Overview

In this short guide weâll encrypt plaintext to ciphertext using FHE, demonstrating how simple it is to enable confidentiality in your smart contracts with Fhenix.

- Contract Example:
- pragma solidity
- ^
- 0.8
- .17
- ;
- import
- "@fhenixprotocol/contracts/FHE.sol"
- ;
- contract
- EarlyWin
- {
- uint8 _plaintext
- ;
- euint8
- public
- _cipherText
- ;
- function
- setCipherText
- (
- inEuint8 calldata _encryptedNumber
-)
- public
- {
- // convert inEuint8 type structure to euint8
- _cipherText
- =
- FHE
- .
- asEuint8
- (
- _encryptedNumber
-)
- ;
- }
- function
- setPlainText
- (
- uint8 _number
-)
- public
- {
- //set standard plaintext
- _plaintext
- =
- _number
- ;
- }
- function
- decrypt
- (
-)
- public
- view
- returns
- (
- uint8
-)
- {
- return
- FHE

- .
- decrypt
- (
- _cipherText
-)
- ;
- }
- }

Overview

First, FHE is imported directly into your contract with a single line of code. Next we establish two numbers or unsigned integers, withhowever the _cipherText number beingwill be encrypted. This means that it will not be publicly accessible from anyone other than the intended viewer. The standard _plaintext unit8 represents a number that is public for all to view. On line 11 we have a setter function that allows us to pass in a new encrypted number that will stay private. On line 21 we have a decrypt function that allows us to view the number that was written to state in the setCipherText function. As you can see there is no requirement for off-chain proof generation or storage.

import

"@fhenixprotocol/contracts/FHE.sol" ; Edit this page

Previous Deploying (WIP)