Crypto Will

Decentralized testaments on the blockchain

Abstract

During the early ages of crypto-currencies, more and more people started to join the network. Every person has the inevitable outcome of death. In the age of fiat-currencies it’s possible for others to manage your funds. With crypto-currencies, assuming that only the person in question has the private key, no one would be able to recover said person’s funds. The person has to do it – post-mortem.

In this white-paper, we take a look into the CryptoWill-program, which is capable of storing both physical and digital assets and goods.

Table of Contents

The cryptographic workings

There are 3 roles in the CryptoWill network:

- Will author (makes a will for his successors)
- Successors (i.e children)
- Will keepers (nodes in the network keeping the wills in the form of cold-storage)

When the process begins, the will author will create a list of items, both physical and digital, and writes them down in his will. Each item can be assigned to a successor, like the family.

Each successor gets his own password for decrypting their part of the will. This is happening using symmetric encryption (i.e aes256). In essence, when you leave your will to your mother and sister, the sister can view the part of the will directed to her, but not the mother's, unless the mother discloses the information with the sister.

As a final result, a **master key** is created**.** This is a public-private keypair, where the public key is stored & signed with the will on the cold-storage network. The master key allows the creator to make changes to the will. Because the public key is stored with the will, all one needs to edit one's will, is the master key. The master key serves as a digital signature method to prove ownership, in case that's needed, and also serves as a seed to derive any other keys from (like the passwords for the successors) or a third-party data-storage network (like Siacoin).

The encrypted result is being uploaded to a decentralized cold-storage network, along with some WillCoins, representing a finite amount of time of storage. Note that no one has to pay for the storage for a long time, this makes the will-creation process prepaid, and unattended for years to come.

For the creator of the will, it's important to share the passwords with each successor, as upon death, they can access the will's contents.

Will Creation & Upload

Upon will creation, the file is being encrypted and sent to a fixed amount of random, independent nodes. Let's make that *n = 10.* The nodes all get a full copy of the file. Note that we don't have to trust the nodes won't grab your private keys, since the contents are unreadable for them.

Along with the will, a smart contract is created and deployed on the block-chain, serving 3 purposes: **Proof-of-Life, External Proof-of-Storage** and **Rewards**. More on that later.

The will author will be presented with an interface where a few options can be selected, for instance:

- Respond after 7 days
- Respond after a month
- Respond after a year

What will be selected will determine the deadline for when the creator has to ping and perform a Proof-of-Life. When the creator fails to respond, death, or an otherwise inability to respond is assumed (i.e kidnapping), and the nodes storing the will, upload it to a third-party storage network, so successors can inherit the money.

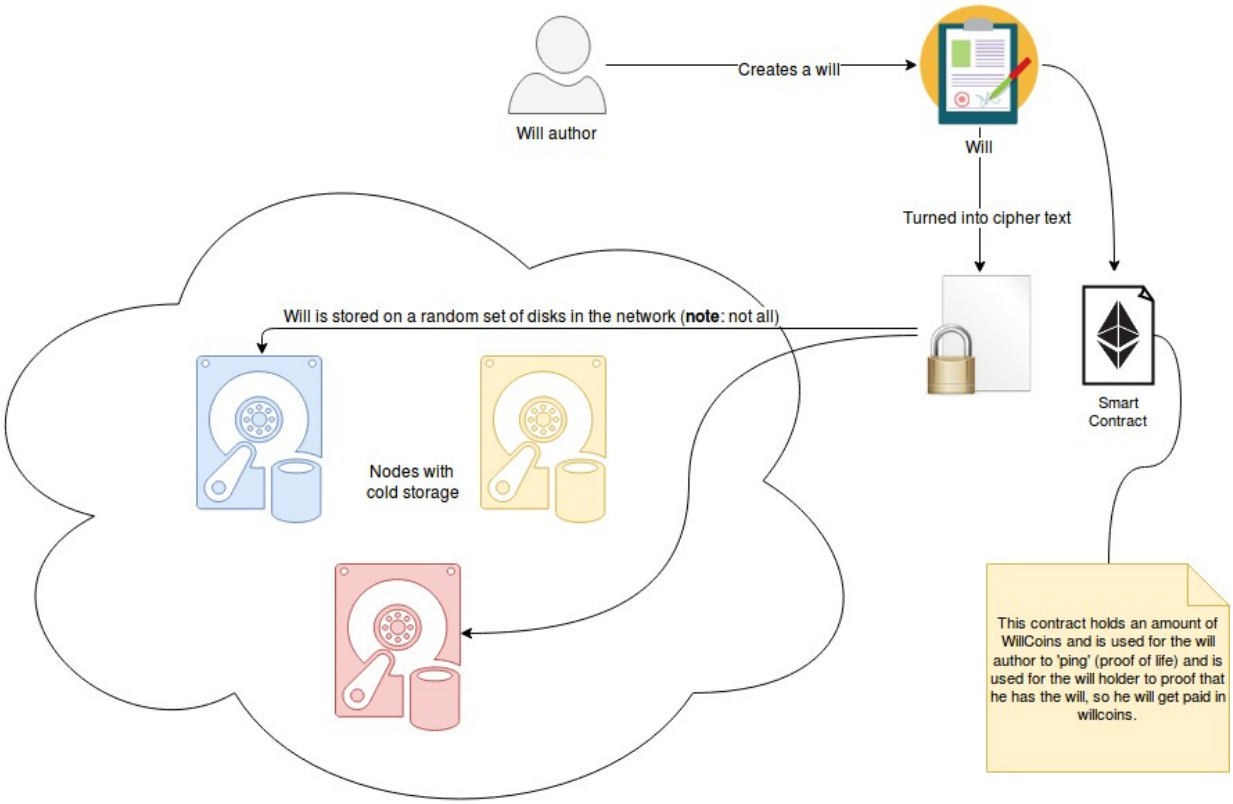

Illustration 1: Creating a will

Cold-Storage

Cold-Storage is one of the key elements in the CryptoWill network. It’s purpose is to store wills in such a way that it is guaranteed to be kept for as long as necessary.

In ‘Will Creation & Upload’ we’ve discovered that the full will is spread across *n*-nodes. Those nodes will deny access to anyone trying to access the full file (with exception of validation, where only hashes of the files are exchanged).

When the will author failed to respond to his proof-of-life before the deadline hits, the will is published to the blockchain.

This prevents the evil successors from trying to access the will before the author has died.

Cold-Storage: Validation

Making sure the will stays on the network for as long as necessary is crucial for the availability of your wills. To ensure a good trade-off between high availability and cheap, cryptographic cold storage, we explore 2 different kinds of validations in the CryptoWill network: **Internal** and **External** validation.

Internal validation

This validation happens between nodes that hold the same will. Ideally this is *n* nodes.
The purpose of this validation is making sure that 1) guarantee 100% that at least *n* nodes have the file and 2) replicate will to new nodes if there are not enough nodes with the file.

The validation is quite simple: nodes will ask each-other to present a hash, made of: hash*(will_contents + nonce)*, where the nonce is generated by the node requesting the validation. If another node replies with the same hash, you can be certain that node has the will completely. This process is repeated with different nonces until the will holders are satisfied, and if otherwise, replicate accordingly.

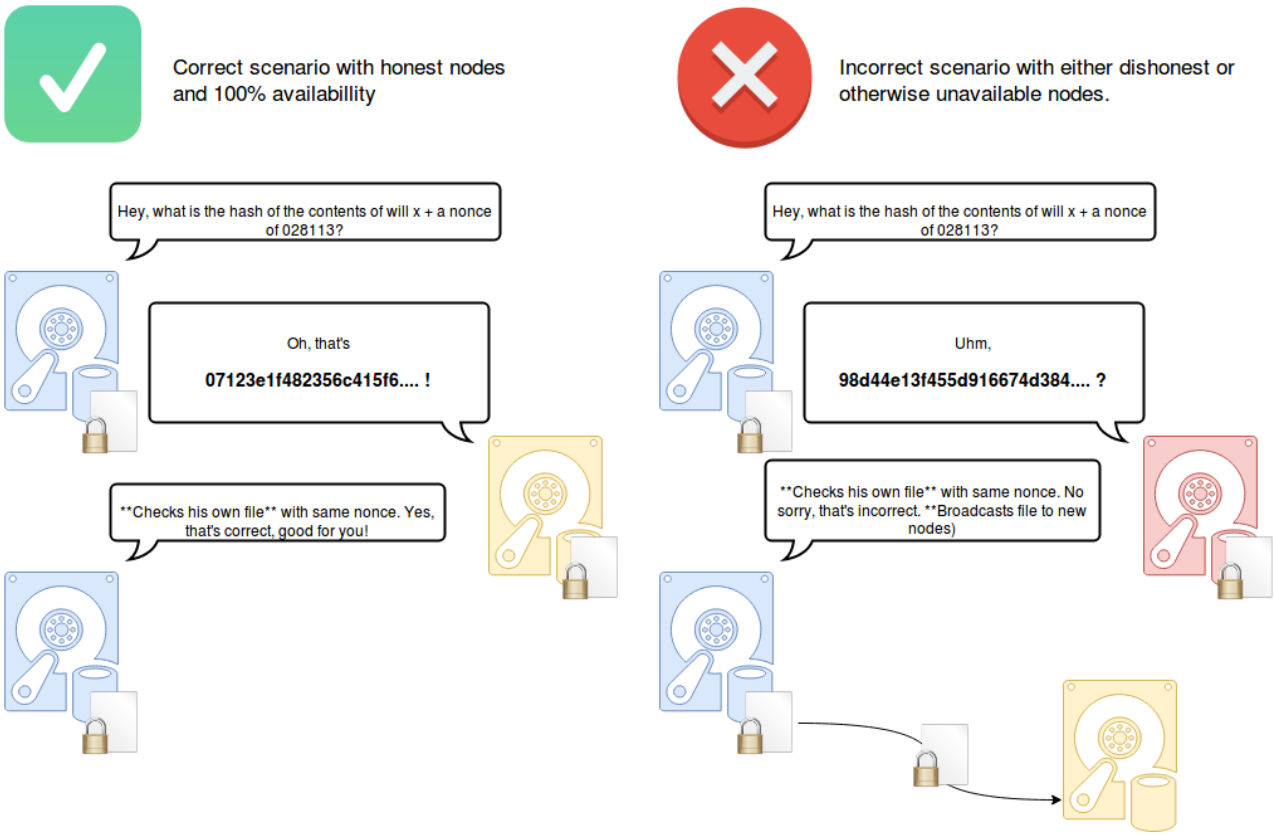

Illustration 2: Internal validation

External validation

External validation, like internal validation, helps to validate that a certain node has a will, but does so against a smart contract on the blockchain instead of directly to other nodes.

The purpose is **rewarding WillCoin** to the nodes for storage. The smart contract, made during will-creation, has a Merkle-tree of the will uploaded.

Each certain amount of blocks (i.e each 300 blocks, corresponding to around each month) the nodes have to proof their storage to the contract. The contract will select (based on previous events on the blockchain) a random leaf for nodes to send proof of storage from.

There are 2 big advantages to using Merkle-trees:

1. The contract doesn't need to store the whole file, making the contract a lot smaller, but also preventing that an evil successor can grab the will by downloading each contract and try their password against each will in that contract until a match has arrived.
2. The amount of data needed to be sent to the contract is minimal (possibly around 4kb). This is because we only have to send the contents of a leaf.

The contract then validates the hash given by the node(s) against it’s internal Merkle-tree and if there is a match, WillCoin will be awarded to the node giving the correct contents.

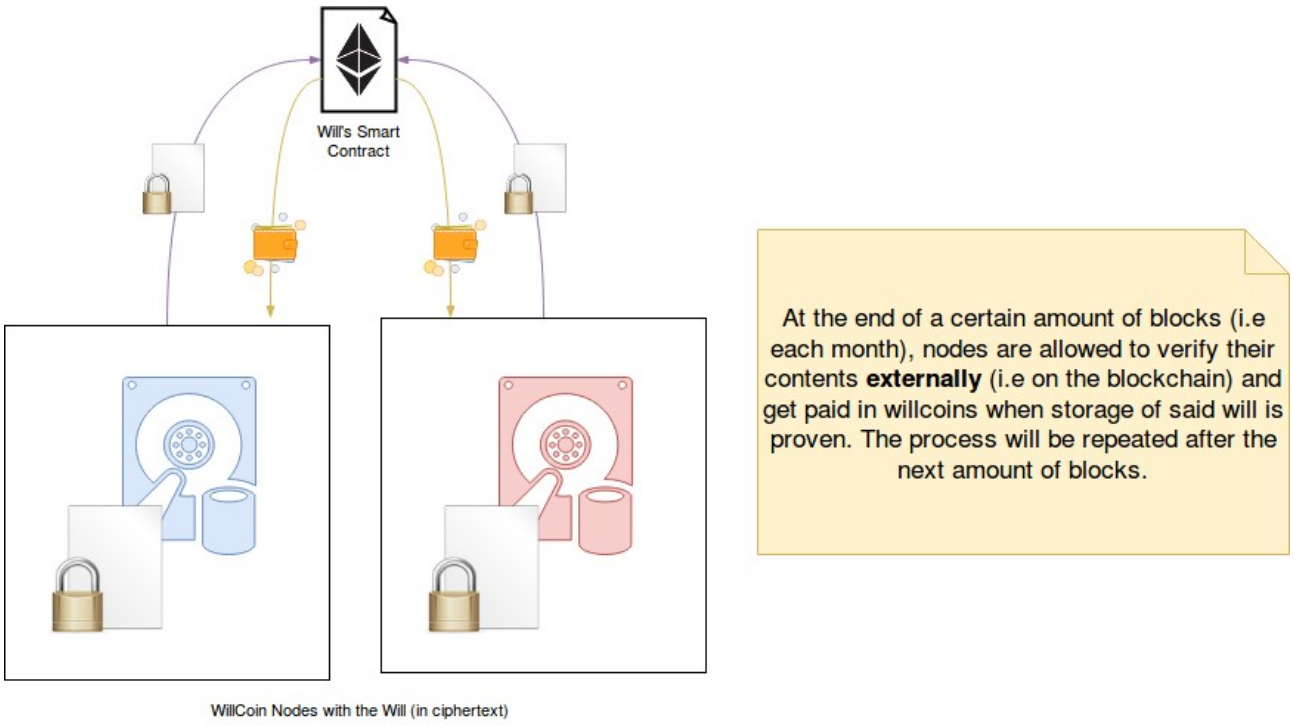

Illustration 3: External Validation