A Simple Blockchain Application for Data Verification.

Learn how to code a viable application for Ethereum in a handful of lines.

Alberto Cuesta Cañada

Follow

HackerNoon.com

--

Listen

Share

All the great things are simple, and many can be expressed in a single word: freedom, justice, honor, duty, mercy, hope. — Winston Churchill

Introduction

In this article I'm going to describe the simplest blockchain application that I know, with code examples. This is useful for those starting in the blockchain space as well as a design pattern for more experienced architects.

Imagine the following use case. You have received a document that has passed through several hands and you want to make sure that it is authentic. The document might be the service book for a car that you are buying, or it might be a document asserting that the house that you are buying really belongs to the seller. It could be anything.

Digitally signing documents to ensure that they haven't been tampered with is nothing new. You can take the contents of the document and produce a cryptographic signature with them, which is sent along with the document itself.

In this article I'm going to describe the simplest blockchain application that I know, with code examples.

The receiver of the document can produce the signature again and verify that it matches with the one provided. A match means that the document hasn't changed.

This is what you get with MD5 checksums and it is very convenient to use. Its weakness is that you need to receive the signature to verify the authenticity of the document. If someone gets in the middle and changes both the document and the signature you will be none the wiser.

Since getting data from individuals is not something that can be easily trusted, sometimes a third party steps in to provide a record keeping service for a profit. This profit motive is what keeps the record keeper honest.

Workable, but not a perfect solution. The record keeper will naturally work itself to a position of power and extract high rents with no competition. Even worse, if economic incentives change the record keeper might become corrupt. Who watches the watchmen?

What Blockchain can do for you

A blockchain datastore is decentralized, robust and immutable.

- Decentralized means that the datastore runs collectively on the hardware of several different people.
- Robust means that the datastore will continue running even if some of the participants leave or stop cooperating.
- Immutable means that once data is stored in a blockchain it cannot be altered.

Blockchain solves the document registry problem in an elegant way. Once we enter a signature in the blockchain registry we don't need to fear that the signature will be altered to match a tampered document. To do that a majority of network participants would have to agree on the change which makes it hardly secret.

At the same time, there is no one in a position of power to extract rent for running the service. The network participants provide the service themselves.

In this context, a document can be any data set. The same pattern will work for attesting to the authenticity of any business transaction, IoT data set or user identity, among many others.

Blockchain solves the document registry problem in an elegant way. Once we record a signature it cannot be tampered with.

If I would be writing this in 2016 I could implement about a hundred lines of code, then a whitepaper with two dozen pages, and raise a few million in an ICO. The applications for this pattern are endless and lots of people did take the opportunity.

Nowadays, it is useful to know how to use this building block to design more complex solutions. All blockchain solutions rely on storing user generated data that can be trusted without depending on anyone.

Implementation

This time I didn't code the contract from scratch. I'm trying to stop reinventing the wheel and blockchain registries have been implemented by the dozen. A quick google search took me to the GitHub repo that I'll use to show how the pattern works. Even if it uses an old version of solidity it works just fine and you can test it (thanks Svetlin!).

The contract is very simple with only one relevant contract variable and two functions.

- A documents

mapping links the hash calculated for a document with the block in which it was added.

- An add

method takes a hash and stores it in the mapping.

- A verifiy

method returns the timestamp for a hash.

The frontend allows you to upload a document with contract.add

and calculates a signature as a sha256 of the document contents.

The frontend also allows you to upload a document with contract.verify

and if it was uploaded before it will tell you approximately when.

That's all that you need to implement a decentralized document registry that allows two things:

1. To sign a document
2. To verify if a document has changed since its last recorded signature.

This works because the chance of two different documents having the same signature is very close to zero. If you get a timestamp back you can be certain that the document that you are providing was introduced to the registry at that time.

The contract code can certainly be updated and improved, but the core behaviour in those 17 lines is still correct.

Conclusion

A document registry is the simplest implementation of a blockchain application with business value. In 2016 the few lines of code in this article would have been enough to raise several million dollars in an ICO. Today they are still relevant as one of the building blocks that you can reuse again and again in more complex solutions.

A document registry effectively uses the decentralization and immutability properties of blockchain to eliminate the need of trusting anyone that the data provided is genuine. The idea is simple but revolutionary.