chestnut

7/9/2019
Draft 2

Table Of Contents

About

A significant obstacle to widespread adoption of blockchain technology and crypto assets is the fear users have in taking responsibility for managing and protecting their own money. Chestnut bridges the best security practices from the traditional banking system into the blockchain space by providing users with familiar safety features as well as the autonomy and transparency that the blockchain offers.

Chestnut utilizes a smart contract within a multi-signature account to allow users to set specific and customizable rules or restrictions on the activity, size, or type of transactions that can be executed by their account.

Similar to security and fraud protection offerings by traditional credit card companies, Chestnut users can set spending limits, whitelist/blacklist recipients, freeze the account, or nominate beneficiaries in case of emergencies. With Chestnut, users can set their own account parameters, rather than rules imposed by a third party institution.

Each account transaction passes through our smart contract and if it does not fit within the user’s set parameters, Chestnut does not approve (or sign) the transaction as part of the multi-signature process. Chestnut does not require, collect, or store private information and Chestnut cannot make changes to a customer’s account unless the customer initiates the change.

Chestnut provides users with the peace of mind to leap into the world of crypto assets.

The Problem

One of the many benefits of blockchain technology is its ability to create a single source of indisputable truth over time, making the blockchain auditable and transparent. However, the permanence of blockchain transactions is unfamiliar and can be devastating when a mistake occurs.

Users have become accustomed to the protections of the traditional banking and credit card systems, which typically offer protective services, like identification, the reversal of fraudulent transactions and spending limits that restrict uncommon large purchases or withdraws. To date, there is no blockchain solution that offers users this type of comfort in transacting with cryptocurrencies. Even in a basic token transfer, a small typo can lead to tokens being lost forever. Stories of lost crypto assets have plagued the cryptocurrency ecosystem, deterring potential users from participating in the new economy.

How can we expect mainstream adoption of blockchain technology when individuals and businesses face such large risks transacting with cryptocurrency? We must develop the tools that lead the way to safe and prosperous blockchain communities.

Previous attempts to solve the issue of user mistakes, such as ECAF, relied on off-chain arbitration that attempted to correct a mistake after it was made. This is reactive, doesn't prevent the transaction from happening in the first place and has scaling challenges.

Chestnut is proactive and prevents mistaken or fraudulent transactions from happening prior to being added to the blockchain. Its on-chain solution can scale to meet the demands of the millions of users that

Given that blockchain and EOS.IO are relatively new technologies, it is fair to assume that the average reader is unfamiliar with much of the EOS Mainnet design and therefore a few fundamental concepts must first be understood.

Introduction To EOS.IO

The EOS.IO software is a blockchain architecture utilizing an operating system-like construct which can host a new generation of smart contract based applications. At its core, EOS.IO facilitates permission based accounts that communicate by transmitting actions amongst each other. Accounts can handle these actions accordingly and record application state to the blockchain by running a smart contract. Accounts are permission based[1] and allow any weighted combination of other accounts and private keys to control them. This feature allows multiple entities to own a single, multi-user, account.

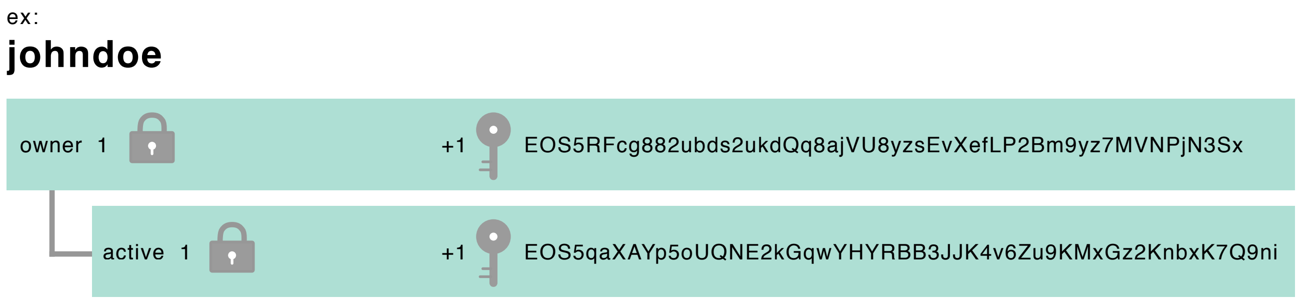
Smart Contracts

Smart contracts are computer programs which can run on top of any EOS.IO account allowing them to process incoming actions, evaluate business logic, and save data to the blockchain. Often smart contracts and the account running them share the same name as to avoid confusion. For example, as you will learn in our usage, the account running the `chestnutmsig` contract is also named chestnutmsig.

Default EOS Accounts

chestnut

A default EOS.IO account has two permissions, `@owner`, which can perform any action for the account, and `@active`, which can perform any action other than changing the owner key. The most commonly used account structure consists of an `@owner` permission private key, which is kept offline for back-up, and an `@active` permission private key which is imported into a wallet and used to sign everyday transactions on the blockchain.



ex:
johndoe
owner 1
+1
EOS5RFcg882ubds2ukdQq8ajVU8yzsEvXefLP2Bm9yz7MVNPjN3Sx
active 1
+1
EOS5qaXAYp5oUQNE2kGqwYHYRBB3JJK4v6Zu9KMxGz2KnbxK7Q9ni

Figure 1. A default EOS account

Even though this default configuration works for regular blockchain use and provides users with the ability to recover a lost active key using their owner key, it still requires the unfamiliar and cumbersome process of maintaining two private keys – one offline and one in a wallet. If the owner key is lost, there is no recovery. Placing all of this risk and security on the user rightfully instills fear in many newcomers.

To minimize the risk of lost, hacked, or stolen private keys/accounts, people can adopt multi-signature accounts with trusted third parties. Removing the security burden from the customer and placing it on a known and trusted third party is analogous to third-party financial institutions who are entrusted to handle and secure customer funds.

The utilization of EOS.IO smart contracts and account permissions give multi-signature accounts an advantage. Firstly, multi-signature accounts do not need to hold custody of the customers' assets. Secondly, customers can knowingly trust a smart contract as a third party since the code is both open-source and publicly viewable on the blockchain.

Smart Account System Overview

The Chestnut smart account system needs two EOS.IO accounts to function; one running the `chestnutmsig` smart contract and the other as the smart account user. The Smart account user converts their existing EOS.IO accounts from a single user account into a multi-user account – requiring signatures from both the user and the chestnutmsig account to execute a valid transaction. Rules programmed into the `chestnutmsig` smart contract for approving transactions allow the chestnutmsig account to automatically determine whether or not to sign the given transaction along with the user's initiated action.

Smart accounts are multi-user accounts that link themselves with the chestnutmsig account (running `chestnutmsig` smart contract). A smart account utilizes a custom account structure that requires both the chestnutmsig account and a users' private key to act as either the users' `@active` or `@owner` permission.

Smart Accounts are created by turning an accounts `@active` permission into a multi-user permission made up of both the chestnutmsig accounts permission `@smartaccount` and the users `@chestnut` permission. Whenever the user wishes to send a token transfer or any other secured "smart" transaction supported in the future, they must complete a multi-signature transaction with valid signatures from both themselves and the chestnutmsig account.

ex:

smartjohndoe

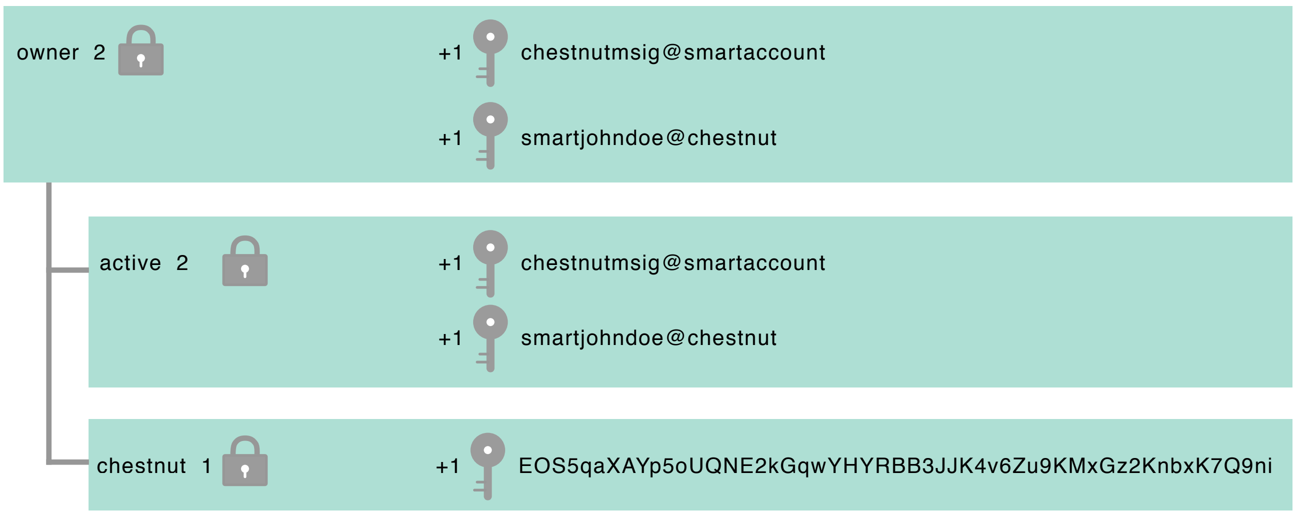

owner 2
+1 chestnutmsig@smartaccount
+1 smartjohndoe@chestnut
active 2
+1 chestnutmsig@smartaccount
+1 smartjohndoe@chestnut
chestnut 1
+1 EOS5qaXAYp5oUQNE2kGqwYHYRBB3JJK4v6Zu9KMxGz2KnbxK7Q9ni

Figure 2. A Chestnut EOS smart account

In Figure-2, we assume a user owns the Chestnut smart account smartjohndoe. The account has split its active permission between its `@chestnut` permission and the `@smartaccount` permission of the chestnutmsig account. Both `smartjohndoe@chestnut` and `chestnutmsig@smartaccount` have an authority weight of 1. `smartjohndoe@active` requires an authority weight threshold of 2 to validate any transaction for `smartjohndoe@active`. The only way for transactions to succeed is if both the user and the chestnutmsig account permission `@smartaccount` sign two halves of a multi-sig transaction.

A smart account has three permissions; an `@owner` permission used for recovery and beneficiaries, an `@active` permission used for smart account smart transfers, and a `@chestnut` permission used by the end user to control the account.

The `@chestnut` permission can only interact with the chestnutmsig account and can propose and approve multi-signature transactions forcing the smart account to send proposals to the chestnutmsig account to validate. The permission can also be expanded to allow interaction with other smart contracts (dApps) by linking the `@chestnut` permission authority with the dApps contract allowing the smart account to maintain an active whitelist of dApps they feel secure using.

The `chestnutmsig` smart contract runs on top of the chestnutmsig account. The chestnutmsig account has a child permission to its `@active` permission called `@smartaccount` with the single authority `chestnutmsig@eosio.code`. The `@eosio.code` authority is a way of stating that the smart contract code itself has the authority to sign transactions. Since this is the only authority added to `chestnutmsig@smartaccount` only the smart contract code can sign and execute the second half of smart account multi-signature transactions.

ex:

chestnutmsig

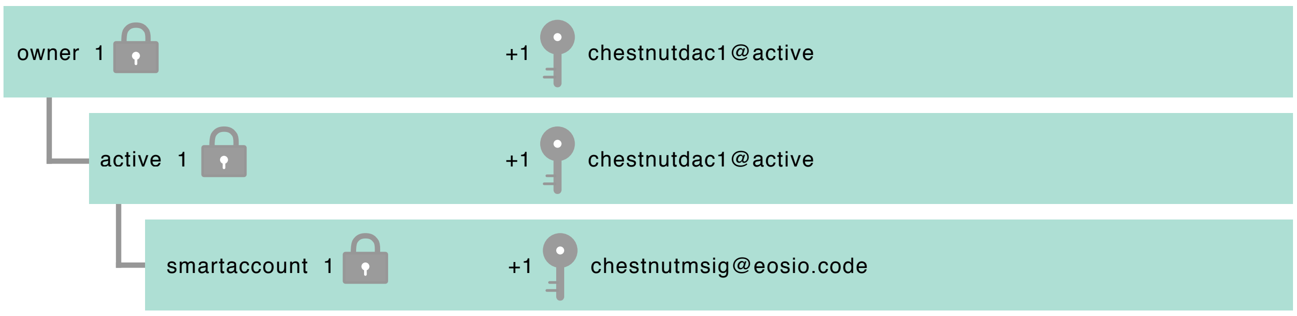
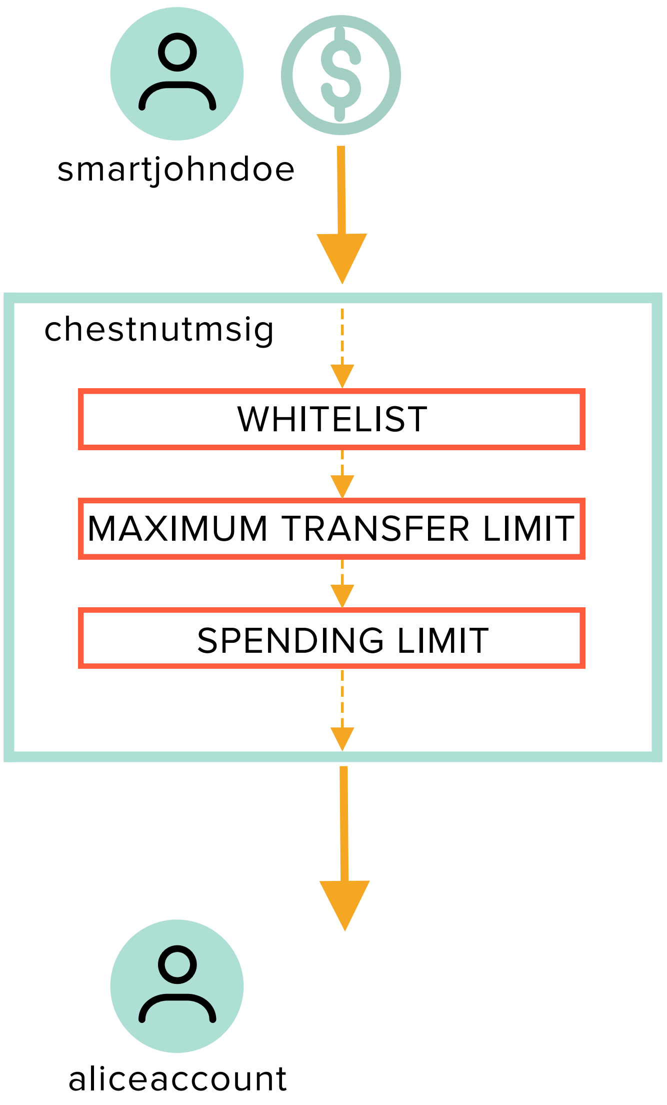


Figure 3. The chestnutmsig EOS account

Multi-signature token transfers proposed by a smart account and executed by the chestnutmsig account are called smart transfers since they add a level of brains to the transfer. Currently, smart transfers check against three user settable security parameters. Smart transfers check the recipient against a whitelist, the quantity against a transfer limit, and the total amount spent overtime against a spending limit. The whitelist confirms that the smart account trusts the recipient and that there are no errors or typos in the recipient account name. The quantity limit protects the smart account from accidentally sending an unintentionally large amount of tokens at once, and the spending limit provides fine-grain control over the smart accounts' total spending. Each of these parameters can be toggled on or off by the smart account allowing the use of specific features as needed.



Smart accounts also contain a smart contract (dApp) whitelist to manage which smart contracts the smart account is comfortable using. The smart account can interact with any contract on this whitelist and automatically blocks unknown contracts for the user.

To fully understand the importance of a smart contract whitelist, one must first know that all EOS.IO smart contracts are by default permissionless allowing anyone to call them. One can see this by looking at the vast amount of unique block explorers that although have very different user interfaces all call the same contracts under the hood. Nevertheless, along with this power comes responsibility and numerous scam and phishing imitator apps have emerged. These scams look identical to the real app but instead call different malicious smart contracts on the backend tricking users into losing tokens and stealing digital property. Allowing smart accounts the ability to whitelist trusted smart contracts renders this deception obsolete and apps calling imitator contracts are automatically blocked.

It is important to note that the smart account user has full control over which smart contract accounts they whitelist leaving them liable for their own research when adding to the whitelist.

As trusted community whitelists are released in the future, we will pursue partnerships and integrations for further improved user experience. This could reflect users having the ability to set thresholds automatically based on these services (rather than manual inclusion onto their personal whitelist) as well as using them as a jumpoff point for their own customized inclusions.

If a user wishes, they can set up their smart account with a multi-signature `@owner` permission to enable trusted account recovery using one or more beneficiary accounts. Signatures from these trusted beneficiary accounts are then used to change the smart accounts private key if it is ever lost or misplaced.

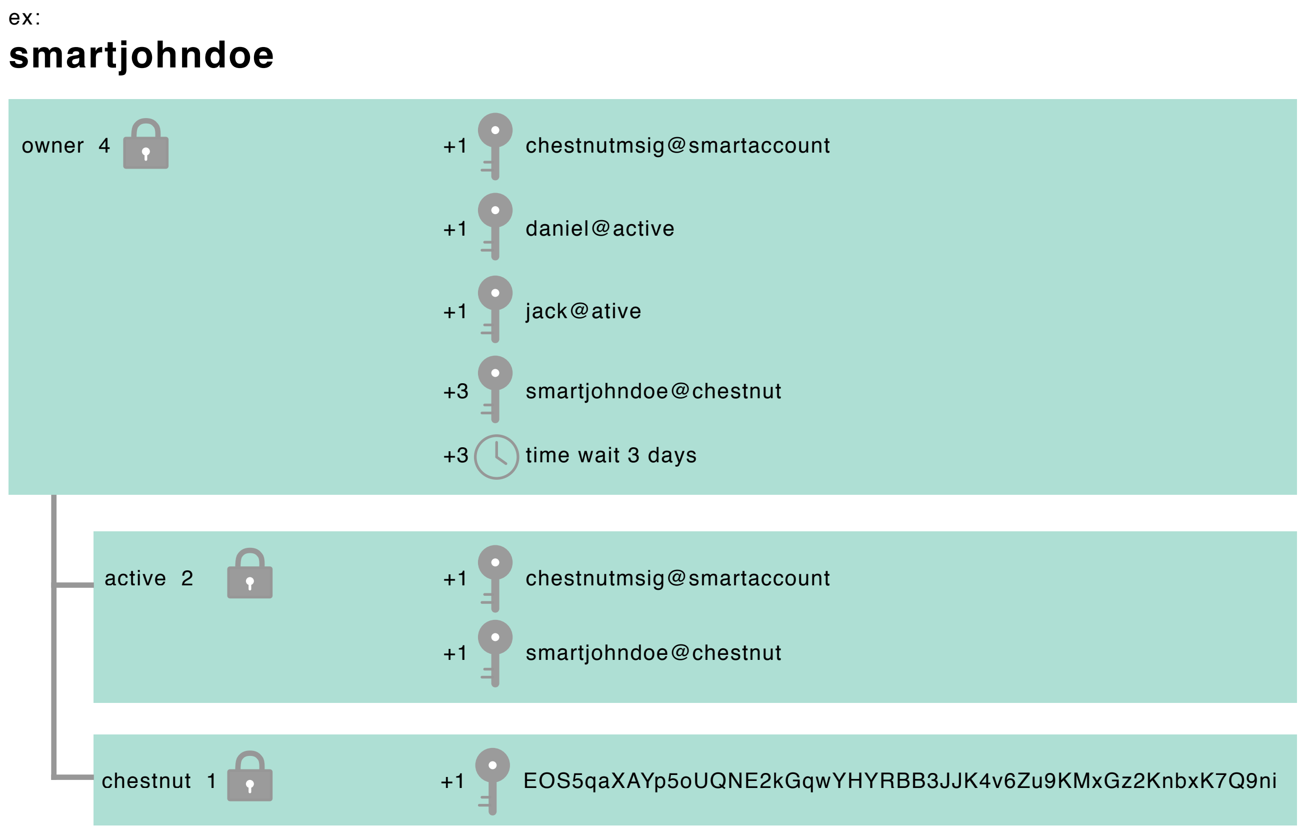


Figure 4. A Chestnut EOS smart account with recovery beneficiers

In the above example, the smart account smartjohndoe has added both jack and daniel as trusted recovery partners and beneficiaries. The `@owner` permission is required to change the `@active` or `@chestnut` authorities of a smart account. In the above configuration, the `@owner` permission has a threshold of 4, which means the weight of all authority signatures summed together must be equal or greater than 4 to be considered a valid transaction. Anything less does not pass.

smartjohndoe has lost his key `@chestnut` and needs to switch it to a new key. He needs one of his trusted beneficiaries, either **daniel** or **jack** to recovery his key.

1) **smartjohndoe** loses his key and will need either **daniel** or **jack** to for this account for him

```
+1  daniel          +1  jack
+3  wait time       +3  wait time
-------------       -------------
 4  PASS             4  PASS
```

2) The three days wait till will give **smartjohndoe** time to cancel the transaction with the chestnutmsig smart contract in the event **smartjohndoe**'s beneficiaries are acting maliciously.

```
+1  chestnutmsig@smartaccount
+3  smartjohndoe
-----------------------------
 4  PASS
```

It is also possible to use an additional multi-user account composed of multiple beneficiaries to recover a smart account. Doing so would let the smart account require more than one beneficiary in a multi-signature to recover the account. The newly created account must also be paid for and funded with resources.

Chestnut Smart Accounts provide an additional level of security and certainty to any EOS.IO account without compromising property ownership. We envision most every day EOS.IO users utilizing the entirety of Chestnut Smart Account features to feel safe within the blockchain realm. We also see a strong use-case for businesses to adopt a similar smart account structure to keep their digital assets as safe and automated as possible.

Learn more at *http://www.chestnutaccounts.com/*

[1] See “Role Based Permission Management” in *https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md#accounts*