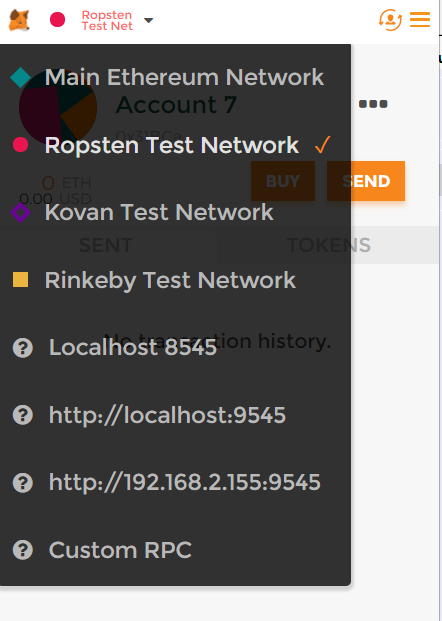
Remix IDE and Metamask

One of the best and easy-to use tool do build, deploy and debug smart contracts is Remix IDE (remix.ethereum.org ), which is an online browser based-compiler and IDE (Integrated development environment) that enables developers to build and deploy smarts contracts using the Solidity language but also to debug transactions. More about this you can find on <https://github.com/ethereum/remix-ide>

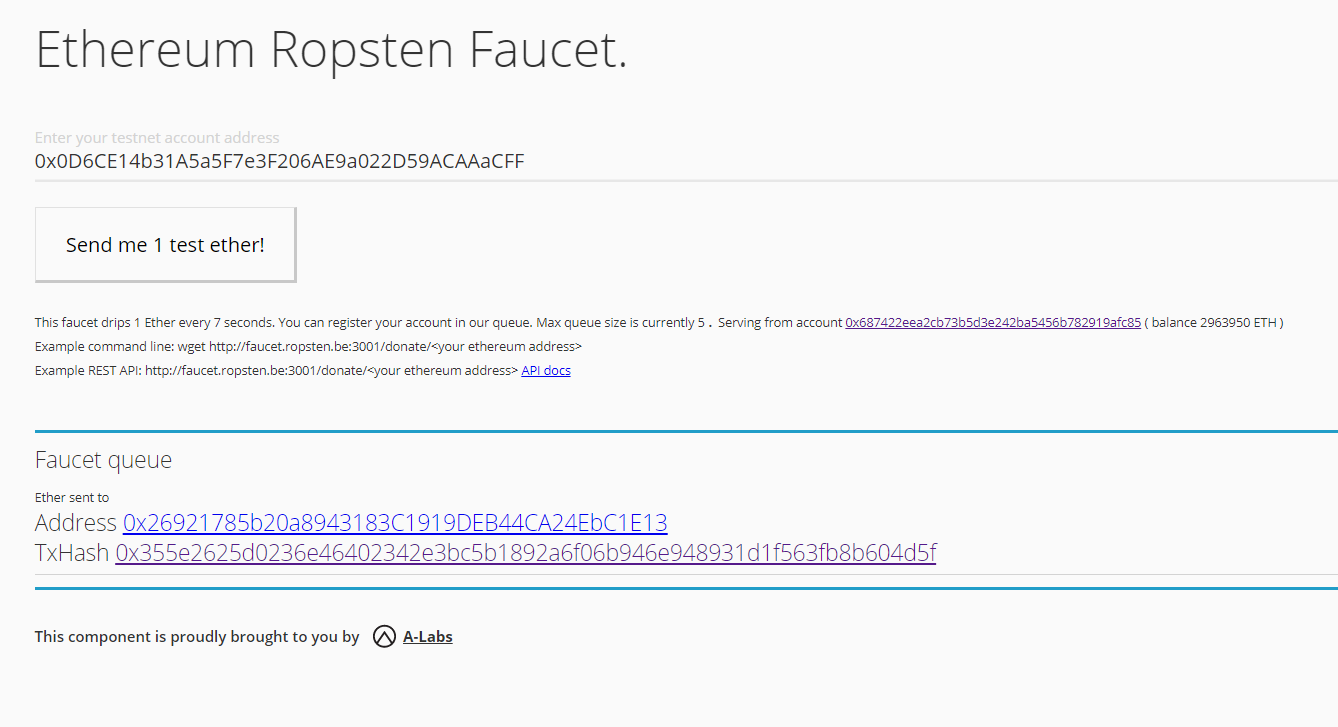
One of the easiest way to access the Ethereum blockchain is to use the browser extension Metamask, that allows anyone to interact with Ethereum blockchain without running a full node. <https://metamask.io>

How to install, create wallet, etc about metamask go to Radu Part …

For our example we will connect to the Ropsten Test Network, but Metamask allows you to connect to all the official Ethereum test networks, local running node and even the Main Ethereum Network. As always, we have to remind you: Safety first! Don`t forget that deploying smart contracts and making transactions on the main network will imply some risk. Always develop and test your code vigorously on the test environments before deploying it on the main one.



In order to be able to deploy code on the Ethereum network, we will need some funds, some test Ethers. We can use that by mining, or by using a faucet service. To get test Ether you will need to access http://faucet.ropsten.be:3001, then add your Ropsten Ethereum address, for example: 0x0D6CE14b31A5a5F7e3F206AE9a022D59ACAAaCFF



If we go on TxHash (Transaction Hash), we will see we just received 1 ETH, enough for us to play with the Ropsten network:

<https://ropsten.etherscan.io/tx/0x355e2625d0236e46402342e3bc5b1892a6f06b946e948931d1f563fb8b604d5f>

In order to interact with a smart contract on the Remix IDE and Metamask, we will need first to deploy one. Let`s follow the bellow example for a simple contract that stores a value:

**pragma** solidity **^**0.4.0;

**contract** SimpleStorage {

**uint** storedData;

**function** set(**uint** x) **public** {

storedData **=** x;

}

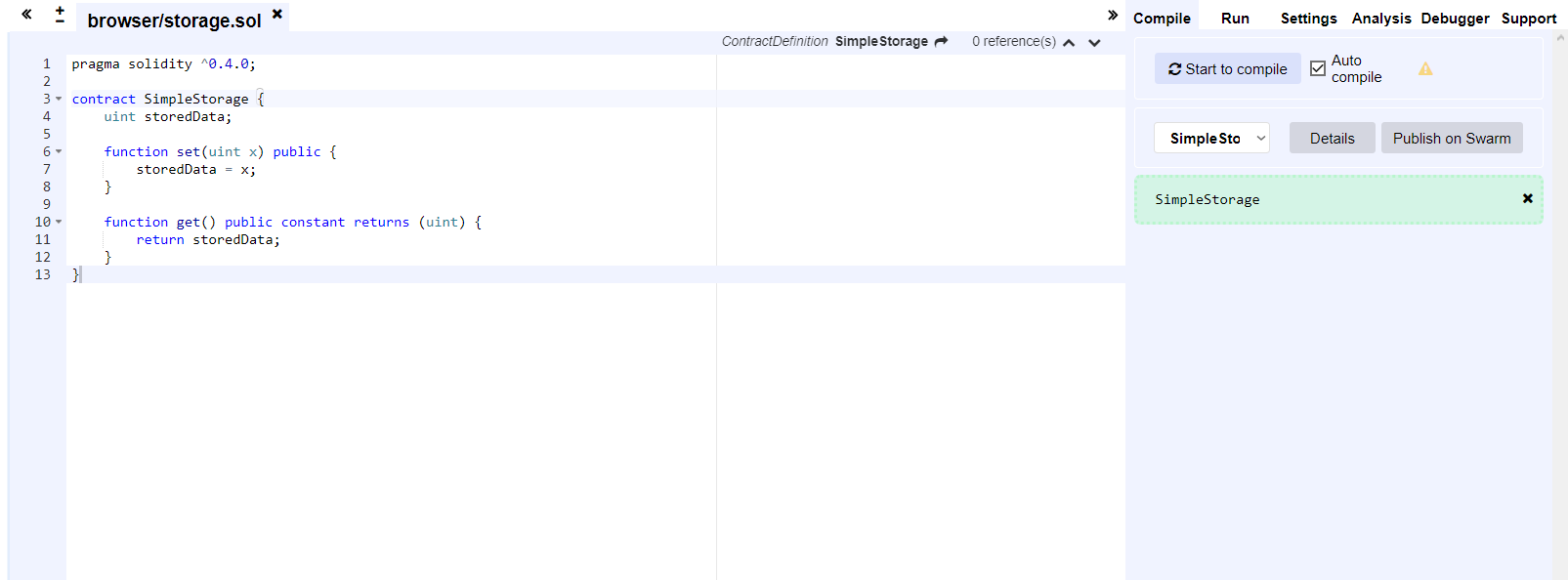
**function** get() **public** **constant** **returns** (**uint**) {

**return** storedData;

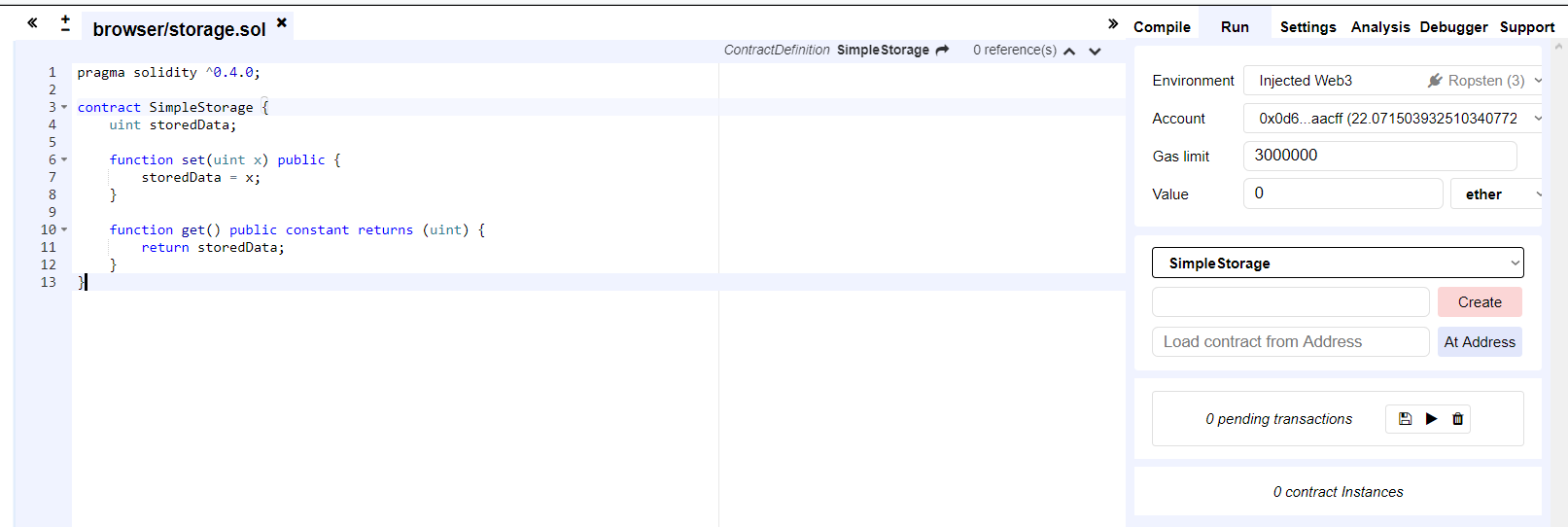
}

}

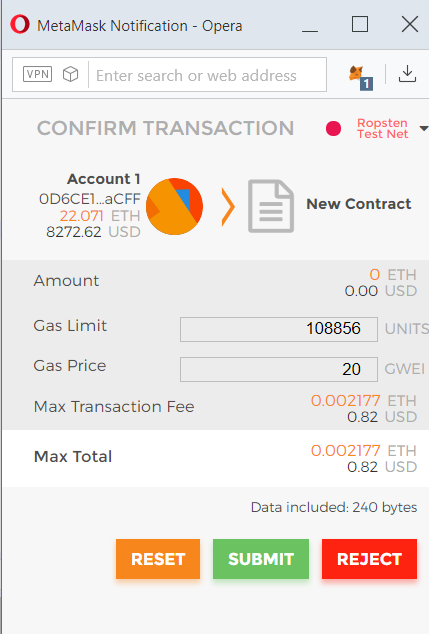
Just copy and paste the code in the console. You should now see something like this:



After pasting the code in the Remix IDE compiler, we will see that there is no error found on the contract, so we can go in the **Run** tab (just next to **Compile**). Here we will be able to see the network that we are using (in our case Ropsten), the account (the Metamask wallet address) and the Gas Limit (the Gas has direct conversion with ETH, and it is used to pay fees in order to use the ethereum blockchain; gas limit = the maximum amount of gas which we are willing to pay in order to deploy our contract)



To deploy our contract, you will need to press on **Create,** and a popup will appear to submit our transaction.

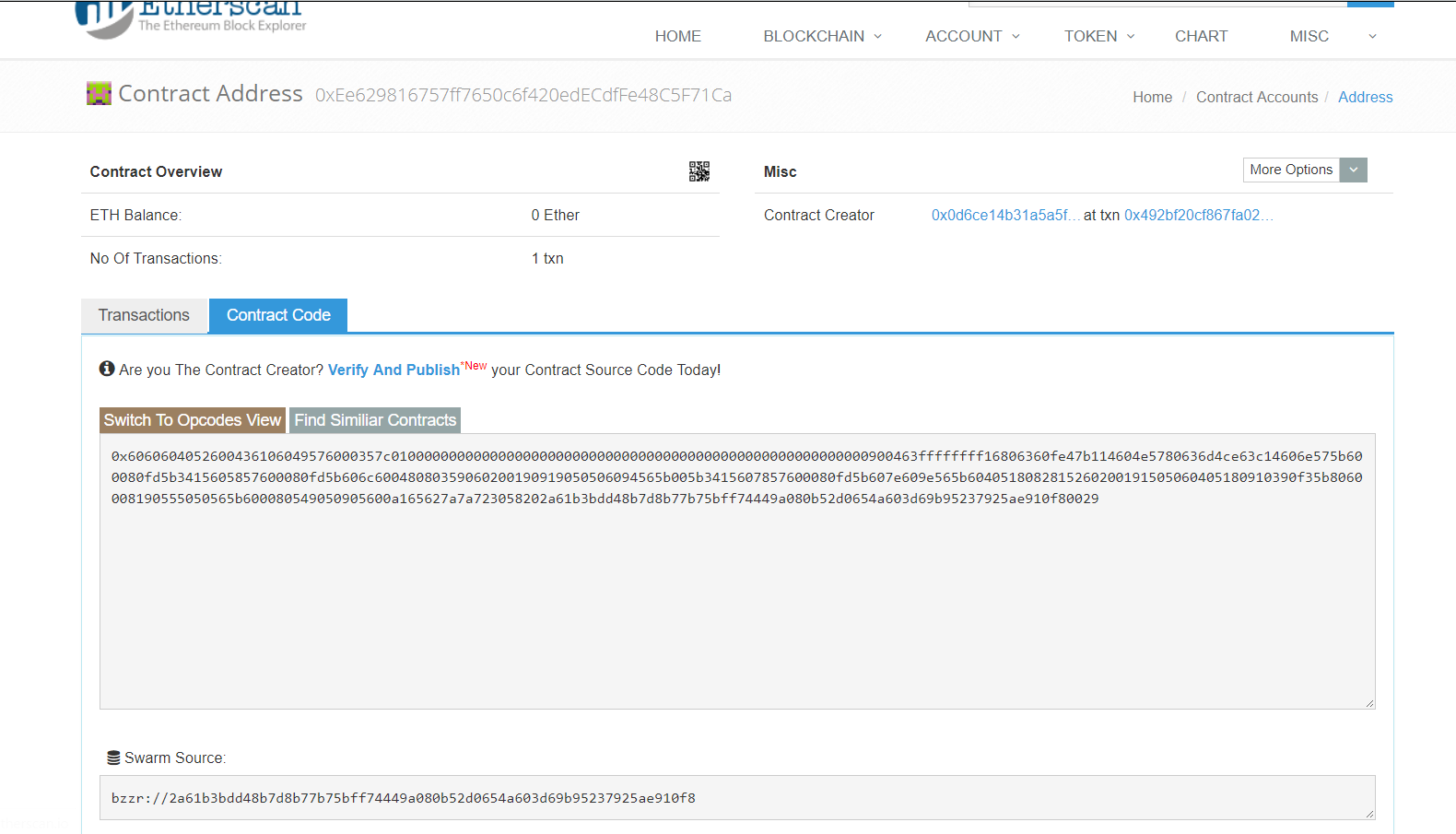


Thanks to Remix interface we will be able to find helpful information easily like when the transaction has been minter, from which address, the amount of gas consumed or the transaction hash.



Here is the transaction hash that we`ve generated: <https://ropsten.etherscan.io/tx/0x492bf20cf867fa0262700fb0645bfa20e7be216e4d6fdf4b5e3ed7a8bd9e4ad4>, using an alternate way to view the same info as in Remix IDE, but on [Ethereum Ropsten Explorer](https://ropsten.etherscan.io/).

In this explorer we can verify our smart contract, by making it public, and allowing anyone to read it and interact with it from their interfaces. In order to do that, we need to navigate to our contract 0xEe629816757ff7650c6f420edECdfFe48C5F71Ca, click on it then Contract Code –> **Verify and Publish**.



You will need to add the same Solidity contract code, same code as the one we`ve entered in Remix IDE before:

**pragma** solidity **^**0.4.0;

**contract** SimpleStorage {

**uint** storedData;

**function** set(**uint** x) **public** {

storedData **=** x;

}

**function** get() **public** **constant** **returns** (**uint**) {

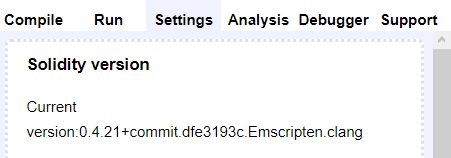
**return** storedData;

}

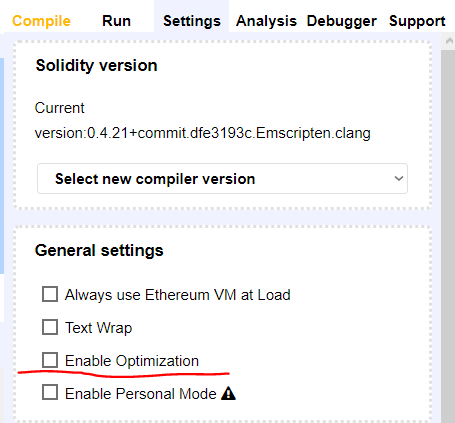
}

**Contract name:** in our code the contract it’s called **SimpleStorage**

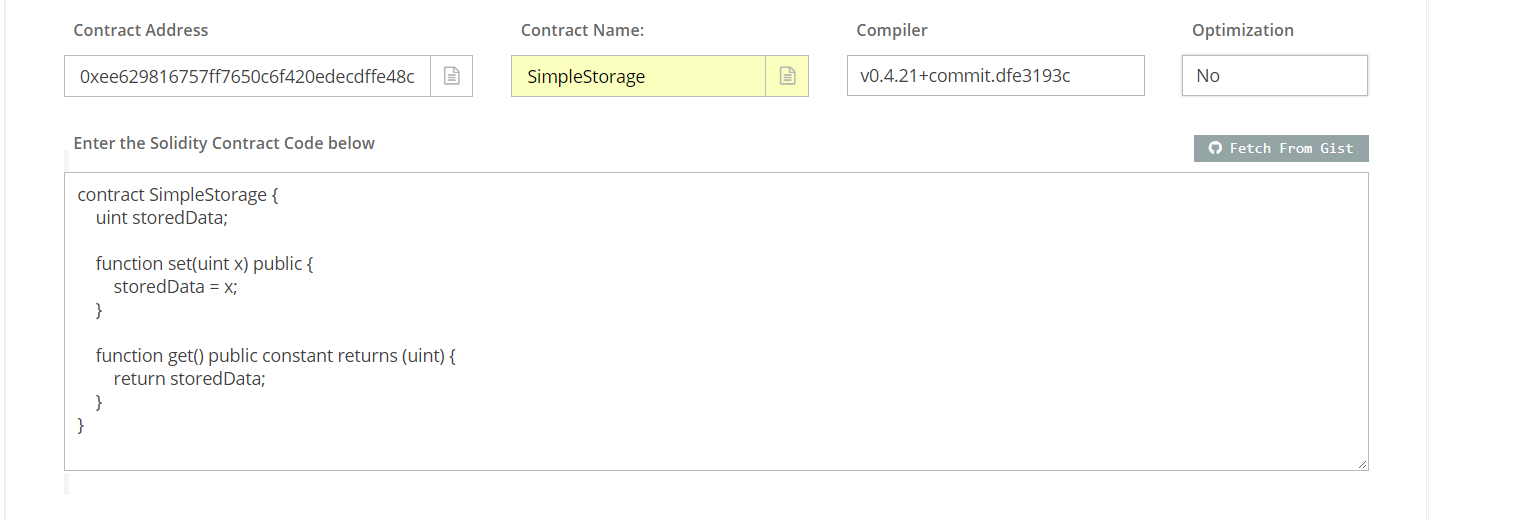
**Compiler:** in remix if we go on the Settings tab, we will see that our solidity compiler version is **0.4.21+commit.dfe3193c.Emscripten.clang**



**Optimization:** No, the default setting in Remix ide it’s without optimization, if you want optimization enabled you can check in settings tab: **Enable optimization**



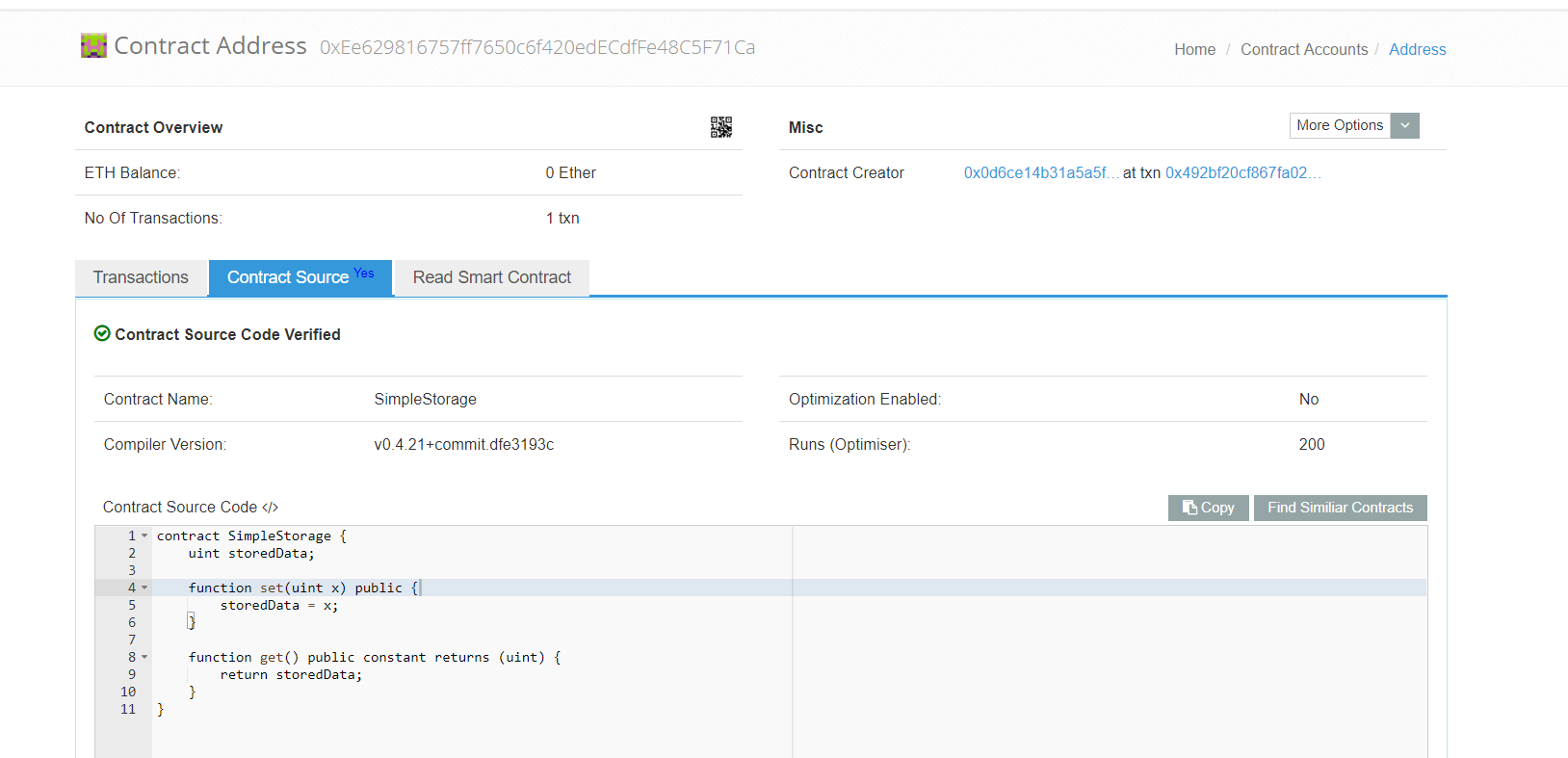
Now the verification process screen should look like this:



Then, we need to press Verify and Publish, and if everything matches with the Remix IDE deployed code, our contract will be verified.

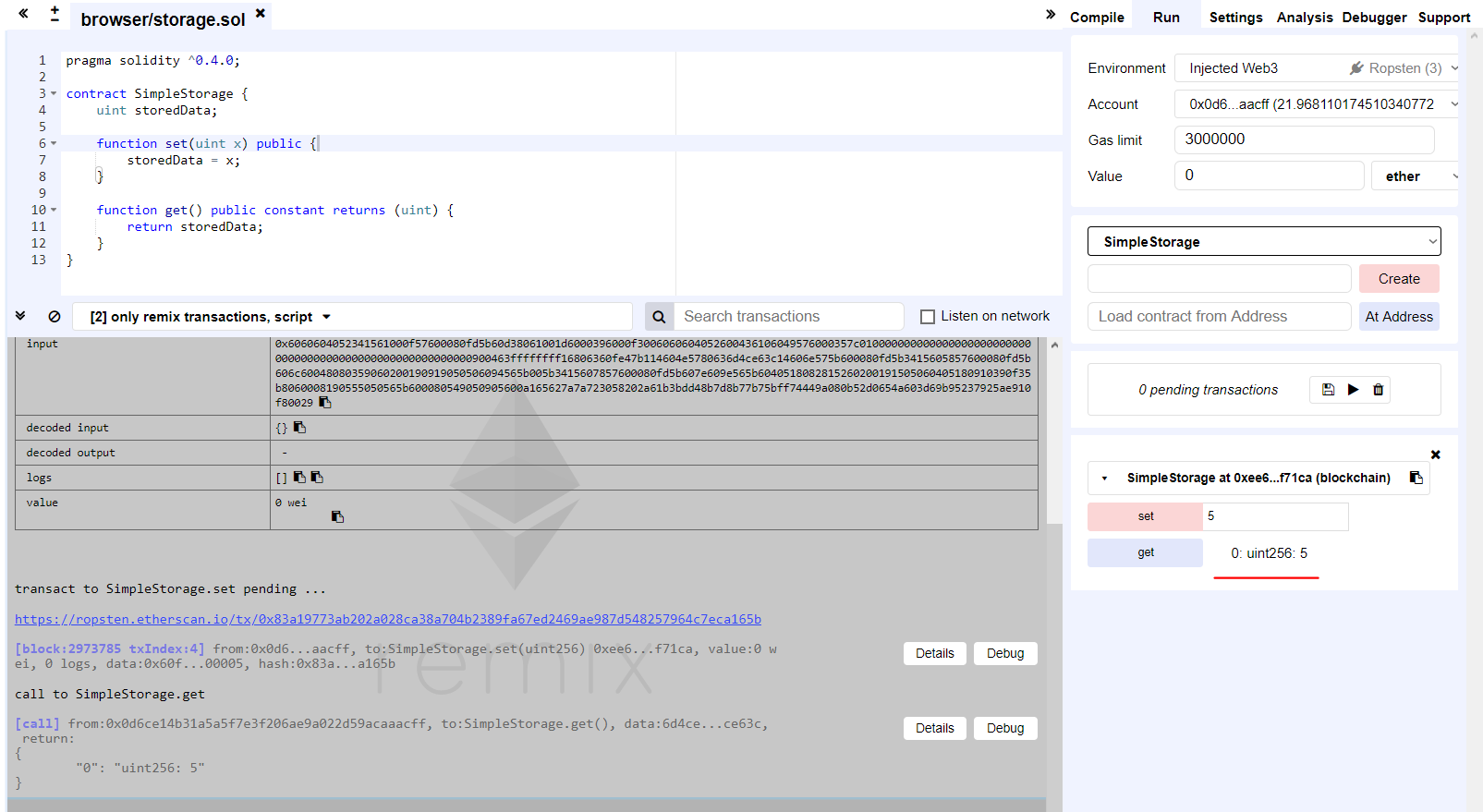


Starting now, everyone can see Contract Source and read the smart Contract that we`ve wrote.

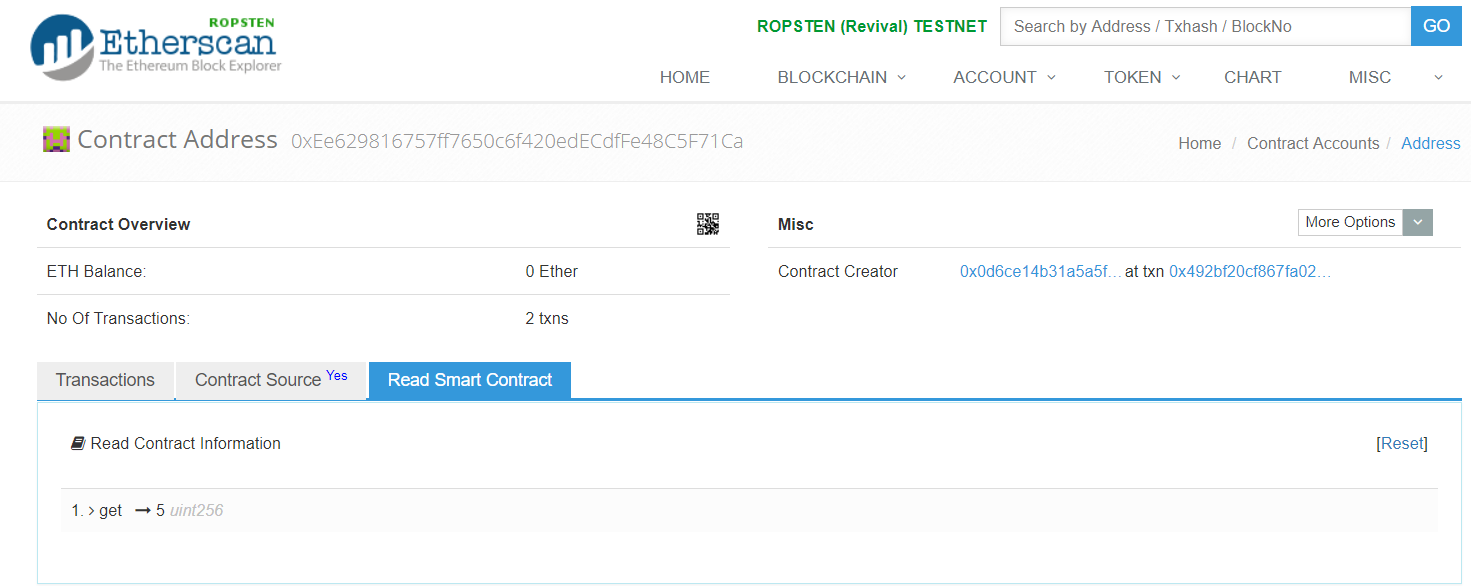


Now that we have deployed and verified our contract, let`s see how we can interact with it from Remix IDE:

* Once we deploy it, we will be able to see our functions from the smart contract: **set** and **get** can be used to modify or to retrieve the value of the variable **StoredData**
* If we want to add the value 5, we need to press on **set**, to confirm the transaction, then when we will press again on **get**,the contract will return our value – 5.



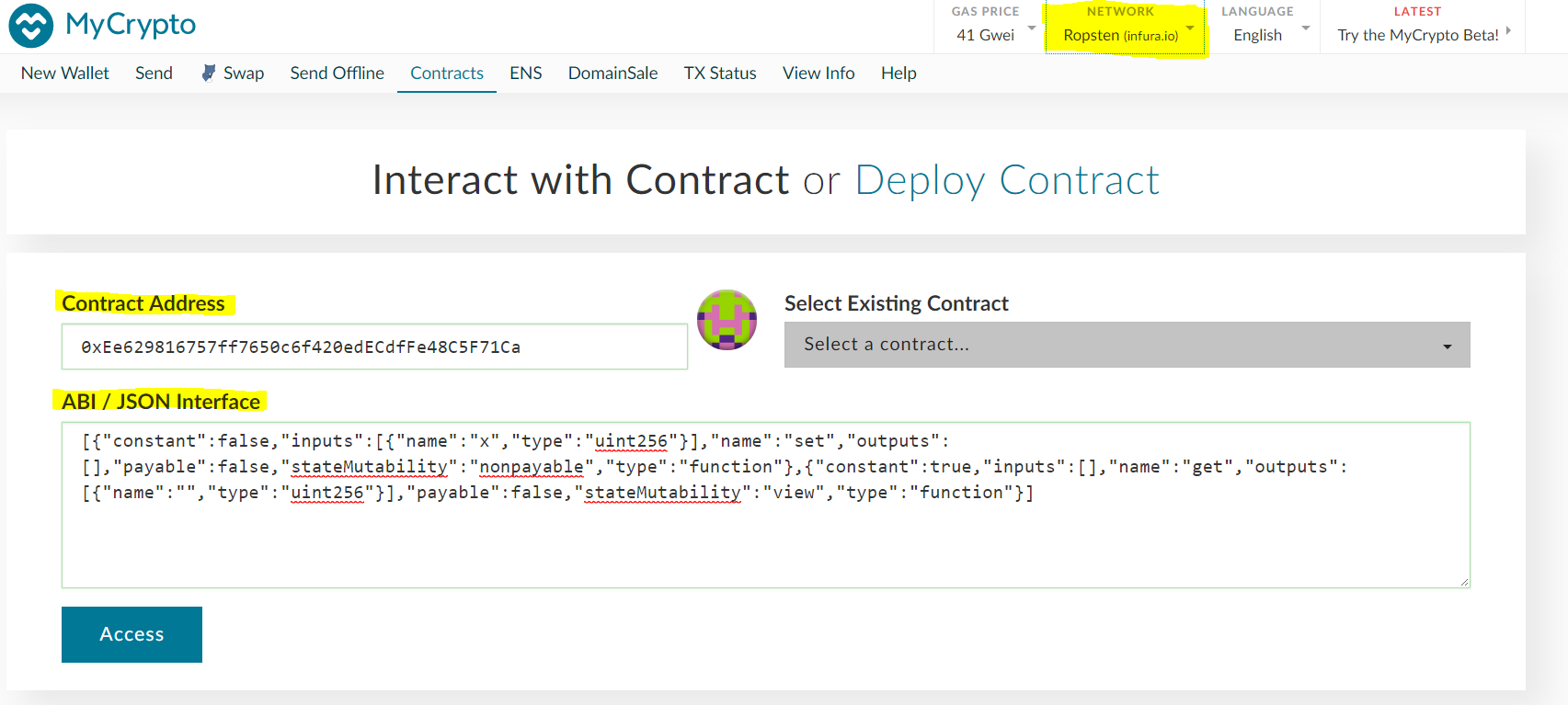
We can check the value stored on our smart contract`s variable on etherscan: <https://ropsten.etherscan.io/address/0xee629816757ff7650c6f420edecdffe48c5f71ca#readContract>



In this moment , everyone can call the set function and change the storedData variable as the contract doesn’t have an owner defined.

This time we will call the **set** function from another Interface: <https://mycrypto.com> or <https://www.myetherwallet.com> , be sure to always double check when you access these sites because there are a lot of phishing sites which are trying to steal your money.

First we have to select the network on <https://mycrypto.com> in order to access our contract on Ropsten Network, so you should see something like this:



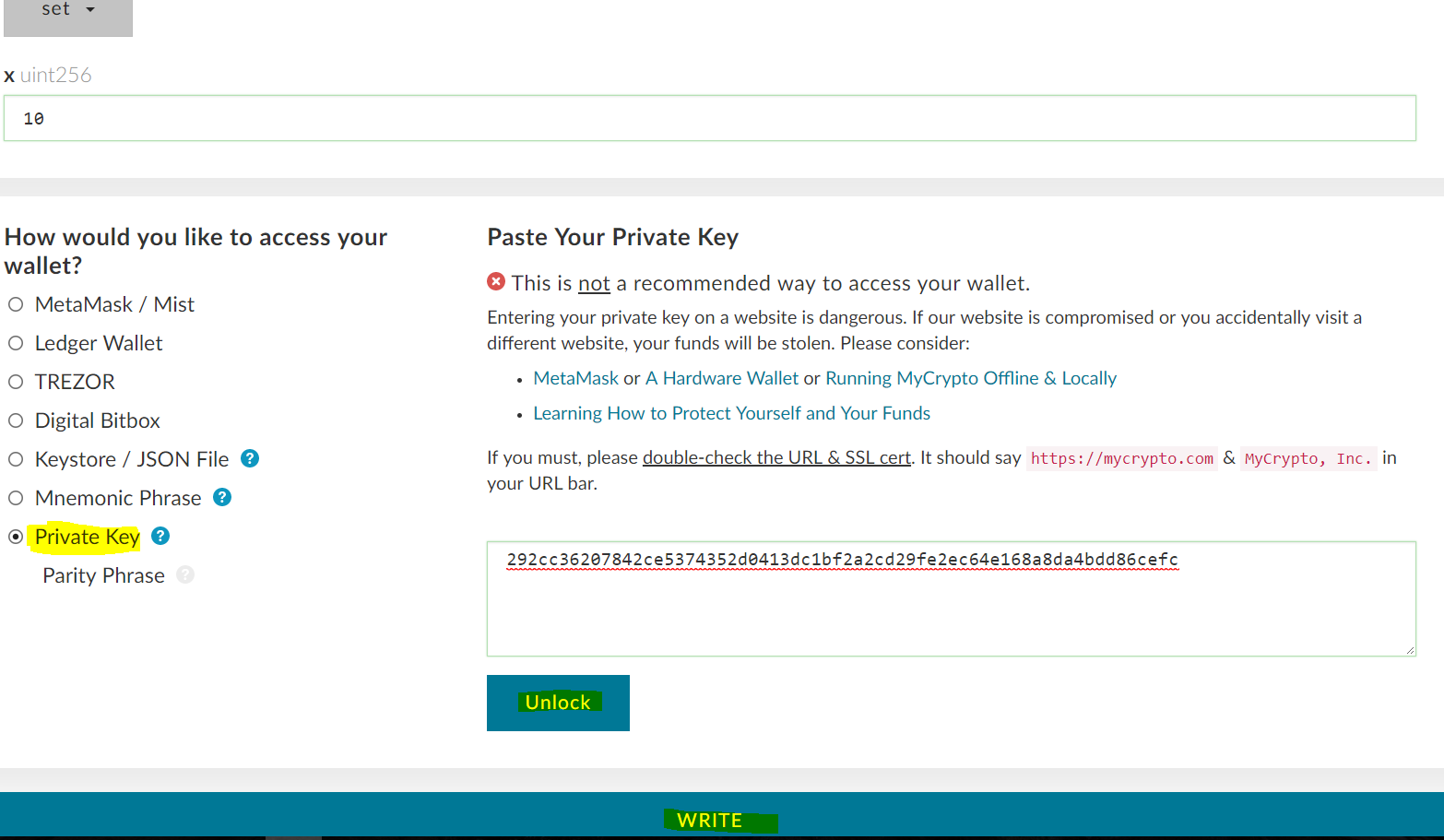
# Second, we will need to go to <https://mycrypto.com> -> Contracts, where we can add our Contract Address: 0xEe629816757ff7650c6f420edECdfFe48C5F71Ca and the ABI/JSON interface. You can easily find your ABI/JSON contract in the Ropsten explorer, if you will go to our contract 0xEe629816757ff7650c6f420edECdfFe48C5F71Ca and press on Contract Source . You will now see Contract ABI, just copy-and paste the code in the ABI/JSON interface.

# 

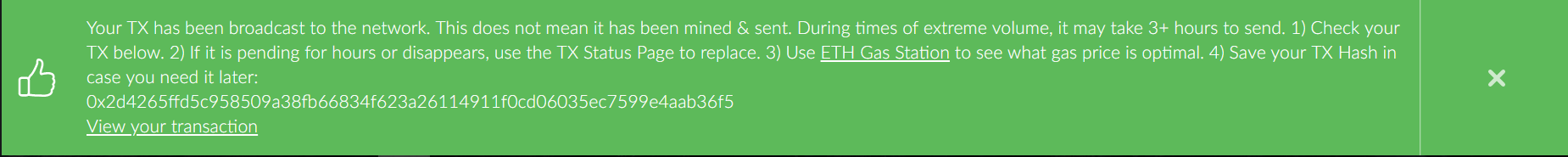
Now we`ve managed to access our contract and see the same functions like in Remix IDE. Let’s change the value to from our variable to 10, in order to see what happens in our contract when we will call function **get**.

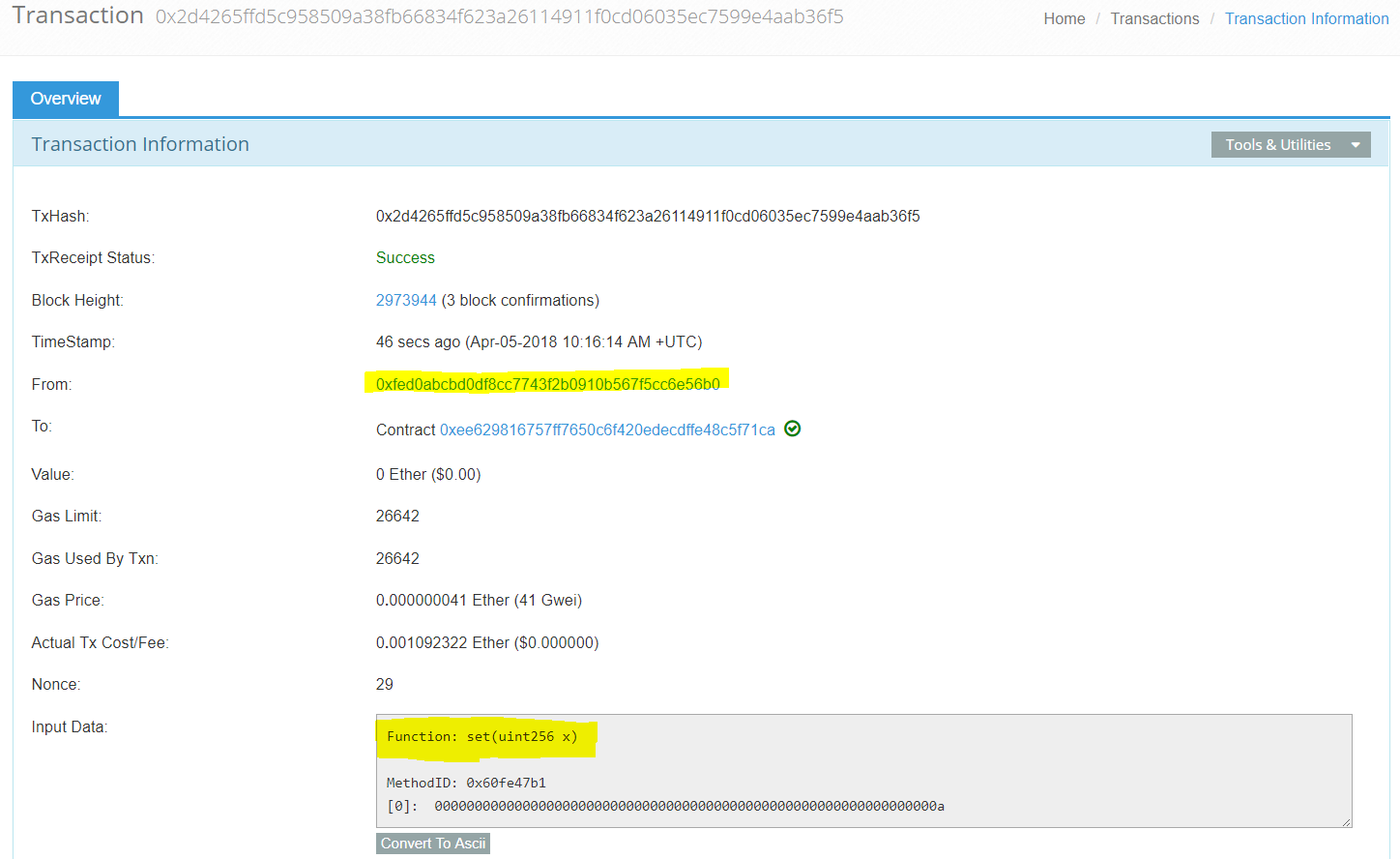
To submit the transaction in <https://mycrypto.com> , we can access and unlock our wallet in many ways, depending on the type of the walled that we have. The safest way to store your founds it is a cold storage device like Ledger Wallet or Trezor. But as we are using a test network, we don`t have to be so precautious, and we will use a private key method ( not safe to store the private key on any online device ). Here is our private key **292cc36207842ce5374352d0413dc1bf2a2cd29fe2ec64e168a8da4bdd86cefc** from Ropsten, there are 0.10 ETH right now, feel free to use them if needed.

We will **Unlock** our wallet, then **Write** (submit) the transaction:

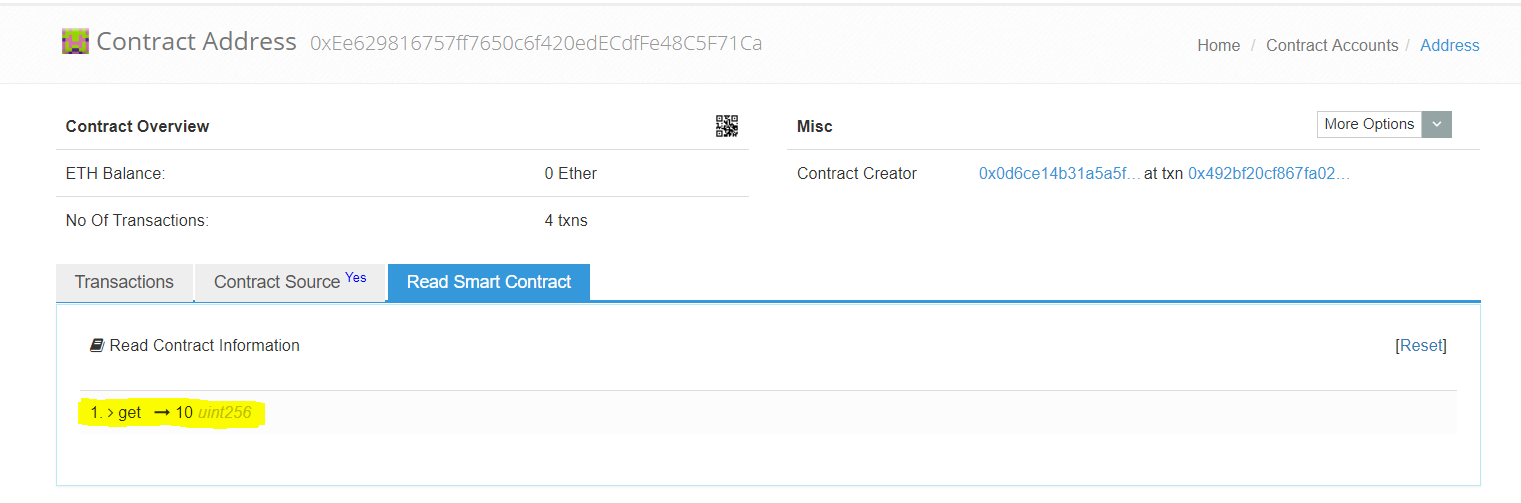


We now have generated the transaction: <https://ropsten.etherscan.io/tx/0x2d4265ffd5c958509a38fb66834f623a26114911f0cd06035ec7599e4aab36f5>. If we check it, we will see that the address [0xfed0abcbd0df8cc7743f2b0910b567f5cc6e56b0](https://ropsten.etherscan.io/address/0xfed0abcbd0df8cc7743f2b0910b567f5cc6e56b0) which called the function **set,** it’s different that the one which deployed the contract.





And if we will go back to our contract on etherscan: <https://ropsten.etherscan.io/address/0xee629816757ff7650c6f420edecdffe48c5f71ca#readContract> , we will see the value of our variable`s value is now 10.



You should now be able to deploy, verify and interact with a smart contract on the Ethereum blockchain from the two of the best interfaces available currently. Please feel free to play with the tools, use your imaginations, and contact us if you need any help.