ð ï¸ Foundry

This tool will allow you to simulate development on the Fhenix network including Solidity native testing. The tool uses Foundry Cheatcodes to represent the execution of our Fhenix native precompiles. Please be aware that the operations performed by using this template only simulate the behavior of real FHE operations, and will probably not be a good representation of performance, gas, etc.

The code is pretty new, and may contain bugs or edge-cases that we have not tested, so your feedback is very important for us! If you have any issues, comments or requests please open an issue in the Fhenix Foundry TemplateRepository .

Getting Startedâ

To create a new repository using ourFhenix Foundry Template , click theUse this template button. Alternatively, install the template manually as follows:

mkdir my-project cd my-project forge init --template fhenixprotocol/fhenix-foundry-template bun install # install Solhint, Prettier, and other Node.js deps If this is your first time using Foundry, refer to theinstallation instructions for guidance.

Featuresâ

- Simulated FHE Operations: All FHE operations, including encryption, decryption, and encrypted data handling, are
- simulated to replicate their behavior in a network environment. This approach facilitates seamless development and
- testing without requiring a fully operational FHE network.
- Permissions: The template includes utilities (PermissionHelper.sol
-) for creating permissions related to FHE
- operations. These utilities enable users to test and verify that contracts correctly implement access-controlled
- actions, such as viewing balances of encrypted tokens. For more about permissions, seehere
- .

Writing Testsâ

Define a new Forge Test Contractâ

1. Start by importingTest
2. fromforge-std
3. .
4. Inherit the test contract.

Note that: Forge Std comes with a pre-instantiatedcheatcodes environment, which is accessible via the vm property. To view the logs in the terminal output, add the -vvv flag and useconsole.log .

Add support for FHE operationsâ

This template includes an example test contractFHERC20.t.sol .

For contracts utilizing FHE operations, insert FHE mock operations using theFheEnabled contract. By inheriting theFheEnabled contract in the test contract, you gain access to FHE operations. The following code demonstrates this.

import

{ FheEnabled }

from

"./util/FheHelper.sol" ;

contract

MyTestContract

is Test , FheEnabled { // Your test contract code here } During test setup,initializeFhe the FHE environment using the initializeFhe function:

function

setUp ( )

public

{ initializeFhe ( ) ; } For a complete example, including mocked encryption, decryption, sealing and permission usage, refer to the exampletests provided in the tests directory of the repository.

Permissionsâ

ThePermissionHelper contract provides utilities for managing permissions related to FHE operations. These utilities enable users to test and verify that contracts correctly implement access-controlled actions, such as viewing balances of encrypted tokens.

Consider using the following code as an example for aPermissionHelper contract in a test contract:

import

{ Test }

from

"forge-std/src/Test.sol" ;

import

{ ContractWeAreTesting }

from

"./src/ContractWeAreTesting.sol" ; import

{ PermissionHelper }

from

"./util/PermissionHelper.sol" ;

contract

MyContract

is Test { ContractWeAreTesting private contractToTest ; PermissionHelper private permitHelper ;

function

setUp ( )

public

{ // The contract we are testing must be deployed first contractToTest =

new

ContractWeAreTesting ( ) ;

// The PermissionHelper contract must be deployed with the address of the contract we are testing // otherwise the permission generated will not match the address of the contract being tested permitHelper =

new

PermissionHelper (address (contractToTest)) ; }

function

testOnlyOwnerCanViewBalance ( )

public

{ // Owner key and address uint256 ownerPrivateKey =

0xA11CE ; address owner = vm . addr (ownerPrivateKey) ;

// Generate a permission for the owner using the permitHelper and the private key Permission memory permission = permitHelper . generatePermission (ownerPrivateKey) ;

// Call function with permission uint256 result = permissions . someFunctionWithOnlyPermitted (owner , permission) ; } }

Note that thePermissionHelper contract is initialized only after we know the address of the contract being tested. The reason is that the permission generated by thePermissionHelper contract is tied to the address of the contract that is being tested.

Differences from Real FHE Operationsâ

FHE operations in this template simulate the behavior of a real FHE network. Instead of processing encrypted data, operations are performed on plaintext data, which enables seamless development and testing without the need for a fully operational FHE network. However, there are important differences between these mocked FHE operations and actual FHE operations:

- Gas Costs â Gas costs associated with the mocked FHE operations do not accurately reflect those of real FHE
- operations. Instead, they are based on gas costs of equivalent non-FHE operations.
- Security Zones â In this mocked environment, security zones are not enforced. Thus, any user can perform operations
- between ciphertexts, which would otherwise fail in a real FHE setting.
- Ciphertext Access â The mocked FHE operations do not enforce access control restrictions on ciphertexts, which allows
- any user to access any mocked "ciphertext." On a real network, such operations could fail.
- Decrypts during Gas Estimations: When performing a decrypt (or other data revealing operations) during gas estimation
- on the Helium testnet or Localfhenix, the operation returns a default value, as the gas estimation process does not
- have access to the precise decrypted data. This can cause the transaction to fail at this stage, if the decrypted data
- is used in a way that would trigger a transaction revert (e.g., when a require statement depends on it).
- Security â The security provided by the mocked FHE operations does not represent the high level of security offered by
- real FHE operations. The mocked operations do not involve actual encryption or decryption.
- Performance â The performance of mocked FHE operations is not indicative of the real FHE operation speed. Mocked
- operations will be significantly faster due to their simplified nature. Edit this page

Previous âï¸ Gitpod Next ð· Hardhat