Lecture 24 - Key Custodian

Last time we looked at a app that kept the keys in MetaMask (the hello world) with the user holding the private keys.

This has a number of problems that have to be dealt with.

1. System failure - did the person make backups of the keys
2. Loss of password to the keys - they are encrypted. It is AES 128 encryption!
3. Theft of keys - poor security on individuals computers.
4. Inheritance - what happens to crypto when a person dies.
5. Know Your Customer - Money Laundering - how do you “know” your customer if all you have is an account number.
6. Limitations on what you can do - Document System, Disintermediation Tokens etc.

Your bank uses keys like this - but you don’t know about it.

The alternative is to use a username/password system that we are all familiar with and keep the keys for the user.

You loose the “distributed” nature of the blockchain - but gain all the features of a client/server.

An example. Document Attestation System.

Let’s say we want to build a document system for a law office. This is a system where the lawyers can take digital evidence and upload it to a system that tracks it with a description and a case number - and signs - and attests to the authenticity of the evidence.

So... We have to have some metadata that we track. This is the case number and case description and the description of the document. We have to have the document that we upload and save. Then we hash the document and save the hash on chain so that we can prove date/time/name/content of the document in a court of law.

Let’s draw a model of this...

Server to save the metadata

```
/api/doc/law_cases
```

1. GET - retrieves a list cases
2. POST - inserts a new case
3. PUT - updates a case - saving a history of the case.
4. DELETE - marks a case as deleted

    ```
    /api/doc/law_cases/ID
    ```

5. GET - Returns a single case.

    ```
    /api/doc/law_doument?case_id=ID
    ```

6. GET - retrieves a list of metadata items for a case.
7. POST - inserts a new metadata item on a case.
8. PUT - updates a metadata item on a case - saving a history of the item.
9. DELETE - marks an item deleted.

```
/api/doc/law_doument?case_id=ID&item_id=ID2
```

10. GET - Returns a single item from a single case.

```
/uplaod
```

Takes a file and uploads it and returns an FILE_ID and a HASH_FILE for the file. The file is stored on the server. In a real system we would need to encrypt the file and copy it to something like Amazon S3 so that we have a "permanent" copy of it.

We create a "key" for every case. We "sign" every document with the key. We keep a "signature" for the documents on chain.

```
/fetch_file?file_id=fid
```

gets a file back based on id.

```
/fetch_file?hash_file=hash
```

gets a file back based on its hash.

In PostgreSQL the tables for this:

```
 1:
 2: drop table if exists law_keys ;
 3: drop table if exists law_doument_hist ;
 4: drop table if exists law_doument ;
 5: drop table if exists law_case_hist ;
 6: drop table if exists law_case ;
 7:
 8: create table if not exists law_case (
 9:     case_id         serial primary key not null,
10:     case_name         text not null,
11:     case_desc         text not null,
12:     deleted         varchar(1) not null default 'n',
13:     updated         timestamp,
14:     created         timestamp default current_timestamp not null
15: );
16:
17: create table if not exists law_case_hist (
18:     case__hist_id     serial primary key not null,
19:     activity         text not null,
20:     case_id         int not null,
21:     case_name         text not null,
22:     case_desc         text not null,
```

```
23:     deleted         varchar(1) not null,
24:     updated         timestamp,
25:     created         timestamp
26: );
27:
28: create table if not exists law_doument (
29:     document_id     serial primary key not null,
30:     case_id         int references law_case(case_id),
31:     document_name     text,
32:     document_desc     text,
33:     hash             text,
34:     file_id         text,
35:     file_signature    text,
36:     tx                text,
37:     deleted         varchar(1) not null default 'n',
38:     updated         timestamp,
39:     created         timestamp default current_timestamp not null
40: );
41:
42: create table if not exists law_doument_hist (
43:     document_hist_id serial primary key not null,
44:     activity         text not null,
45:     document_id     int not null,
46:     case_id         int references law_case(case_id),
47:     document_name     text,
48:     document_desc     text,
49:     hash             text,
50:     file_id         text,
51:     file_signature    text,
52:     tx                text,
53:     deleted         varchar(1) not null,
54:     updated         timestamp,
55:     created         timestamp
56: );
57:
58: create table if not exists law_keys (
59:     keys_id serial primary key not null,
60:     case_id int references law_case(case_id),
61:     key_enc text not null,
62:     deleted varchar(1) not null default 'n'
63: );
```