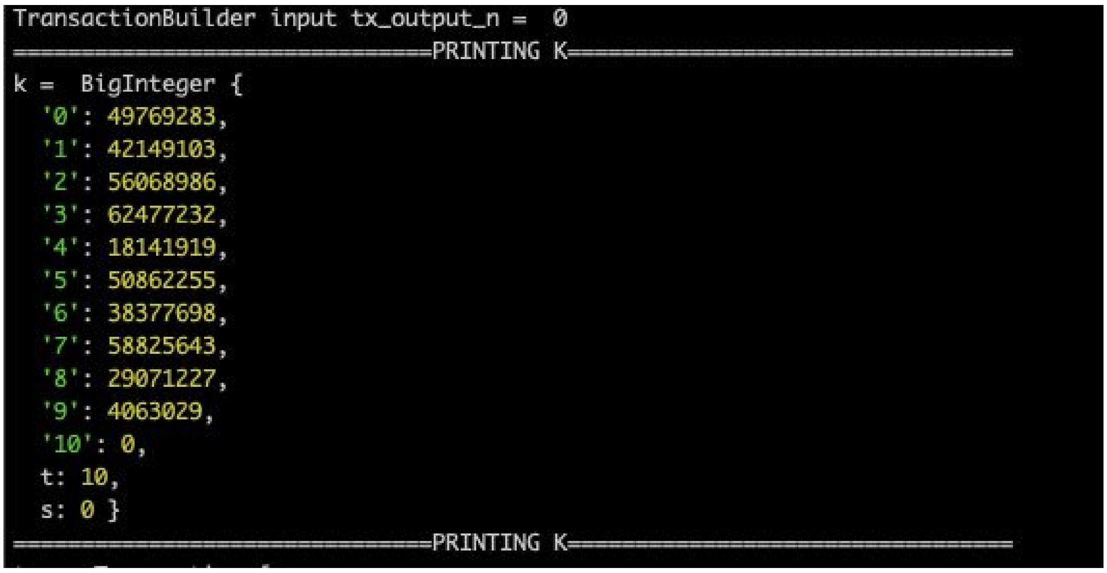
Here's my progress briefly:

-I searched a lot on what API service to use, among the most popular ones.
At the end, I think using blockcypher's API to hit the bitcoin's official testnet is the best solution and free of charge of course.
(Other services required me to pay to execute more than 5 POST requests to their servers)
--> https://live.blockcypher.com/btc-testnet/

-After experimenting a lot with python scripts of mine, and testing available scripts online, I found some nodejs(javascript) scripts to execute some basic functions like, generating bitcoin addresses, public, private keys, funding a test account, querying bitcoin's address information on the blockchain to find out data on balance, received and sent transactions and also the most important one, broadcasting transactions through code, into the official testnet bitcoin blockchain.

-Now the most most important part of my research and work is to manipulate the k (nonce) used in signing bitcoin transactions from being random to being an encoded, encrypted message.

-Ok so in theory, in the paper, it was easy to say, but in practise is way more difficult, so I think that succeeding making it work should be very significant and innovative.
I have downloaded the library bitcoinjs-lib@2.2.0 , I found the file ecdsa.js, where all the magic becomes reality, I found the k (nonce) and printed out right before using it to execute the core equation : s=k^(−1)*(z+r*d) mod(n)
What I found out is that it's not just a random number but an object :



```
TransactionBuilder input tx_output_n =  0
===============================PRINTING K=================================
k =  BigInteger {
  '0': 49769283,
  '1': 42149103,
  '2': 56068986,
  '3': 62477232,
  '4': 18141919,
  '5': 50862255,
  '6': 38377698,
  '7': 58825643,
  '8': 29071227,
  '9': 4063029,
  '10': 0,
  t: 10,
  s: 0 }
===============================PRINTING K=================================
```

I guess that it selects just one of them and use them in the equation.I will have to dig deeper to find out exactly what this code does.But I feel like s: 0, means selects 0 at first and tries if the result is not good s becomes 1 and 2 ..etc until the result is satisfying.
So now a core question arises, how to encode or encrypt a line (command) into an 8 digit decimal number, I feel like there are many ways to do it, but I would like to hear your opinions as well.

-Assuming that we succeed on this.Then the next step is to make sure the transaction is still a valid one and can be verified by the network, doing so will be a very good step.

-After that, getting the transaction from the network is easy, I have done this, it’s just an API call.

-Decoding the transaction to get the k should also be a bit tricky.
According to the paper, if the receiver knows the private key d, he can easily retrieve the message solving the following equation: $kmsg = s^{(-1)} * (z+r*d) \bmod(n)$.

In conclusion, I am now at the step of encoding a message and inserting it into the signature as k (nonce).Any help would be really appreciated.

Thank you for your time,
Best regards,
Aristofanis.