zk example

proof-of-factor

- I want a smartcontract that gives 1ETH to whom factors a number
- `play(uint256 p, uint256 q)` can be frontrunned with higher gas
- I need to proof to the smartcontract that I factored it without revealing p,q

Creator

Create the verifier
Deploy the SC with challenge

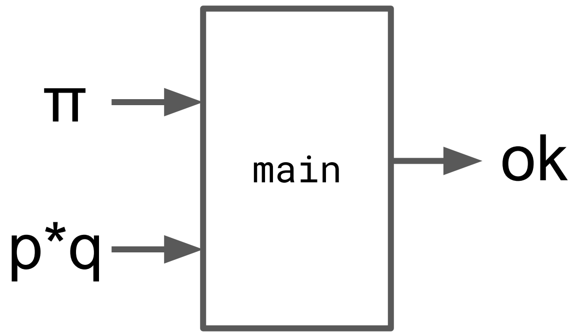
```
uint256 challange;
prove(proof) {
  verify(challange,proof)
  msg.sender.transfer(1 ether);
  selfdestruct();
}
```

User

Read challange from SC
Factorize challange and get p,q
Generate proof that I own p,q
Call SC `prove(proof)`

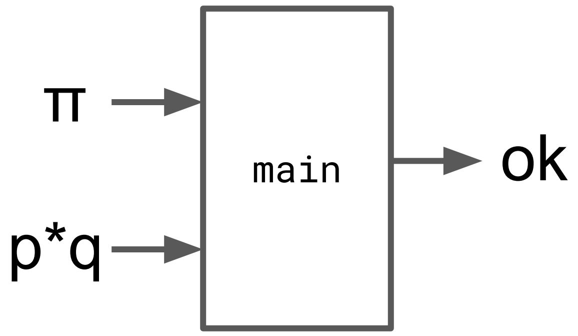
verifier v0

```
def main(private field p, private field q) -> (field):

  return p * q
```



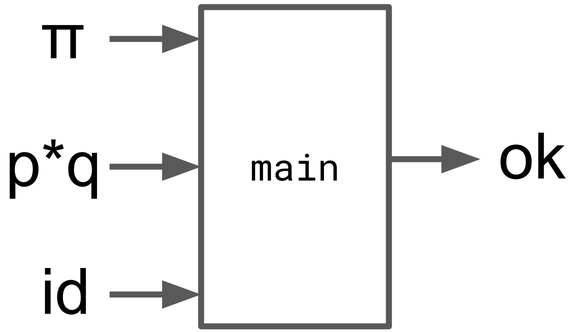
verifier v1

```
def main(private field p, private field q) -> (field):
  0 == if 1 == p then 1 else 0 fi
  0 == if 1 == q then 1 else 0 fi
  return p * q
```



verifier v2

```
def main(private field p, private field q, field id) -> (field):
  0 == if 1 == p then 1 else 0 fi
  0 == if 1 == q then 1 else 0 fi
  0 == if id == id then 0 else 1 fi
  return p * q
```

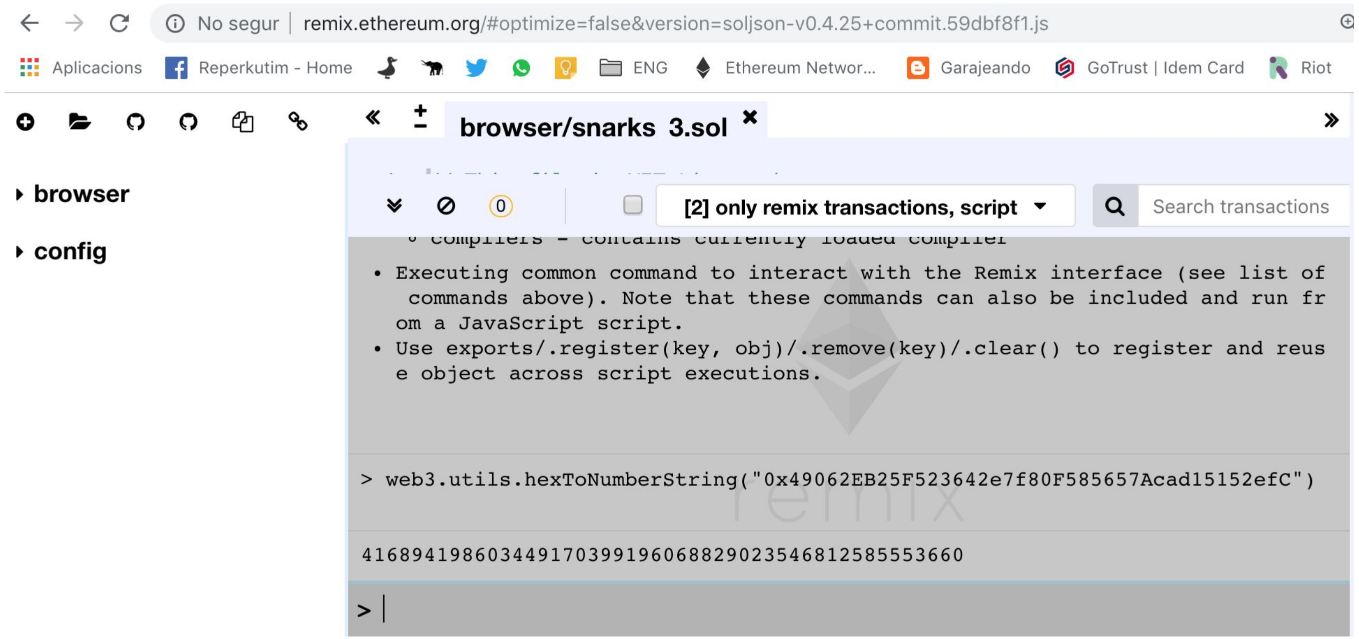


set-up zokrates and generate verifier

```
docker run -ti zokrates/zokrates /bin/bash
# vim is not installed by default, run in another window
#   docker exec -u 0 -it <container> bash
#   apt update
#   apt install vim
# create file sybil.code with verifier v3

# generate the circuit
./zokrates compile -i sybil.code

# generate trusted setup
./zokrates setup

# generate solidity code
./zokrates export-verifier
```

set-up get msg.sender as decimal number

```
# we need to know the decimal address of our ethereum address
#  go to a remix console and execute web3.utils.hexToNumberString(address)
```



```
  o compilers - contains currently loaded compiler
• Executing common command to interact with the Remix interface (see list of
   commands above). Note that these commands can also be included and run fr
  om a JavaScript script.
• Use exports/.register(key, obj)/.remove(key)/.clear() to register and reus
  e object across script executions.

> web3.utils.hexToNumberString("0x49062EB25F523642e7f80F585657Acad15152efC")

416894198603449170399196068829023546812585553660

>
```

generate proof that I can factor 6

```
# compute the witness
./zokrates compute-witness -a 2 3 0x49062EB25F523642e7f80F585657Acad15152efC

# generate the proof
./zokrates generate-proof

# modify the generated verifyer.sol and add

event CanFactor(bool yes);
function check() public returns(bool) {
    var proof = Verifier.Proof ({  /* put here the output of generate-proof */ });
    uint[] memory inputValues = new uint[](2);
    inputValues[0]=uint256(msg.sender);
    inputValues[1]=6;
    emit CanFactor(verify(inputValues,proof)==0);
}
```

test

```
# deploy in ropsten

# check that it works using the good sender
# call the check() method using the account 0x49062EB25F523642e7f80F585657Acad15152efC

# check that it works using bad sender
# call the check() method using another account
```