Smart Contract Compilation

- The solidity source code is passed to the solidity compiler and the compile returns the EVM bytecode that is deployed to the Ethereum Blockchain and the contract ABI (Abstract Binary Interface);
- There are many solidity compilers available: Remix browser-based compiler, command line solc compiler that must be installed (sudo add-apt-repository ppa:ethereum/ethereum && sudo apt-get update && sudo apt-get install solc on Ubuntu), solcjs which is a JavaScript based solidity compiler and can be installed via Node.js (nmp install solc);
- ABI is list of contract's function and arguments and it's in JSON format. ABI is known at compile time;

Smart Contract Compilation

- ABI is generated from source code through compilation. **If we don't have the source code we can't generate the contract ABI** than only from the bytecode using reverse engineering;
- Anyone that wants to interact with the contract must have access to the contract ABI. ABI is basically **how you call functions** in a contract and get data back;
- Contract bytecode is public. It is saved on the Blockchain and can't be encrypted because it must be run by every Ethereum node;
- Opcodes are the human readable instructions of the program. It can be easily obtained from bytecode;
- There are tools like **Porosity** that can reverse engineer the bytecode to source code;
- Contract source code doesn't have to be public;