#1 – Blockchain Will Lodging and Checking

Problem statement:

There is no central repository to deposit wills. A hardcopy is kept by the person leaving the will (testator) and the lawyer who drafted the will.

In event of sudden death or illness (e.g dementia) of the testator, the next of kin may not be aware of the existence of the hardcopy will. Or the hardcopy could have been replaced by a newer version which the next of kin is unaware of.

The lawyer may also have passed on or the law firm may have closed or merged, making tracking the will extremely challenging

Solution

1. Create a blockchain to register a will has been written and validated. Once the testator’s will is lodged, it is immutability. A new block will be added, hence invalidating the old version, should testator create a new version.
2. Lawyer will sign using his private keys to confirm the will is valid
3. Similarly, witness or witnesses can sign to confirm will is lodged
4. The hardcopy or softcopy of the will can include the transaction hash of the blockchain.

Project Scope:

Front end

1. UI for testator to submit will. Form to include, location of hardcopy or softcopy will, particulars of lawyer, witness, public keys of lawyer and witness. Up to 2 witnesses can be added
2. Testator will pay for the service to the lawyer and witnesses using Ether
3. UI for lawyer to sign using his keys
4. UI for witnesses to sign
5. UI for next of kin to upload death certificate.

Backend

1. Create Testator object. To include testator name, IC, address, sex
2. Create Will object. Events and methods include
   1. checking if will exists against testator details
   2. name of lawyer,
   3. name of witness,
   4. location of will
   5. others - TBC

pragma solidity ^0.6.0;

contract WillContract {

enum lodgeStatus { submitted, lawyersigned, witnessok, completed }

struct will {

address testatorAddress;

address lawyerAddress; // lawyer's account address

address witnessAddress; // witness account address

uint8 willCount; // number of wills submitted by a particular testator

lodgeStatus willStatus;

}

event submitted (address testator);

event replacementWillSubmitted (address testatorAdd);

event laywercompleted (address testator);

event witnesscompleted (address testator);

event willcompleted (address testator);

mapping(address => will) public wills;

//function to create a new will, and add to 'wills' map. requires at least 0.01ETH to create

function add() public payable returns (uint8) {

//check if will exists

if (wills[tx.origin].willCount > 0 ) {

// set the lawyer and witness address to address (0)

wills[tx.origin].lawyerAddress = address(0);

wills[tx.origin].witnessAddress = address(0);

wills[tx.origin].willStatus = lodgeStatus.submitted;

emit replacementWillSubmitted(tx.origin);

} // end of if

else

{

//new will object

will memory newWill = will(

tx.origin, // testator (owner)

address(0), // lawyer has not signed, so use 0 as address

address(0), // witness has not signed, so use 0 as address

0, // number of wills (willcount) is initalised to 0

lodgeStatus.submitted

);

wills[tx.origin] = newWill;

//commit to state variable

emit submitted (tx.origin);

} // end of else

return wills[tx.origin].willCount++;

}

//lawyer sign on a will

function lawyerSign (address testatorAdd) public {

wills[testatorAdd].willStatus = lodgeStatus.lawyersigned; //set status to lawyersigned

wills[testatorAdd].lawyerAddress = msg.sender; //set the the laywer address to the laywer's key address

emit laywercompleted(testatorAdd);

}

function witnessSign(address testatorAdd) public {

wills[testatorAdd].willStatus = lodgeStatus.witnessok ; //set status to witnessok

wills[testatorAdd].witnessAddress = msg.sender;

emit witnesscompleted (testatorAdd); //emit witness completed signing

}

//check completeness of signing. If completed, change status to completed

function checkcompleteness(address testatorAdd) public {

if ( ( wills[testatorAdd].lawyerAddress != address(0) ) && ( wills[testatorAdd].witnessAddress != address(0) ) ) {

wills[testatorAdd].willStatus = lodgeStatus.completed ;

emit willcompleted(testatorAdd);

} // end if statement

}

}