EthGlobal Brussels - 2024

Cavoletto - Andrea Ciceri & Ivan Sala

Who are we?

Andrea Ciceri
Tecnophilosopher

Ivan Sala
AI & Blockchain enjoyer

Also known for participating at EthGlobal Lisbon23, Paris23, London24 and EthRome23 - We just love to explore new tech

Our Project - Cavoletto 🪦

"Cavoletto" is an Italian expression used when something bad happens but it is also Brussels sprouts

The DApp allows users to deploy smart contracts on the Fhenix testnet and specify a list of heirs and which percentages of which token they will inherit.

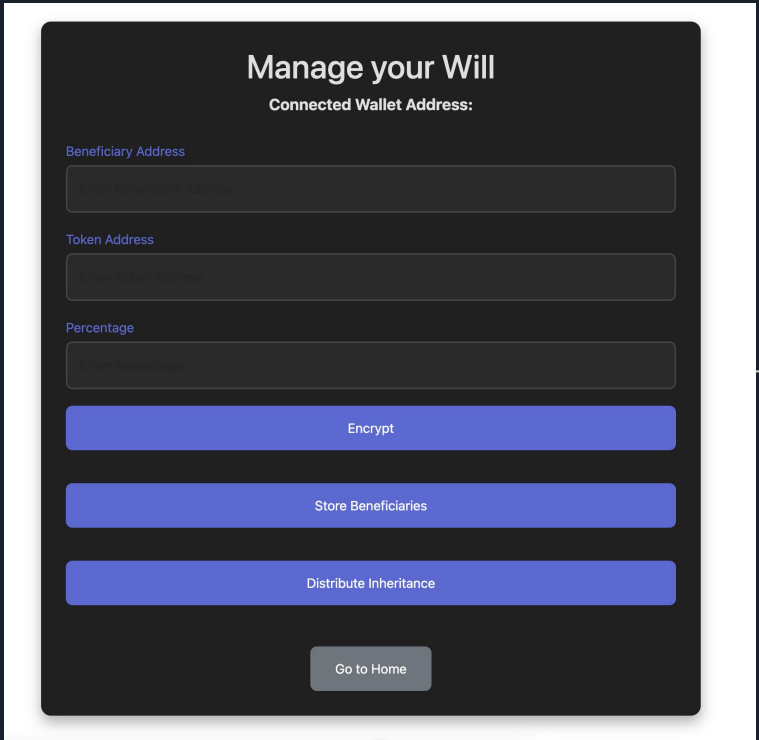
Tokens can be both TEth, generic ERC-20 tokens and NFTs.

All the information (hiers, tokens and percentages) will be stored on chain encrytped, so they will be secret as long as the testator is alive.

When the testator/deployer will declared dead whoever will be able to trigger the contract and distribute the inheritance.

As possible strategies for the proof of death we considered TLSNotary or simply a mechanism that automatically marks the testator as dead after a certain amount of time if he/she doesn't recurrently call a contract's method.

What we did



Vue.js

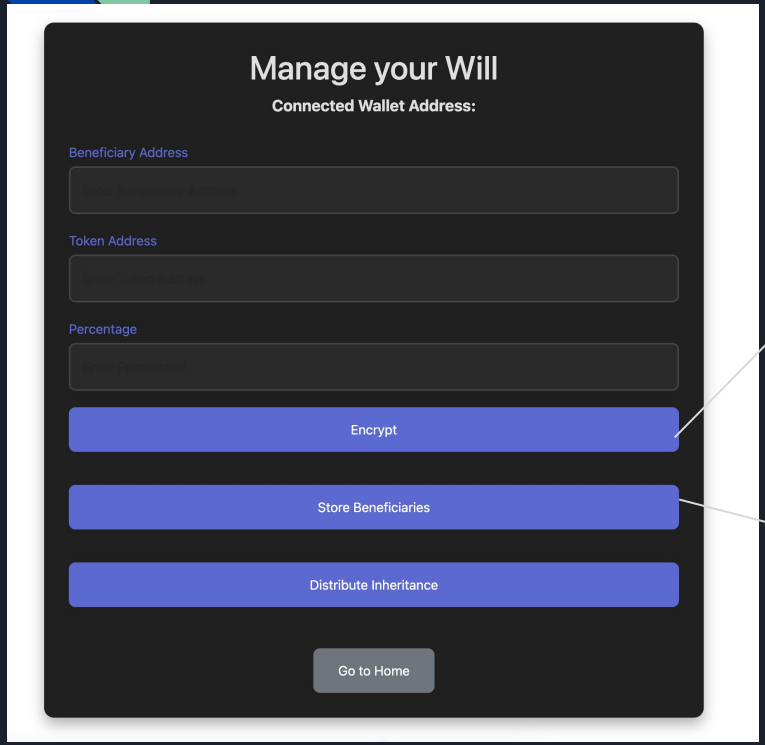
SOLIDITY

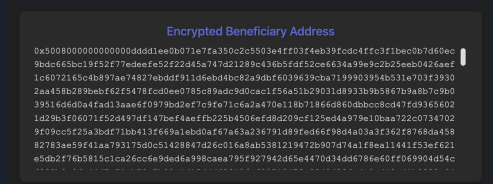
InheritanceManager

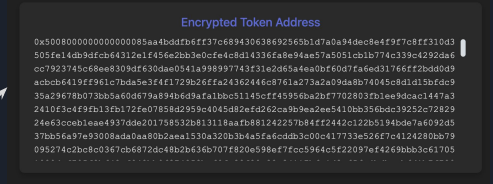
- address owner
- address foo
- EncryptedBeneficiary encryptedBeneficiaries
- Beneficiary beneficiaries

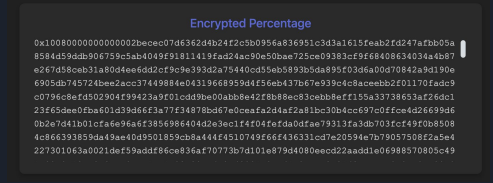
- __constructor__()
- storeBeneficiaries()
- _storeBeneficiary()
- distributeInheritance()

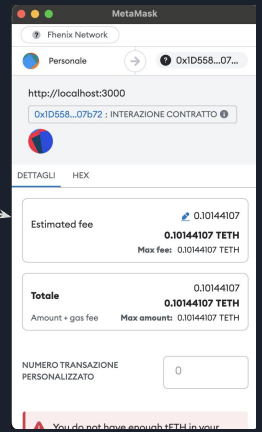
What we did


Manage your Will
Connected Wallet Address:
Beneficiary Address
Enter Beneficiary Address
Token Address
Enter Token Address
Percentage
Enter Percentage
Encrypt
Store Beneficiaries
Distribute Inheritance
Go to Home

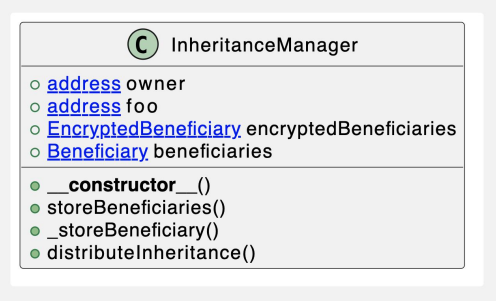

Encrypted Beneficiary Address
0x500800000000000dddd1ee0b071e7fa350c2c5503e4ff03f4eb39fcdc4ffc3f1bec0b7d60ec
9bdc665bc19f52f77edeefe52f22d45a747d21289c436b5fdf52ce6634a99e9c2b25eeb0426aef
1c6072165c4b897ae74827ebddf911d6ebd4bc82a9dbf6039639cba7199903954b531e703f3930
2aa458b289bebf62f5478fcd0ee0785c89adc9d0cac1f56a51b29031d8933b9b5867b9a8b7c9b0
39516d6d0a4fad13aae6f0979bd2ef7c9fe71c6a2a470e118b71866d860dbbcc8cd47fd9365602
1d29b3f06071f52d497df147bef4aeffb225b4506efd8d209cf125ed4a979e10baa722c0734702
9f09cc5f25a3bdf71bb413f669a1ebd0af67a63a236791d89fed66f98d4a03a3f362f8768da458
82783ae59f41aa793175d0c51428847d26c016a8ab5381219472b907d74a1f8ea1141f53ef621
e5db2f76b5815c1ca26cc6e9ded6a998caea795f927942d65e4470d34dd6786e60ff069904d54c


Encrypted Token Address
0x500800000000000085aa4bddfb6ff37c68943063869256 5b1d7a0a94dec8e4f9f7c8ff310d3
505fe14db9dfcb6431 2e1f456e2bb3e0cfe4c8d14336fa8e94ae57a5051cb1b774c339c4292da6
cc7923745c68ee8309df630dae0541a998997743f31e2d65a4ea0bf60d7fa6ed31766ff2bdd0d9
acbcb6419ff961c7bda5e3f4f1729b26ffa24362446c8761a273a2a09da8b74045c8d1d15bfdc9
35a29678b073bb5a60d679a894b6d9afa1bbc51145cff45956ba2bf7702803fb1ee9dcac1447a3
2410f3c4f9fb13fb172fe07858d2959c4045d82efd262ca9b9ea2ee5410bb356bdc39252c72829
24e63cceb1eae4937dde201758532b813118aafb881242257b84ff2442c122b5194bde7a6092d5
37bb56a97e93008ada0aa80b2aea1530a320b3b4a5fa6cddb3c00c417733e526f7c4124280bb79
095274c2bc8c0367cb6872dc48b2b636b707f820e598ef7fcc5964c5f22097ef4269bbb3c61705


Encrypted Percentage
0x100800000000000002becec07d6362d4b24f2c5b0956a836951c3d3a1615feab2fd247afbb05a
8584d59ddb906759c5ab4049f91811419fad24ac90e50bae725ce09383cf9f68408634034a4b87
e267d58ceb31a80d4ee6dd2cf9c9e393d2a75440cd55eb5893b5da895f03d6a00d70842a9d190e
6905db745724bee2acc37449884e04319668959d4f56eb437b67e939c4c8aceebb2f01170fadc9
c0796c8efd502904f99423a9f01cdd9be00abb8e42f8b88ec83cebb8eff155a33738653af26dc1
23f65dee0fba601d39d66f3a77f34878bd67e0ceafa2d4af2a81bc30b4cc697c0ffce4d26699d6
0b2e7d41b01cfa6e96a6f3856986404d2e3ec1f4f04fefda0dfae79313fa3db703fcf49f0b8508
4c866393859da49ae40d9501859cb8a444f4510749f66f436331cd7e20594e7b79057508f2a5e4
227301063a0021def59addf86ce836af70773b7d101e879d4080eecd22aadd1e06988570805c49


MetaMask
Fhenix Network
Personale
0x1D558...07...
http://localhost:3000
0x1D558...07b72 : INTERAZIONE CONTRATTO
DETTAGLI
HEX
Estimated fee
0.10144107
0.10144107 TETH
Max fee: 0.10144107 TETH
Totale
0.10144107
0.10144107 TETH
Amount + gas fee
Max amount: 0.10144107 TETH
NUMERO TRANSAZIONE PERSONALIZZATO
0

What we did



Deployed Smart contract:
0x1D55838a9EC169488D360783D65e6CD985007b72

https://explorer.helium.fhenix.zone/address/0x1D55838a9EC169488D360783D65e6CD985007b72

What we did

```
function storeBeneficiaries(
    inEaddress[] calldata encryptedAddresses,
    inEaddress[] calldata encryptedTokenAddresses,
    inEuint32[] calldata encryptedPercentages
) public {
    require(msg.sender == owner, "Only owner can store beneficiaries");
    require(
        encryptedAddresses.length == encryptedTokenAddresses.length &&
        encryptedTokenAddresses.length == encryptedPercentages.length,
        "Arrays must have the same length"
    );

    for (uint256 i = 0; i < encryptedAddresses.length; i++) {
        _storeBeneficiary(encryptedAddresses[i], encryptedTokenAddresses[i], encryptedPercentages[i]);
    }
}
```

What we did

```
function _storeBeneficiary(
    inEaddress calldata encryptedAddress,
    inEaddress calldata encryptedTokenAddress,
    inEuint32 calldata encryptedPercentage
    ) public {
        encryptedBeneficiaries.push(EncryptedBeneficiary({
            encryptedAddress: FHE.asEaddress(encryptedAddress),
            encryptedTokenAddress: FHE.asEaddress(encryptedTokenAddress),
            encryptedPercentage: FHE.asEuint32(encryptedPercentage)
        }));
}
```

What we did

```
function distributeInheritance() public {      You, 9 ore fa • maybe a first commit with the smc
    require(encryptedBeneficiaries.length > 0, "No beneficiaries stored");

    for (uint256 i = 0; i < encryptedBeneficiaries.length; i++) {
        eaddress encryptedBeneficiaryAddress = encryptedBeneficiaries[i].encryptedAddress;
        eaddress encryptedTokenAddress = encryptedBeneficiaries[i].encryptedTokenAddress;
        euint32 encryptedPercentage = encryptedBeneficiaries[i].encryptedPercentage;

        // Decrypt beneficiary address and token address
        address beneficiaryAddress = FHE.decrypt(encryptedBeneficiaryAddress);
        address tokenAddress = FHE.decrypt(encryptedTokenAddress);

        // Calculate the amount to transfer based on encrypted percentage
        uint256 tokenBalance = IERC20(tokenAddress).balanceOf(owner);
        euint32 encryptedTokenBalance = FHE.asEuint32(tokenBalance);
        euint32 encryptedAmount = (encryptedTokenBalance * encryptedPercentage) / FHE.asEuint32(100);
        uint32 decryptedValue = FHE.decrypt(encryptedAmount);

        Transfer the amount from the owner to the beneficiary
        IERC20(tokenAddress).transferFrom(owner, beneficiaryAddress, decryptedValue);
        IERC20(tokenAddress).transferFrom(owner, beneficiaryAddress, 1);
    }
```

What could (must!) be done next

What Could (Must!) Be Done Next

- Complete the functionalities on the frontend (dApp).
- Introduce on-chain verification of certain death.

Some consideration on the Tech

Overall Experience

- The technology has been generally satisfactory (more than other FHE provider!).
- It proved to be useful and functional for our needs (more datatype than competitor).

Key Insights

- More examples are definitely necessary.
- A focus on End-to-End (E2E) scenarios is particularly important.

New Connections

- Made new friends and expanded our network!

Thanks for your attention!