Chain Channels Implementation

Based on this paper: https://www.annessi.net/data/2018-subliminalblockchain_preprint.pdf
I will try to make an actual implementation of the protocol,
maybe it can be the same at the beginning,
but we can improve some features and add some more functionality if we think it is necessary.

The plan is this one:

- Firstly, make a python script which will be the client of our application (the bot) and will be responsible for:
    - Connecting to one (maybe more in the future) public blockchains (Bitcoin and others who use ECDSA signature scheme).
    - Read transactions and find the ones that have the public key of the Bot-Master.
    - Gather these transactions and store them, till we find the final final transaction, where the bot-master will reveal the private key.
    - Once the bots have successfully decoded the previous messages using the private key, make a list of all the commands that are subject to execute in the system.
    - Execute the commands ordered by the botmaster.
- Secondly, make a python script which will be the bot-master application and will do the following:
    - Have an intuitive UI, something simple, having a menu, with numbers at each command, and when the user puts a number in the cmd, this number will be translated in a transaction with a subliminal message in it and directly sent in the blockchain, then if the final number is given it will leak the key and close the app.
    - Connect to the same public blockchain.
    - Make very small transactions from the address As with a subliminal message encrypted in each one of them.
    - At the last message reveal the key.
    - Wait and see the results from the bots.

*Make a list of commands that will be encrypted and mean something.
For example we may want to have a command that does something simple (ie write something in a text file), to something more complex like make this PC a miner and send money mined from the PC to the botmaster, download and run a keylogger to this system, make a DDOS attack, etc.

**To secure the private key leakage by the botmaster to the bots, We will be using the Section V.D method which is : Concealed Key Leakage with a Pre-Shared Nonce.Which actually says, intentionally violate the rule of the signature that the nonce should be unpredictable.So we will pre-configure at the bots a random secret nonce, Kleak, it is only known at the bots and the botmaster and remains unpredictable to everyone else and we will use only one address for the botmaster the As.So the sender will send all commands and leakage key message from the same address As.

***We can also think of ways of spreading this “virus” (clients).