Swingby Trusted Cloud: A probably honest blockchain infrastructure

Yusaku Senga - yusaku@swingby.network

Swingby Labs

4 Oct. 2019

WORKING DRAFT

Table of Contents

1. Introduction

1.1 Motivation

The current blockchain ecosystem has come far but still faces two major issues: scalability and cross-chain interoperability. Scalability is the rate at which transactions can be processed by the network and interoperability is the ability to move assets from one blockchain network to another without needing to trust an intermediary.

Ethereum is a good case in point. The Ethereum community is preparing version 2.0 of the protocol as an attempt to increase Ethereum's low transaction throughput. The protocol, still under development, proposes a new Proof of Stake (PoS) consensus algorithm, variants of which are used by other blockchains today. These PoS consensus algorithms rely on the majority of validators being being honest - performing as others expect them to. But validator nodes operators can run customized node software on their machines, and if these operators collude, they can create dishonest, or unexpected, outcomes. This is a problem.

Regarding interoperability (the movement of assets from one blockchain to another), various solutions exist today that bridge blockchains. These include: soft pegging via self-stabilizing mechanisms, relay mechanisms, custodian solutions, and oracle solutions. However, these solutions give rise to new problems. For self-stabilizing mechanisms, a huge amount of liquidity and a large user base is often required. Other solutions require trusting certain entities: Custodian-based solutions require trusted custodians; Relay mechanisms (including oracle solutions) rely on block headers created by trusted server operators.

With today's proposed solutions to both scalability and interoperability problems, we still need to trust the honesty of node operators, and trust that the nodes they operate will perform as the operators claim they will. The problem is this *need to trust*.

2. What is the STC?

The Swingby Trusted Cloud (shortened as 'STC') is an infrastructure consisting of multiple physical hardware STC machines, called STC nodes, connected to each other over the internet.

STC nodes are designed and constructed to be both technically and physically secure. The security principles are described later in this paper.

On these physical STC nodes, various public blockchain validator node software (eg Bitcoin core, geth, bnbchaind, etc) can be run inside containers such as Docker. Each of the blockchain nodes can be publicly verifiable as *honest*, due to the initial setup, described later.

This is a solution at the lowest layer of blockchains - the environment in which blockchain nodes are running, inside containers.

How does STC prove that the blockchain node software being run is trustworthy? Two mechanisms will be used, one at build-time and the other at run-time. At build-time, the blockchain validator node software (eg Bitcoin core) will be open-sourced and published on the internet (eg on Github) for anyone to review. The STC nodes will build a Docker container compiled from the public repositories and push the image to Docker Hub while also running an instance of the same image. This ensures that any member of the public can verify that the blockchain node software running inside the nodes is exactly a compiled version of the published code.

Physically, the STC nodes are inaccessible after deployment into an underground concrete box. Once deployed, the only way to control an STC node is via specialised transactions broadcast on the Ethereum blockchain. STC nodes are programmed to perform operations based on these transactions.

If the STC node's initial setup process is fully verified, these technical and physical mechanisms ensure that app containers running on STC nodes can be trusted to operate as intended: there is no need to trust the operator. More details can be found down in the *App containers* section.

3. STC network and node overview

The STC is a network of physical STC nodes. The main purpose of the STC nodes is to host custom application containers which are ensured to be provably and immutably honest, and behave as expected. Each STC node can host multiple Docker containers that run provably honest immutable blockchain validator node software (eg Bitcoin core, Ethereum, Binance chain, etc) that connect with a whitelist of other validator nodes on their respective blockchain networks (eg other Bitcoin validator nodes, Ethereum nodes, Binance chain nodes, etc).

STC nodes are managed remotely by instructions sent as Ethereum transactions on the Ethereum blockchain. This ensures that the STC nodes will only perform a limited set of known and tested functions. Key functions include: build container, start/stop container, delete container.

The value of the STC is that all participating STC node and application containers can be trusted because of the following factors:

- STC nodes are physically secure
- STC node's filesystem is hashed on initialization (after deployment)
- only immutable container deployment cycles are supported, upgrading a container is not possible
- STC nodes can only be controlled via confirmed transactions transmitted on an established blockchain (eg an instruction sent to a specific smart contract address on the Ethereum mainnet)

Blockchain node software will be developed to run inside Docker containers, for example bitcoind, ipfs-local-node, geth, bnb_light_client.

4. STC node daemon

STC node contains several daemons that manage each container's behavior:

Each STC node runs three core daemons:

• A docker daemon	Use docker daemon's kernel namespace isolation to provide application container with isolated environment from host machine
• STC-core (the STC daemon)	Reads events from the Ethereum contract. all of the processes that related application container is executed in this daemon.
• whitelisted-node-list-server	Holds a list of public, test, and private node list defined on the Ethereum Contract. A container can call API call to get node lists.

5. STC node system architecture

Fig. 1 below is a representation of a simplified version of the system architecture of an STC node.

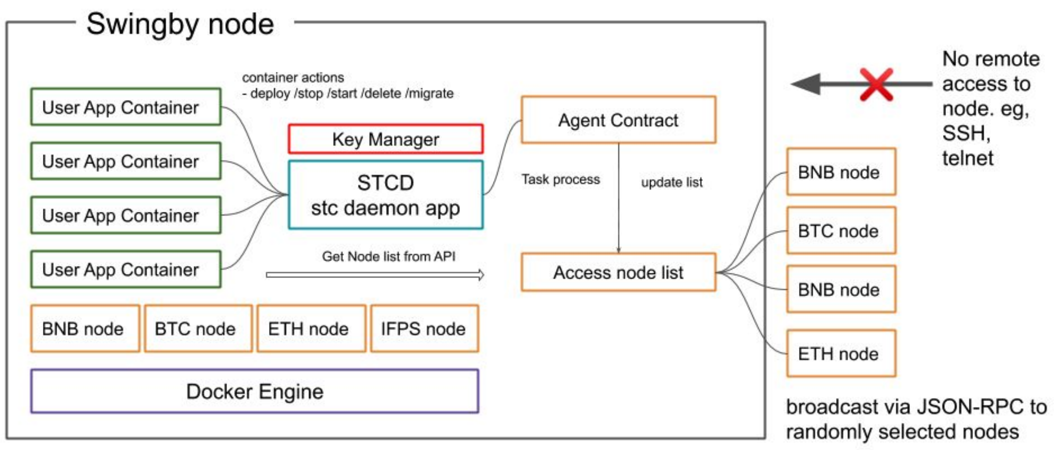

Fig. 1 STC node system architecture

5.1 Proof of restricted access

Every STC node is initialized with a filesystem that has disabled remote access to a node (eg. via SSH, telnet). This filesystem is open and exported can be double checked. A node's filesystem is hashed on initialization after deployment and this hash is reported. Anyone can build the same filesystem locally and create a hash to confirm this hash is correct.

Every STC node is initialized with a filesystem that has disabled remote access to a node (eg. via SSH, telnet). A node's filesystem /bin/* executable files are hashed on initialization after deployment and this hash is reported to IPFS and Etheruem. Anyone can build the same filesystem locally and create a hash to confirm this hash is correct.

6. The physical node deployment

The basic solution for security provided by STC is that a *stc host operator* is de facto also a *landowner* and therefore is subject to the legal rights of land use in whichever country their physical hardware is located. Physical attacks on nodes require trespassing on private land, and such attacks are expensive and predictable. Because a physical attack can make the system malfunction, but you cannot earn money. Attackers intend to take more damage because they can be monitored by surveillance cameras. As a result, it's possible to create an environment that is extremely difficult to attack.

Nodes are initialized after physical installation to ensure the initial setup hasn't been compromised.

6.1 Land governance

Users of STC need to monitor which regions (underground land space for multiple stc host) have what security risks and transparent governance. However, there is no need for a complicated voting system or consensus model, and the choice of nodes can be freely chosen by the user. Physically, STC nodes are set in concrete boxes and reside underground as shown in Fig. 2 below. These structures are physically inaccessible after installation and therefore STC nodes cannot be tampered with by anyone, even someone with physical access to the area. The only thing a physically present malicious actor could do is shut down the power and/or physically shut off network access. Such attacks can, however, be detected and monitored through surveillance cameras and other commonly deployed tools against physical breaches.

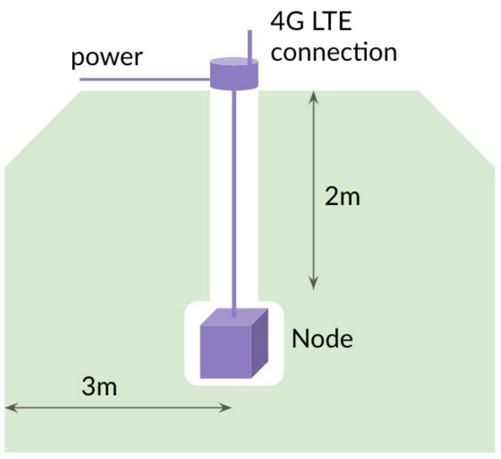

Fig.2 STC Host

Fig. 2 shows a simplified representation of an STC node, deployed into concrete box, and buried at least 2 meter underground and protected by a concrete structure.

6.2 Physical node specs

The STC node must be installed in a place surrounded by concrete where it is easy to visually check the STC node structure. Anyone can suggest a node host. The basic requirements for a physical node are as follows:

- The physical node is guarded by a 2m x 3m x 3m solid concrete block, and is durable to 200g of common C4 plastic explosive.
- Backup power supply is provided by UPS which can last at least one hour.
- External USB connection memory is prepared for storage of the seed key of the application container.
- The wifi antenna is extended over a distance from 2m underground to 10cm underground and connects with the ground WiFi-router.
- Heat radiation by heat sink.
- New Air flow mechanism for underground node.
- Deployed in areas of low humidity and low temperature.
- At least 6 CPUs are installed.
- At least 32GB of memory is installed.

7. App containers

Anyone who wants to make use of the STC network can do so by deploying a custom app container to one or several STC nodes. App containers are purely autonomous applications that only interact via Ethereum transaction - even on initial deployment (see section 7.1 below).

While security and integrity is ensured by the STC node sandbox, the limitation of what an application container can do is only dependent on the imagination of the team that develops it.

7.1 Deployment

An app container cannot be directly deployed to an STC node because the node restricts access physically and digitally. Therefore deployment is done through instructions sent to the node via specialised Ethereum transactions over Ethereum mainnet.

Once a company or community has created an application to be run on STC, it creates two things:

1) a dockerfile that is used to build a docker container running the app
2) a hash of the container image built from the dockerfile

The dockerfile should first be uploaded to IPFS. After this, the IPFS pointer to the dockerfile is submitted to an Ethereum contract, along with the hash of the container built.

7.2 Wallet integration

After the deployment of an application container, a 32 byte random seed is generated for this application container. This random seed can be accessed via environment variables which can be set up before deployment of the container. This can be used to set up an internal wallet for the application container. This makes it possible for each application container to have a private wallet and no one to ever know its private key without physically breaking into the concrete shell of the node.

7.3 Updating an app container

The only way to update an app container is by deploying a completely new app container and terminating the previous one.

Deployment of app containers can enjoy extra security by implementing a voting mechanism before submission of the deploy transaction to the STC network. Updating an app container is not possible by design to ensure the version which was deployed will always be that version.

When a new application container is deployed any assets previously owned will need to be moved over to the new application container. This must be done via logic implemented by the development team of the application container.

8. Value, Cost and Pricing

The value STC brings is that anyone easily has access to a network that allows their applications to **run in a trustless matter**. Even integrating a small part of an existing application with our STC network could greatly improve the trustworthiness of that application.

STC creates countless opportunities for applications that were simply not viable before because they required nodes to be trusted. The biggest value will be our implementation of Bitcoin tokens, explained in detail in the chapter.

8.1 Costs

While we want to keep the network as accessible as possible, we want to keep the costs low so as many people as possible can take advantage of this. That being said, hosting nodes come with initial and ongoing costs. From the land they are installed on, the physically secure structure that's built-in concrete to the electricity and internet connection. And finally the cost of security through network cameras and other methods.

8.2 Pricing

The STC network itself generates some revenue to cover its expenses. Besides this, the countless possibilities for applications hosted on STC will create new businesses and profitability.

Containers deployed on STC are expected to be charged at the same or slightly higher price than container services available on common cloud services, such as Amazon AWS, Google Cloud Platform, and Microsoft Azure Cloud Services. All payments for application containers are settled via a smart contract on Ethereum.