Converting Between Types

As a user ofFHE.sol , you'll often need to convert between various encrypted types or from plaintext to encrypted form within your contracts. This documentation illustrates how you can leverage the type conversion functions provided inFHE.sol to manipulate encrypted data effectively.

Using Conversion Functions

1. Converting Encrypted to Other Encrypted Types:

Suppose you have a voting contract and you want to convert an encrypted boolean vote to an encrypted integer before tallying.

- Contract Example:
- contract
- Voting
- {
- function
- convertVote
- (
- ebool encryptedVote
-)
- public
- {
- // Convert the encrypted boolean vote to an encrypted 32-bit integer
- euint32 encryptedVoteInt
- =
- encryptedVote
- .
- toU32
- (
-)
- ;
- // Further processing with encryptedVoteInt
- }

- }

- Converting from Plaintext to Encrypted Type:

If you're initializing an encrypted counter in a contract, you might start with a plaintext value that needs to be encrypted.

- Contract Example:
- contract
- Counter
- {
- euint32
- private
- encryptedCount
- ;
- function
- initializeCount
- (
- uint256 initialCount
-)
- public
- {
- // Convert a plaintext count to an encrypted count
- encryptedCount
- =
- FHE
- .
- asEuint32
- (
- initialCount
-)
- ;
- }

- }

Note that when converting from plaintext to encrypted, the value is still exposed in plaintext to the contract and on the public blockchain. This pattern should only be used when the plaintext data is not sensitive and can be exposed publicly.

Conversion Functions

Tips for Users

- Understand the Types:
- Know the types you are working with and the implications of converting between them. Ensure the conversion is logical and secure.
- Monitor Gas Usage:
- Be aware of the gas costs associated with type conversions, especially if they occur within functions that are called frequently.
- Test Thoroughly:
- Always test your contracts with various scenarios to ensure that type conversions are behaving as expected.

As a user ofFHE.sol , understanding and utilizing type conversions is essential for manipulating encrypted data within your smart contracts. By following the examples and best practices provided, you can ensure your contracts are efficient, secure, and functional. Remember to test thoroughly and consider the implications of each conversion to maintain the integrity and reliability of your contract's operations. Edit this page

Previous Next Permissions ⚕ Types and Operations