Goal

Develop a solution that will allow owners of documents to confirm their ownership, proof of existence, protect rights, create contracts between the unlimited amount of parties and have information of the latest signed document on the blockchain. The development of backend and frontend, development and audit of Ethereum smart contracts, blockchain consulting and business analysis.

Problems

The Parties could verify the signature authenticity and assure the Parties of the immutability of the signed document. It allows eliminate the intermediaries and allows the users using affordable legal services.

- Notaries or other intermediaries needed to save and verify documents.
- Storing information about signing for infinite time without relying on third party services.
- Automatization of processes after signing a document by all participants of a contract
- Information about signers and signatures cannot be reachable from all over the world in case of paper documents.
- No existing systems that allows to establish a process of creation of a new version of e-documents and have transparent view what is the last approved version by all parties.
- Prove certain data exists at a certain moment of time (proof of existence)
- Assure documents integrity.
- No existing systems that allow to establish a process of creation of a new version of e-documents and have a transparent view of what is the last approved version by all parties.

Information about signers and signatures cannot be reachable from all over the world in case of paper documents

Solution:

Using public blockchains allows store information for an infinite time by paying one-time fee. The immutability is the strength of blockchain that gives it high robustness. What that means is, they are designed to be only ever created, and not edited or deleted. You can think of this as an ever-growing journal that is constantly having new pages added to it over time and which is stored across multiple locations.

To make any change on the blockchain you must create a transaction. All transactions are combined into blocks. To add a block with transactions to the blockchain it must be mined before. All blocks on the blockchain are linked (each block contains the hash of the previous block) and have timestamps and block numbers. In such way, you always can trace all information in a chronological way.

Storing a big amount of data like documents is not suitable for storing on the blockchain. The most efficient way is making a checksum (hash) of a document you want to sing, store it (hash) on a blockchain and with the help of smart contracts make interactions with it. All interactions with the smart contracts (signing, rejecting, confirming versions etc) are linked to related document’s hash.

It is intended to construct all smart contracts without upgradeable functionality to avoid any substitutions and achieve transparent, trustful platform. So, once the smart contracts are deployed their behavior can’t be changed.

So as data is immutable and it is stored in a public distributed way you always can make requests to receive relevant and trustful information.

Prove certain data exists at a certain moment of time (proof of existence)

Problem:

You have been developing some idea for several years with your team. One participant of your team decided to stole it and do it by himself. You should prove that it was yours.

Solution:

A service that securely stores an online distributed proof of existence for any file or document. Such documents are not stored in our database and even on any blockchain so you don’t have to worry about that private information will be in public access and somebody can be “inspired” by it. What is stored is a cryptographic digest of the file (hash), linked to the timestamp in which you submitted the document. In this way, you can certify that the data already existed at that time.

The key advantages are privacy and getting a decentralized proof which can’t be erased, modified or replaced by anyone (third parties or governments etc). Also there is no way to insert a transaction with a document’s hash retroactively cause of blockchain specification. Your document’s existence is permanently validated by the blockchain even if a platform is

compromised or down, so you don’t depend on or need to trust any central authority. All previous data timestamping solutions lack this freedom.

Assure documents integrity

Solution:

Having a proof of existence for a file and later re-upload it to verify integrity, the system will only find it if it is completely the same document. The slightest change, and it will be recognized as a different document, giving you the security that certified stored data can’t be modified.

So cryptographics algorithms in pair with blockchain provide documents integrity.

No existing systems that allow to establish a process of creation of a new version of e-documents and have a transparent view of what is the last approved version by all parties**

Problem:

Everything is changing so contracts between partners may also be appended, detailed or updated. To initiate updating a contract offline, the parties must visit a notary to invalidate a previous contract and sign a new one or append a existing one. It is sometimes difficult to do cause the parties could be in different parts of the world and it is much easier and cheaper to do this online.

Solution:

Using public blockchains with help of smart contracts allows storing information for infinite time in a secure and structured way that guarantee integrity and existence of the data.

Signing process

The signing process established with the help of Document Smart Contract. Each instance of the contract belongs to one deal in the real world. The Document Smart Contract is fully responsible for signing process, linking corresponding data to a document’s hash and doesn’t allow to sign, store and create new versions of a document by a participant if he doesn’t have verified identity and was not accepted by all participants that have already signed the document.

There are such roles during the signing process:

- Consumer
- one of the participants who takes part in a signing a document.
- Service Provider
- an entity which provide service and infrastructure for signing documents.
- Certificate authority
- an entity which is accredited by Service Provider to verify identities and provide proofs about them.

[

image

1600×1109 32.3 KB

](https://ethresear.ch/uploads/default/original/2X/f/f662970c3113ba7b3fdb77465bb21a3ffc8ad5f8.png)

The signing process

Identity verifying process

The main challenges we’ve faced with: The Parties could verify the signature authenticity and assure the Parties of the immutability of the latest version of the signed document.

Each participant that was invited to sign the document must verify his identity in one of the certificate authorities that were accredited by service that provides the ability to sign documents (Service Provider).

The process of identity verifying established with a proposed standard for blockchain-based identity ERC 725. ERC 725 describes proxy smart contracts that can be controlled by multiple keys and other smart contracts. ERC 735 is an associated standard to add and remove claims to an ERC 725 identity smart contract. These identity smart contracts can describe humans, organizations, groups, objects and machines.

To verify an identity Certificate authority can work in tandem with any public eID services like NemId, BankId, Signicat etc or

can be eID itself if it supports protocol ERC725. After successful verification, Certificate authority receives the user's identity information and stores this information in an encrypted manner on own servers. Any private data is not stored on the blockchain. The only hash of identity data signed by authority's key is stored as a claim in Consumer's identity contract. This hash is used to link user's private information which stored on centralized encrypted storage and identity on the blockchain. Users pass verification only once, so they can sign any amount of documents after successful verification. In case when the user deleted the claim from his identity with help of blockchain technology it is easy to trace all old information and find the claim value to find an identity on the moment when a document was signing.

Summarizing the information above, in a case of judicial proceedings, a court can request parties identities' fields in accordance with the formalities provided for by the law.

User's identity information could be retrieved by request:

- Birthdate (1991-08-21)
- Gender (M)
- Identity document number (SPECI2019)
- Nationality (Ukrainian)
- Given name (Stepan Yablochko)
- Family name (Hrushkovych)

Verification

To suggest a new version of an existing document, one of the owners of that document must have verified identity and should initiate the creation of a new version of a document calling a method in the Document Smart Contract which responsible for starting this process. The creator of a document is automatically the first owner so he has the ability to invite new participants to sign that document. If there are more than one owner of a document and there is a need to invite another participant firstly all the existing owners should accept the new signer by signing a corresponding transaction. After the initiation, all existing signers will receive notifications about it and should sing the version to make it legitime or reject if they don't agree with something. All new versions are in pending status and are not ligitime until all owners sing it.

There are three possible statuses:

- Pending
- added document/version but not all participants are signed it.
- Approved
- all participants signed the document / new version.
- Rejected
- one of the participants rejected the version.

After signing a document new participants also become owners and can find this document in the list next to their other documents. After signing a document or specific version of a document, participants can't cancel or modify their decision.

All old signed versions of a document automatically become expired and only the newest version is the knowledge base. Despite of this users have the ability to take a look at them and get all interesting details from previous deals in any time.

To conclude, the hash of the document managed by smart contracts so nobody can delete, modify or replace it and with the help of cryptographic algorithms it is easy to prove that provided for verification document file is exactly the same document which is stored on blockchain and as the smart contracts responsible for linking all data to the specific document's hash it is easy to retrieve these data with predefined methods in the smart contracts.

Technical specifications

Solution: Ethereum based DApp.

Technologies: RoR, Web3, VueJS, Solidity.

Users of the platform:

B2C users who want to create the agreements and sign digitally via the trustworthy system on behalf of the verified user (using blockchain for data immutability)

The document is certified via storing its SHA256 digest on a public blockchain in a way that smart contracts defined.

A smart contract is a computer protocol intended to digitally facilitate, verify, or enforce the negotiation or performance of a contract. Smart contracts allow the performance of credible transactions without third parties. These transactions are trackable and irreversible.

Many kinds of contractual clauses may be made partially or fully self-executing, self-enforcing, or both using smart contracts. The aim of smart contracts is to provide security and to reduce transaction costs associated with contracting.

To manually confirm the document's existence at the timestamped time or retrieve any information that mentioned above, they should just follow these steps:

- calculate the document's SHA256 digest
- with the help of the Coordinator Smart Contract, resolve the address of a Document Smart Contract providing the generated hash during step 1
- having the address of the Document Smart Contract and using ABI make a request to a blockchain node to retrieve desirable info. (link to smart contracts and supported methods)
- returned information by any of the contract's method proves linking this information to the input parameters in rules specified by method's name.

Smart Contracts structure

Coordinator Smart Contract Specification

A hub smart contract responsible for:

- creating a Document Smart Contract using hash and title of the document
- resolving address of a Document Smart Contracts by the hash of a document
- storing the list of user's documents
- register for all Document Smart Contracts
- preventing storing not unique document on the blockchain

Document Smart Contract Specification

A smart contract responsible for:

- signing a document
- storing all versions of a document
- storing all signers and related data
- storing timestamps
- storing details about pending, rejected, approved and old versions of a document

Each document version is a structure with the next fields:

- version Id
- hash of the original pdf document
- name of the document
- when it was added to the blockchain
- who added it to the blockchain
- immutable list of the addresses who signed the version of a document and the time when it was done

Data that can be retrieved with help of smart contracts:

- list of documents that belong to a user's address
- actual, approved, rejected, pending and old versions of a document
- title of a specific version of a document
- signers and dates when it was done of a specific version of a document

- timestamps
- creator of the specific version of a document
- time when a specific version of a document was finally signed

Identity Verification flow overview:

After compiling the contracts, our demonstration takes the following steps.

1. Certificate authority deploys its own identity contract.
2. Certificate authority adds a CLAIM key to its identity contract.
3. Consumer deploys their identity contract.
4. After Consumer successfully undergoes identity verification, Certificate authority signs a identity claim for Consumer.
5. Consumer adds Certificate authority's signed claim to their identity contract.
6. Service Provider deploys its contracts for signing documents (Coordinator Smart Contract…).
7. Consumer participates in Service Provider's storing and signing documents through their identity contract to Service Provider's contracts.
8. Service Provider's contract confirms that the Consumer's identity contract contains a claim by Certificate authority before doing any changes.