
# Vitro Technology

## IoT to Blockchain

Crypto-Authenticated IoT Block Data for Security and Automated Transaction Processing

Yellow Paper

Applicature

# Introduction

The Vitro Technology Corporation is an Internet of Things (IoT) company leveraging crypto-security and open-source to enable blockchain transactions based on authenticated operating data from remote equipment. Vitro's core service is to deliver IoT Block data including Proof-of-Origin: a non-consensus-based method of IoT data authentication via local ECC crypto hashing and signature. Vitro's revolutionary platform allows customers to control and monitor remote sensors and equipment (e.g., water wells, meters, solar panels, electronic billboards) with authenticated operating data and automated transaction processing.

Founded in 2015 in Austin, Texas, Vitro has developed a full-stack, open-source IoT platform that leverages Elliptic-curve Cryptography (ECC) hardware in every device. ECC is the basis for secure Root of Trust (RoT) data transfer as well as in-device data verification via ECC hashing and signatures. The result is a hardware platform that has qualified for Amazon's AWS highest security rating (HSI[1]) and DigiCert PKI[2] best-practices.

Vitro has successfully executed projects associated with financing from The World Bank, Asia Development Bank, USAID, and other governmental and non-governmental organizations (NGOs). Our technologies have been functioning on the global market for two years, delivering advanced insight and solving problems with IoT device connections in various vertical markets. Installations range from the U.S. to Pakistan, from China to Poland.

---

[1] https://devices.amazonaws.com/search?kw=hsi&page=1

[2] https://www.digicert.com/wp-content/uploads/2017/05/Whitepaper_PKISolutionforIoT_4-12-17.pdf

Applicature

## Our Vision

We believe that the devices we all rely on daily can be connected into trusted networks that will expand the availability of services to people all over the world and while making them more sustainable.

## Our Mission

We build trusted IoT and blockchain technologies to deliver authenticated data that enable innovative financial models, optimize operations, and automate transactions across vertical industries around the world.

## Our Goals

The Vitro team is dedicated to creating trusted networks of IoT devices linked to blockchain technology in order to:

- leverage open-source technology for transparency and security;
- integrate ECC natively from Edge to Cloud to Blockchain;
- enable service purchase agreements for transactions of all sizes;
- develop vertical market communities;
- continually adopt best practices to achieve our goals.

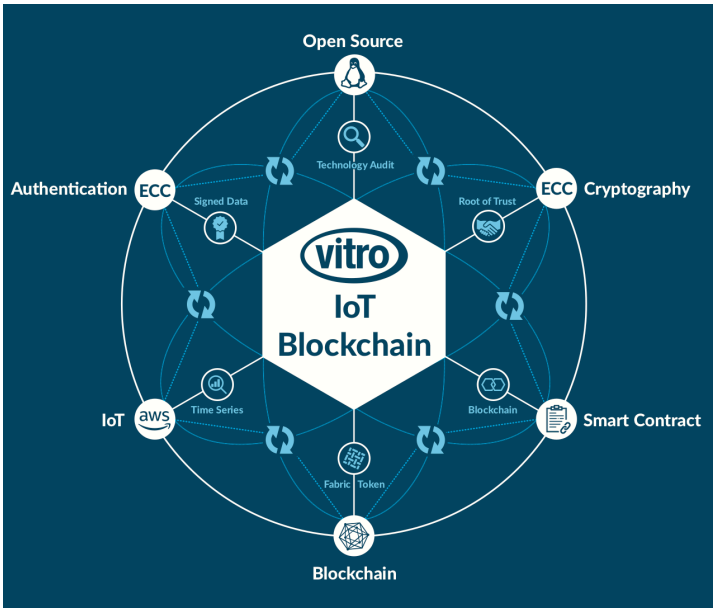
Applicature

# Vitro IoT Blockchain Platform

Merging IoT and Blockchain, Vitro has created a revolutionary new platform that will dramatically improve data security and compliance. Vitro delivers crypto-secure device control and authenticated IoT Block data from Edge to Cloud. This is our core competence. The use of the Proof-of-Origin (non-consensus based method of authentication), allows Vitro to hash and sign data securely into IoT Blocks using local ECC cryptography.

The layers of IoT, Crypto and Blockchain technology make up the Vitro IoT–Blockchain token economy. Each element is capable of being hosted on a variety of platform technologies. For example, IoT device management can be hosted on AWS IoT passing encrypted data to locally or remotely hosted cloud servers. In this way AWS IoT is operating only as a device management system without the keys to decrypt data for edge devices. Higher-order structures of tokens, blockchain, and smart contracts can be applied to IoT Block data across the broad spectrum of emerging global token economies.

Figure 1. Vitro IoT Blockchain




## Motivation For Blockchain Implementation

Blockchain is a decentralized data ledger that serves as a means of storage and distribution of digital information while managing relevant transaction flows in a secure, transparent way. Vitro is targeting Hyperledger as a best practice in Blockchain. Combined with ECC cryptography,

data delivered to and stored in blockchain is secured from end to end, thereby eliminating concern about data privacy on hosted platforms like AWS IoT.

The true value of blockchain is in making business processes faster, safer, simpler, and more secure. Vitro will utilize blockchain technology to:

1. successfully ensure process transparency: all IoT data can be easily accessed, tracked, and verified within the system
2. enable IoT applications to strongly contribute operational and transactional data to the blockchain
3. allow smooth, efficient application of AI and ML on IoT data
4. provide an immutable and irreversible system of data storage
5. ensure a high level of security for data protection with cryptography and encryption
6. radically decrease paperwork and operational expenses
7. ensure fast, secure transaction processing.

## Smart Contracts

A smart contract is a set of codes running on blockchain that controls transactions between parties and ensures that all contract conditions governing transactions are met.

Within the context of IoT, a critical challenge is the management of recurring transaction verification and processing. Each IoT site incurs recurring monthly processing fees that carry costs for everything from hosting to connectivity to billing. The clerical burden of these micro-transactions has limited widespread adoption of IoT despite its obvious benefits. Smart-contract execution of automated recurring service payments is based on authenticated operating data that verifies that services were delivered. Blockchain preserves an immutable record of operating data, service data, and smart-contract logic and execution. This blockchain structure is suitable for audits and financial controls in virtually any application.

Smart contracts can also serve as a transaction engine to deliver securitized equipment purchases to a wide variety of vertical industries. XPA service agreements (‘X’ represents virtually any type of capital equipment) convert the purchase of equipment into the purchase of services yielded by the equipment. A common example is the PPA, or Power Purchase Agreement. This model has permitted the solar power industry to flourish by pairing power consumers with equipment producers. PPAs are securitized agreements for power produced by solar equipment. Smart contracts play the role of pairing the verification of power produced with the matched price per kilowatt hour. Demand for this type of securitized agreement spans many industries, from civic water plants to telecom cell towers. Building a securitized blockchain structure on the crypto-secure Vitro IoT platform will open this purchasing model to multiple vertical markets, expanding demand for equipment manufacturers, operating companies, and customers.

Applicature

# Technical Specifications of Platform Functionality

As of today, there exists a variety of blockchain platforms that differ one from one another in terms of their features. Because Vitro aims to function as a global platform in the IoT industry, our main goal was to find the best matching platform capable of solving a number of relevant tasks.

Consequently, we decided to go with the Hyperledger permissioned blockchain.

## Permissioned Hyperledger Blockchain

Considering the fact that public blockchains and digital ledger technologies charge high transaction fees, Vitro was seeking a solution to incentivize its users in terms of providing a convenient environment for deal settlement and cost-efficient billing process procedures.

Permissioned Hyperledger is exactly the right choice, as this blockchain platform is known for its functionality and fully-designable nature. We have designed our smart contracts within the permissioned blockchain in order to provide IoT device access only to predefined parties, not everyone.

## HyperLedger Main/Root Ledger

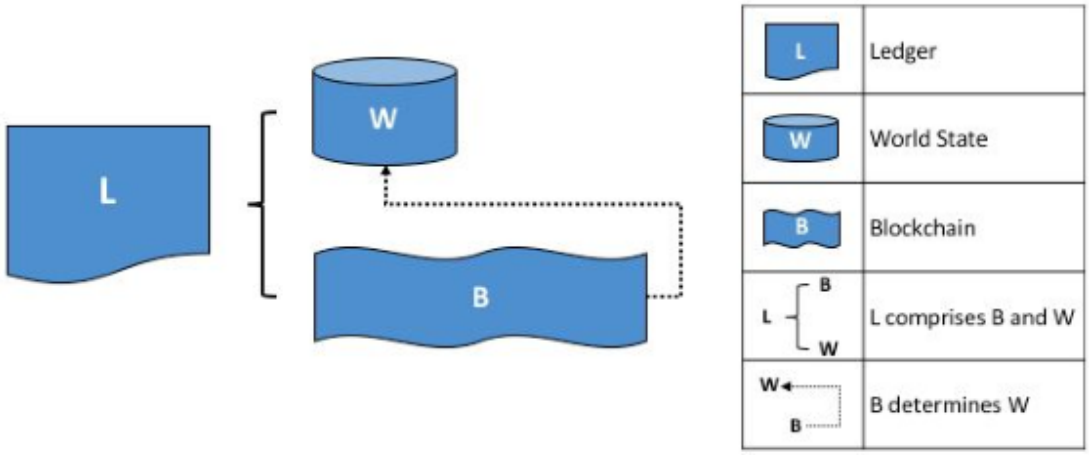
HyperLedger Root will serve Vitro as the mainnet. This root ledger will immutably store and provide users the records and data regarding IoT data consumption - securely, and in a sequenced manner.

State changes will appear as a result of transactions approved and validated by participating parties. Consequently, each transaction will result in a group of asset key pairs within the ledger: creates, updates, and deletes.

The root ledger will keep track of data and records, which will consequently determine the world state (contemporary states). As soon as transactions are written into a block, they'll join the root ledger and enable the user to view the entire change history.

The major aspect differentiating the main blockchain (root ledger) from the world state is data immutability. As soon as IoT consumption data is added to the root ledger, the opportunity to change or alter the data is eliminated.

Figure 2. HyperLedger structure




Within the root ledger, users will be able to access consumption data kept in a block. The block, consisting of its header, block data, and metadata, will outline all the relevant details according to the following structure:

Figure 3. HyperLedger data block structure

| | |
|---|---|
| **Block Header** | **Block number:** an integer starting at 0 (the genesis block), and increased by 1 for every new block appended to the blockchain. |
| | **Current block hash:** the hash of all the transactions contained in the current block. |
| | **Previous Block Hash:** a copy of the hash from the previous block in the blockchain. |
| **Block Data** | This section contains a list of transactions arranged in order. It is recorded when the block is created. These transactions have a rich but straightforward structure. |
| **Block Metadata** | This section contains the time when the block was written, as well as the certificate, public key, and signature of the block writer. Subsequently, the block committer also adds a valid/invalid indicator for every transaction. This information is not included in the hash, however, as that is created simultaneously with the block. |

Within the Hyperledger architecture flow:

- the administrator will interact with Decentralized Vitro UI comprised of the Angular framework
- the application will process user requests to the network through a REST API
- requests will be implemented to the blockchain state database on Hyperledger Fabric v1
- the REST API will be used to retrieve the state of the database
- the Angular framework will receive the data through GET calls to the REST API

## HyperLedger Chaincode

Chaincode is synonymous with smart contracts deployed within the HyperLedger blockchain. It is a piece of code that is deployed into a network of Hyperledger fabric peer nodes that enable interaction with that network's shared ledger.

When a transaction occurs within the network, it will trigger the chaincode, which will consequently define the state change that should be applied to the ledger. The chaincode implements logic on a predefined basis between participants within the blockchain network. However, chaincodes can provide much greater functionality than regular smart contracts. One

of the characteristic chaincode features is the fact that they are controlled by a peer according to the same interface. This interface is flexible enough that Hyperledger Fabric actually uses chaincodes to implement some system-level features. Every chaincode will be implemented as an isolated program maintaining its own private state on the ledger. Because they are essentially arbitrary codes, our chaincodes will conform to a common interface used by Vitro.

# How It Will Work

When addressing the above-mentioned problems of data tracking, auditing, and billing procedures, the Authentics blockchain relies on two core functions:

- **Smart-contract query:** delivery of payload of data within a specific date range. Data in each block will contain a summary of the full day by field. These fields will vary a great deal (type, number of fields) from instance to instance. The payload will be structured, and will include the various fields of IoT installation. Summary data will normally be delivered for transaction processing.
- **Audit query:** delivery of summary and raw data, organized by sensor and polling interval. Block size will be defined according to root ledger specifications.

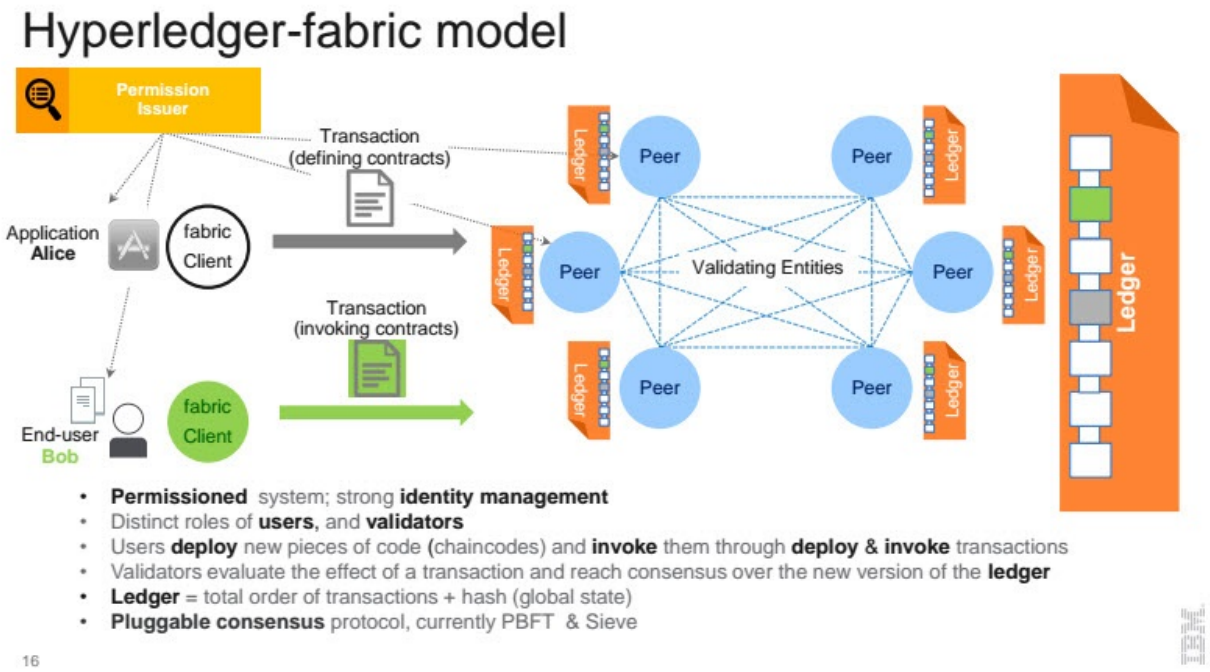
The secure crypto foundation of each Vitro Crystal can serve two roles: authentication and anchoring of blockchain transactions. Unlike mining for tokens in the Bitcoin model, each ECC chip generates its own unique private and public key pair. Unique is an understatement, as each key generated by the ECC chip creates a value 2^256 digits long. An integral part of this key generation is a random number, created in the ECC chip, which is mathematically unique. These elements build a key structure that is unbreakable to the extent of requiring an estimated U.S. $2 billion worth of computing power to break a single key.

## Blockchain Architecture Design

HyperLedger Fabric allows us to address solutions within the Vitro platform as well as carry out IoT data queries and audits. Fabric isn't considered to be a blockchain, but more of a framework, implemented in Go coding language. It is meant to enable consortium blockchains with different degrees of permissions.

To use resources more efficiently, Fabric has fewer nodes than the public chain. It also computes data massively in parallel, which makes Fabric's scale much better than public blockchains. Additionally, its basic architecture supports confidential data, giving its members more privacy than can be obtained on the public blockchain.

Figure 4. Hyperledger Fabric Model



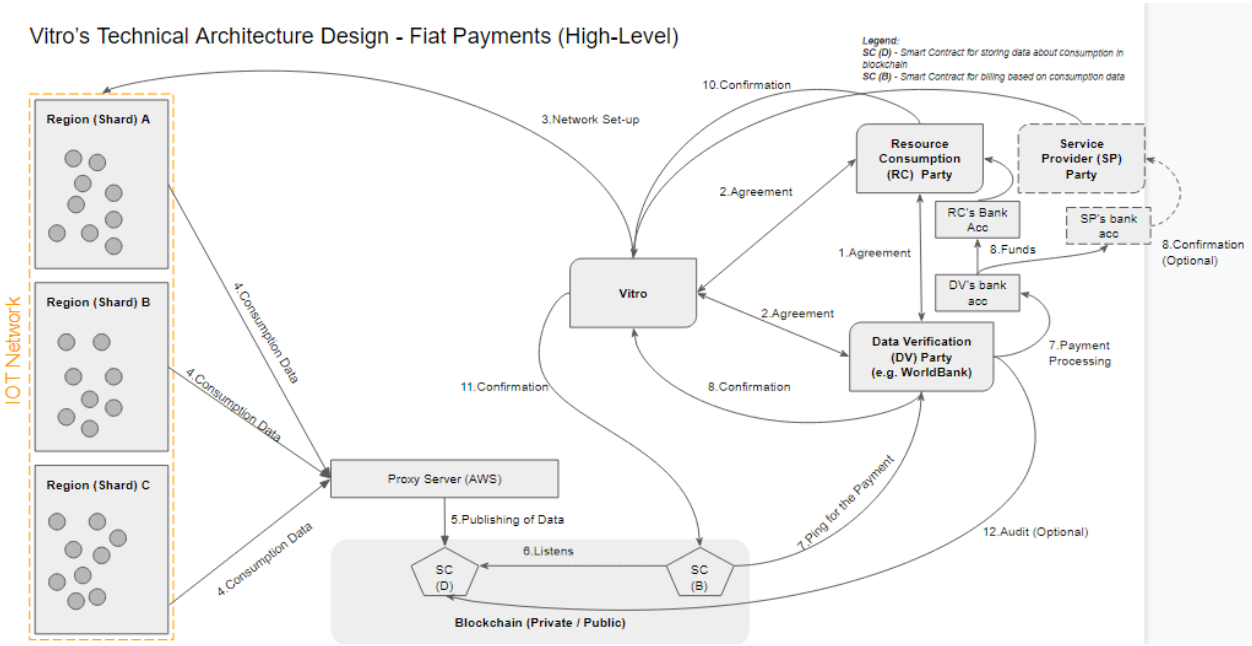

Fabric was one of the first to make separation between endorsers and consensus nodes a design principle. Endorsers have the state and build, validate, and propagate transactions and chaincode, while consensus nodes order pre-validated transactions.

Fabric doesn't implement native currency, allowing the designer to define asset parameters within the Fabric Composer.

Vitro is going to enable fiat payment processing within the ledger, which will provide users with enhanced convenience and ease of platform use.

Figure 5. Blockchain Technical Architecture Design

https://docs.google.com/presentation/d/18rlg0AxplLsQaOdQhe_9NuIYLyDRMZlbiETXvoO5Mmw/edit#slide=id.g4e95ab1b0e_0_2




Data requests, audits, and delivery will be enabled between parties via Fiat processing. For this, a resource consumption party (RC), data verification party (DV), and service provider (SP) will have to reach an agreement with Vitro technology.

As soon as the network is set up, consumption data pertaining to certain regions (placed in shards) will go through the proxy server (an AWS IoT) and get published to the Data smart contract (D).

A billing smart contract will listen for information from the data smart contract, then ping the data verification party for payment. As soon as the payment is processed through the data verification party's account, it will send funds to the service provider's bank account. These actions will be further confirmed by Vitro as well as all participating parties reflected in the billing process and ledger.

As an additional option, the data verification party can audit information through a data smart contract on the blockchain.

## HyperLedger Fabric Chaincode

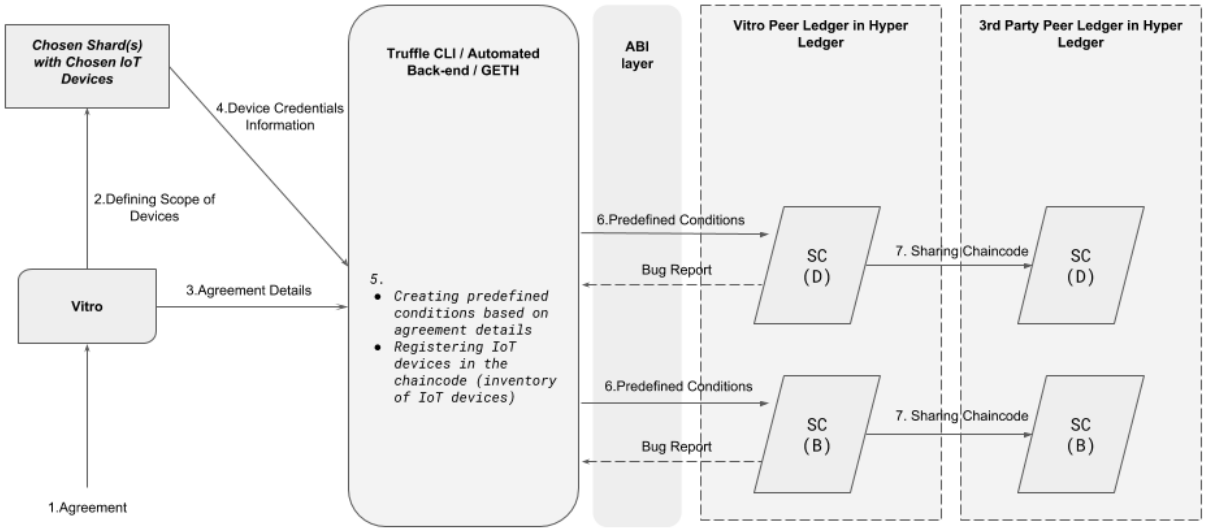
With the implementation of the Hyperledger Fabric blockchain, it will be possible to create predefined conditions based upon agreement details.

Figure 6. Vitro Smart-Contract Fabric

https://docs.google.com/presentation/d/18rlg0AxplLsQaOdQhe_9NuIYLyDRMZlbiETXvoO5Mmw/edit#slide=id.g4e95ab1b0e_0_175

Applicature

Vitro Smart-Contract Fabric




After an agreement has been reached between all participating parties, all IoT devices will be registered in HyperLedger chaincode. When the data is ready to be published on the blockchain ledger, our chaincode will validate that this data was received from a registered IoT device. If this is the case, the data will be published to the ledger.

In accordance with the agreement, Vitro will define the scope of the devices in terms of chosen IoT shards, then send the credentialed information to the truffle CLI/automated back-end/GETH, thereby creating predefined conditions within the agreement. This information will be processed through the billing and data smart contracts (B and D), which will consequently perform an audit for the bug report and deploy under the final conditions.

There exist certain cases in which IoT devices can get out of order. In situations like this, it is important to reflect these changes in the chaincode in order to stop the process of receiving updates from a broken device.

The goal of IoT is to structure it as a utility: a format in which the client is paying for services rendered. The structure for IoT is simple: the client is billed according to metered MQTT traffic. As sampling and polling is increased, it naturally costs more, but delivers more granular data.

The data format proposal:

- Daily block: IoT data is a time series arranged by a given polling interval. Given the asynchronous nature of IoT (latency, lost MQTT packets), AWS IoT will gather data for

presentation on dashboards for operational use, and build a time series that captures the granular operating data.

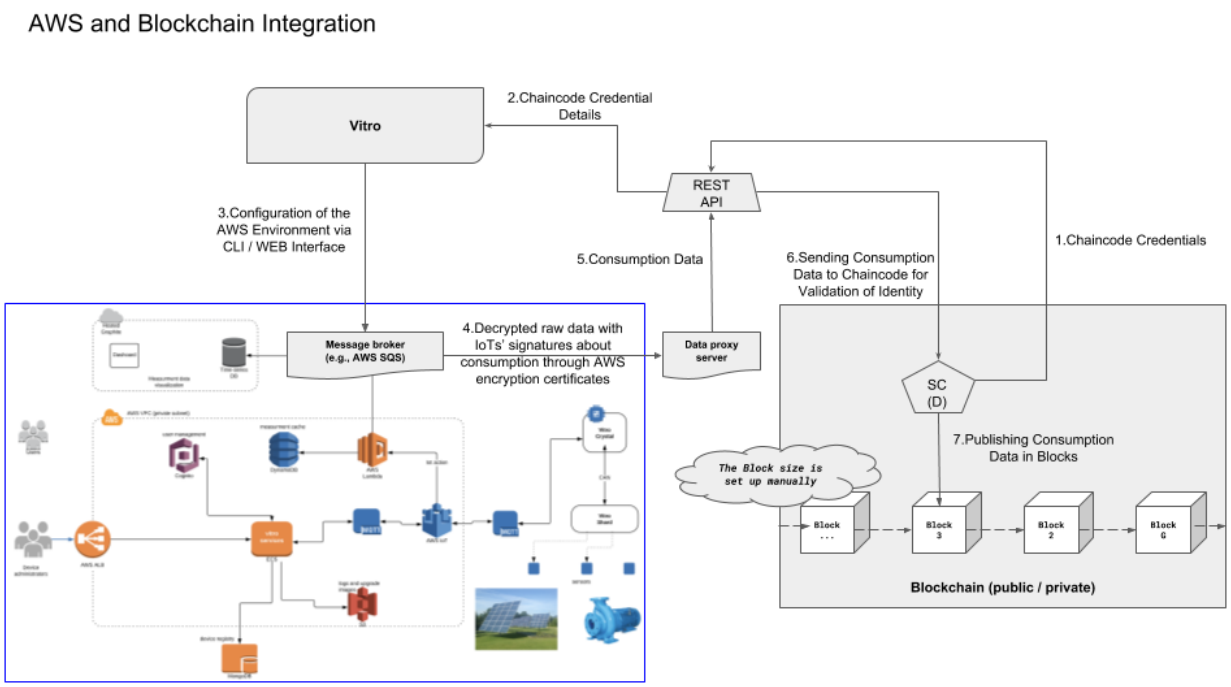
- Operating data will also be gathered on the Crystal IoT gateway and assembled into a time series. The Crystal Gateway will build a parallel time series to AWS IoT.
- Each day at 24:00, the Crystal will build a daily dataset for the block, then hash and sign the block with the local private key.
- The hashed Crystal block will be posted to the hosted Hyperledger blockchain.
- The Crystal block can also be checked against the data stored for that day on AWS.
- The AWS time-series summary will be cached infinitely. The data will be stored in the AWS IoT and recorded on the blockchain ledger.
- The timestamp feature will be used in the smart contract to synchronize the data-tracking process. It will function according to reconciliation of the previous day, and provide the data every 60 minutes to keep the system alive.

## AWS and Blockchain Integration

With the implementation of blockchain in the existing AWS structure, the processes of data collection, monitoring, audits, and exchange will be verified and transparent.

Figure 7. AWS and Blockchain Integration

https://docs.google.com/presentation/d/18rlg0AxplLsQaOdQhe_9NuIYLyDRMZlbiETXvoO5Mmw/edit#slide=id.g4f8fab099d_0_27



After the deployment of smart contracts (chain code), their credentials will be used to set up an environment according to the agreement between parties (the consumption and validation parties). The credentials will be requested via REST API. Vitro will configure the AWS environment via CLI/WEB interface. When the environment is organized, data will go through the message broker after it has been decrypted into raw data with IoT signatures. This will reflect consumption with AWS encryption certificates. Decryption will be conducted through the data proxy server. The details regarding resource consumption and usage from IoT devices will be sent through the REST API to the data smart contract for consumption (chaincode that contains predefined conditions) and finally published to the blockchain ledger.

Block size will be controlled with BatchTimeout and BatchSize parameters. In addition, it is important to note that this will be a use case dependent on transaction size, i.e., the logic of the implemented chaincode.

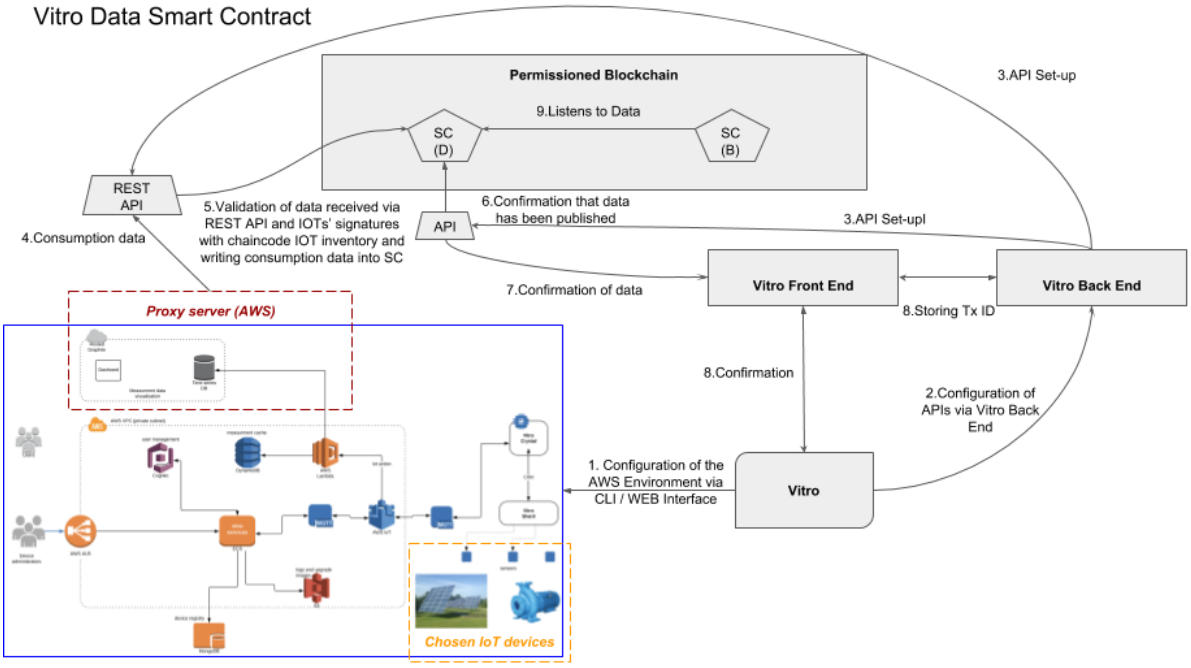
Fees for publishing and auditing data will be not be charged by the system, because the system will be hosted by Vitro. There is no gas for execution of of transactions, because HyperLedger does not support the fee concept, as in the case of Ethereum.

## Data Smart-Contract Architecture

The process of data interaction between the AWS proxy server and the Vitro application will be enabled by an data smart contract mentioned above.

Figure 8. Vitro Data Smart-Contract Chart

https://docs.google.com/presentation/d/18rlg0AxplLsQaOdQhe_9NuIYLyDRMZlbiETXvoO5Mmw/edit#slide=id.g4e95ab1b0e_0_214




First, Vitro will configure an AWS environment via CLI/WEB interface and send configuration to Vitro's back end for API configuration and deployment. API will be set up for data consumption from the AWS proxy server reflected in the REST API.

This data will be validated according to IoT signatures from the chaincode inventory, then written and verified in smart contracts, sending confirmation to Vitro's back end and front end before publishing. Transaction hashes will be stored within the front end and back end, and confirmed with Vitro.

Each Vitro Crystal device contains a unique crypto private-public key pair. This pair serves as a unique foundation for a blockchain constructed to record all transactions associated with each Vitro Crystal edge device.

Service providers can post their billings to the blockchain for each device being serviced and monitored.

The smart contract can audit the data and services provided to and from the remote edge device, in effect auditing each billing against the services provided while confirming checks and balances. Once each transaction has been audited, the smart contract associated with each Vitro Crystal edge device can authorize a payment transaction with fiat.

All transactions are recorded in blockchain records for each remote edge device, creating a closed ecosystem dedicated to secure control, operation, and transactions.

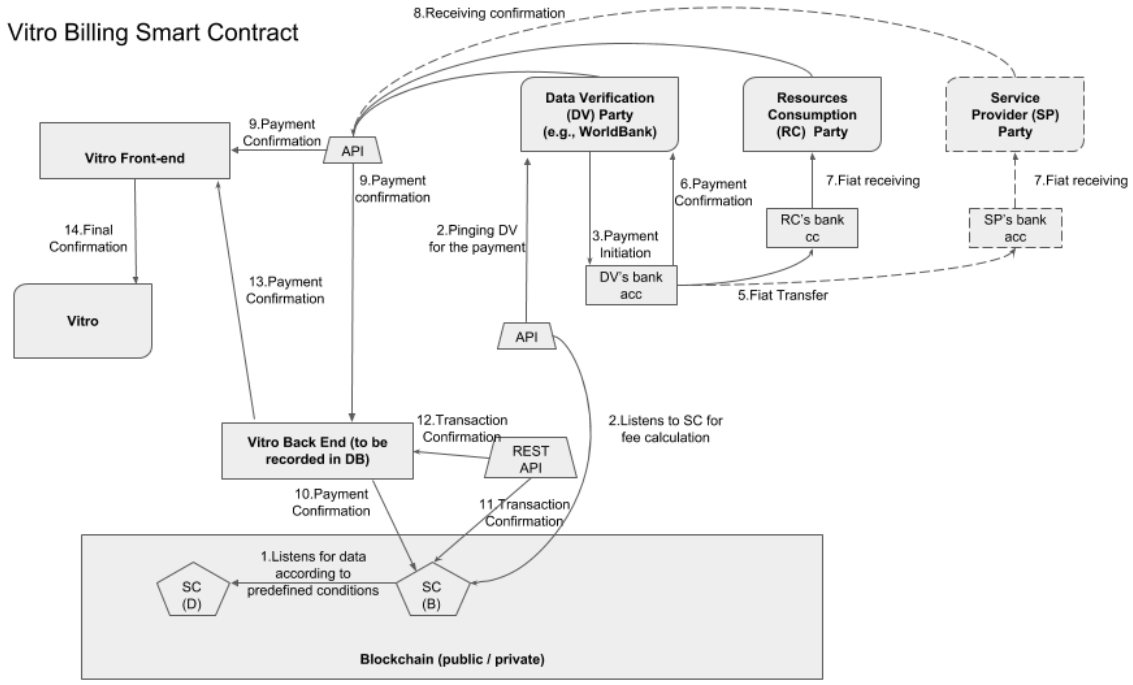
Applicature

## Billing Smart-Contract Architecture

With the use of a permissioned blockchain, we establish a reliable process of billing between participants. The main functionality of the Billing Smart Contract is calculation of the amount of payments to be covered by the Data Verification Party, off-chain and based on real consumption.

Figure 9. Vitro Billing Smart Contract

https://docs.google.com/presentation/d/18rlg0AxplLsQaOdQhe_9NuIYLyDRMZlbiETXvoO5Mmw/edit#slide=id.g4e95ab1b0e_0_183




A billing smart contract will listen for the data from a data smart contract within the blockchain environment according to the predefined conditions set up based upon the agreement. The information calculating the amount of payment will be translated to the DV Party via deployed API. Then, the DV Party will initiate the payment by pinging its bank account and transferring the payment (in fiat) to the account of the RC and SP parties. After receiving confirmation of payment, the RC and SPn Parties will get the funds, and further confirmation will be sent to Vitro's front- and back-end APIs. This will be reflected in the billing smart contract. Further, the billing smart contract will confirm transaction processing, with the Vitro back end and front end initiating the final confirmation.

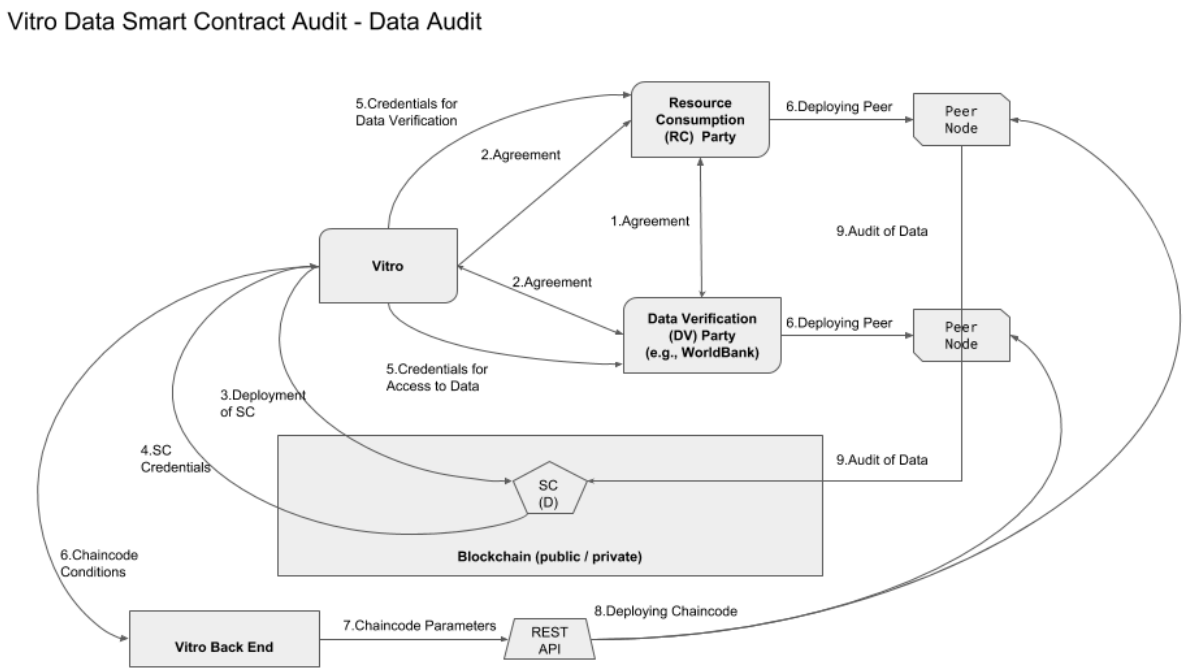
Applicature

## Data Audit Process

The use of an audit smart contract will ensure that the consumption party does not receive unverified information from an IoT device. In order to avoid a pay-and-pray scenario, the featured smart contract will execute only when all parties have reached an agreement according to the written conditions by validating the data in the chaincode.

Figure 10. Vitro Data Smart-Contract Audit

https://docs.google.com/presentation/d/18rlg0AxplLsQaOdQhe_9NuIYLyDRMZlbiETXvoO5Mmw/edit#slide=id.g4e95ab1b0e_0_187




Data will be audited based on an agreement between Vitro and the resource consumption and data verification parties. When the data smart contract is deployed, Vitro receives these credentials and deploys the REST API from the back end to deploy peer nodes to the DV and RC parties. When the peer node is deployed with shared chaincode on it, the DV and RC parties could, at any point in time, deep-dive into the chaincode and execute the audit.

## Timestamp for Data Publishing to the Blockchain and Main Principles

It is very important to have a clear mechanism for publishing data to the blockchain. One of the common problems in IoT data collection is its asynchronous nature. Very often, data gets out of order due to circumstances in edge device data collection, which, consequently, causes

significant latency. Because Vitro edge devices are capable of creating their own hashes for data that will be time-stamped with that hash, the most crucial problem is to collect data in a trusted manner and send it to a hosted blockchain.

It's important to keep all devices updated without overloading the system with data. To synchronize the record-tracking process, we are using a timestamp feature in a smart contract that will write data to a blockchain. This type of record will be fully complete with regard to the previous day. After a reconciliation period, each of the elements will be published to a block, but only after it is hashed and verified. In this way, we will provide trust in the finished block that goes to the blockchain.

Publication of consumption data to the blockchain will occur according to the following two principles:

- **Principle 1 - Publishing on Demand**

This principle is based on the rule that all data should be published on the blockchain as soon as it is gathered by the IoT device. In our case, when consumption of a particular resource (e.g., water) is tracked by the IoT device, the relevant data goes directly to the AWS Amazon database and is then published on HyperLedger.

This approach serves as a method of data backup, and at the same time, it allows us to keep the system on the safe side in terms of data latency. As soon as the data is aggregated, it will be signed and hashed. Only after these steps will it be published on the ledger, in accordance with the predefined conditions written in the HyperLedger chaincodes. Even though the synchronization will not be conducted in real time because of the latency problem, the data will be trusted. This is the most crucial point.

- **Principle 2 - Keeping the system alive**

Another principle is based on the rule that all devices must prove their live status. Even if a party isn't consuming resources, the device will still provide data about it (even if the data equals 0) to the AWS, and then to the blockchain every 60 minutes.

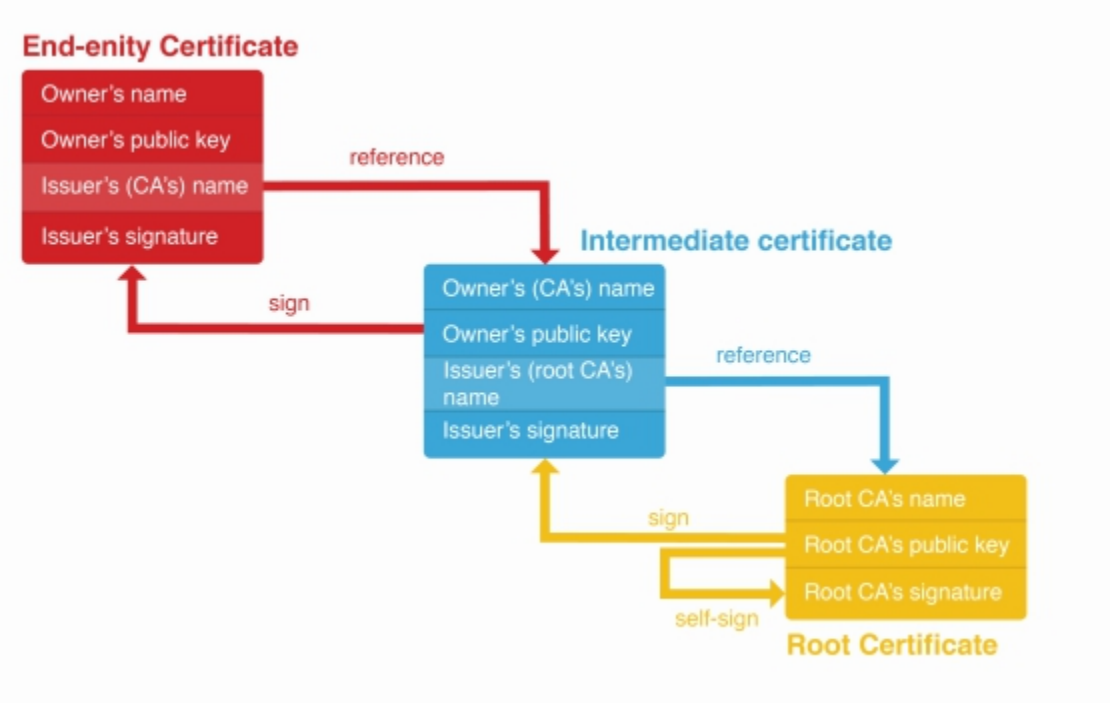
This approach ensures transparency and provides the user with the ability to track the current state of resource consumption. When the user sees that the latest updates were equal to 0, for instance, several times in a row, they will get the picture that these resources were not used instead of thinking that the data wasn't updated.

## Data Validation and Encryption Method

**Certificate Chain of Trust**

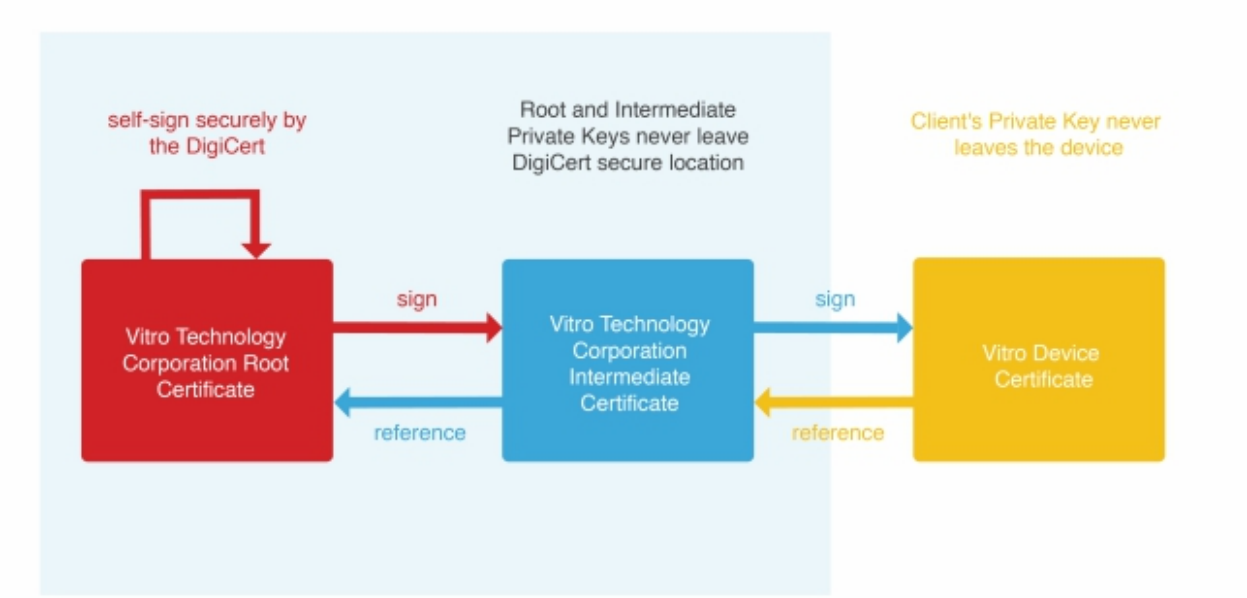
At the bottom of the Certificate Chain of Trust, there is a Root Certificate. This is a self- signed certificate issued by the trusted Certificate Authority. This Root Certificate can be used to sign the Intermediate Certificate; then the Intermediate Certificate can be used to sign the certificates of individual entities. This chain can be made longer by adding multiple Intermediate Certificates, signed one after the other.

Figure 11. Certificate Chain of Trust: Standard process flow




Our Certificate Chain of Trust:

Figure 12. Certificate Chain of Trust: Modernised by Vitro process flow

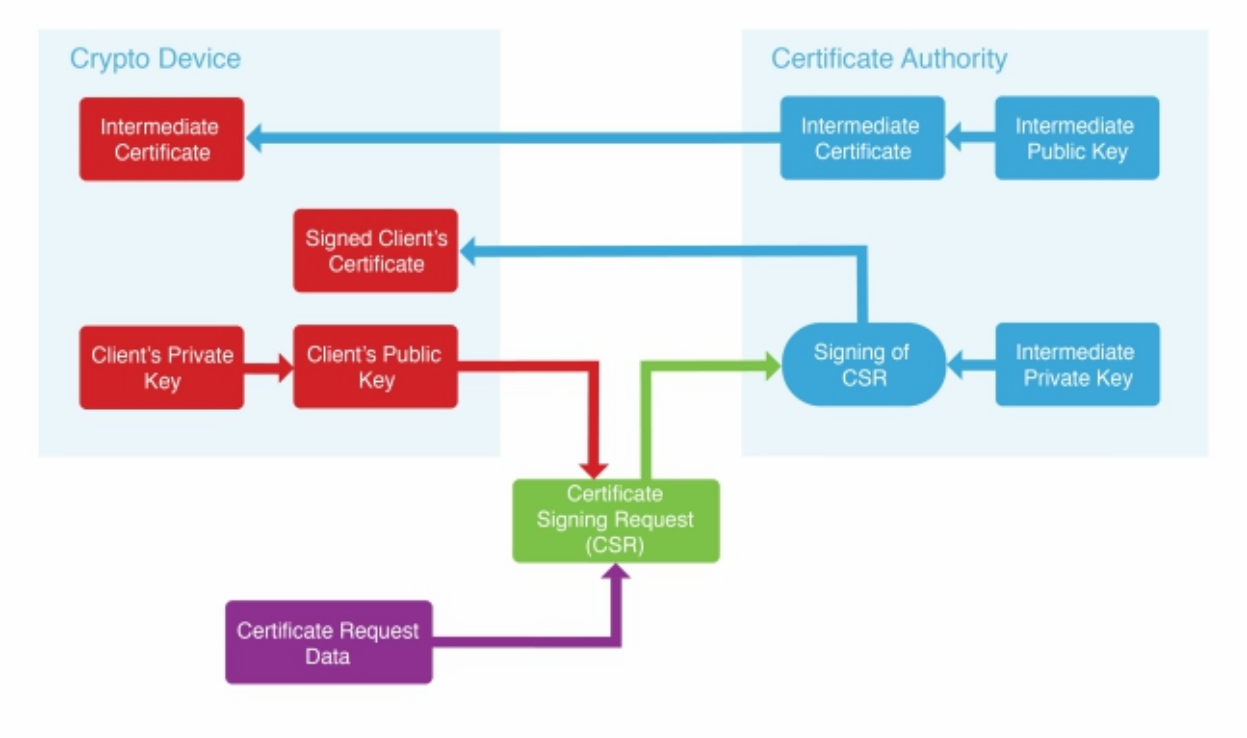



The process of data encryption is enabled by Amazon AWS IoT. As soon as IoT device data regarding resource consumption is collected, it is hashed and signed within AWS IoT.

We have chosen DigiCert, a world leader in the certification area, as our certificate provider. We are taking advantage of their experience and high security standards in terms of certificate generation and key storage. We've been assigned a custom ECC Root of Trust tailored to our own requirements. The DigiCert CertCentral solution received the Top Award for an IoT Security Solution. It has been proven secure, but at the same time easy to use and automate. Thanks to this solution, we are able to generate Client Certificates in a secure and automated manner through a single API call. After the acquisition of Symantec by DigiCert in October 2017, their market share in website SSL certificates exceeded 13 percent.

**Create Signing:**

Figure 13. Certificate Chain of Trust: Signature creation process




Signing the certificate consists of the following steps:

- Generation of a private/public key pair by the subject. The private key is a secret, and should never leave the subject device! The private key is the most important part of authentication. Basically, having an entity private key means being this entity. Storing keys on regular storage devices connected to the Internet may create the danger of leaking the key. In our case, the ECC private key is generated internally inside the Cryptographic Device. There is no way to extract the private key from it. The device has a cryptographic engine that performs a set of cryptographic operations in order to prove key possession to the outside world.
- Generation of the Certificate-Signing Request (CSR).
- Signing of the Certificate-Signing Request by the Certificate Authority, and thereby issuing a digital certificate. In the signing process, certificate data is hashed and signed with the issuer's (signer's) private key. The issuer confirms the information provided in the certificate and proves that the subject is the owner of the public/private key pair.

**Certificate Verification**

The following steps are required for Certificate Verification:

- The digital signature of the certificate is checked
- The certificate chain is checked
- Expiration and activation dates are checked

- Revocation status on the Certificate Revocation List is checked

**Why use ECC?**

Small ECC keys have the equivalent strength of larger RSA keys because of the algorithm used to generate them. For example, a 256-bit ECC key is equivalent to a 3072-bit RSA key. Because of the smaller key size with an ECC certificate, less data is transmitted from the server to the client during the SSL handshake. ECC certificates also require less CPU and memory, thereby increasing network performance and making a potentially significant difference for high-volume or high-traffic applications like IoT.

**How should the data secured by the certificates and blockchain be used?**

This raw data is processed by Amazon. It must first be decrypted, then go through the message broker and REST API before being written into the block of the HyperLedger root ledger. Before decryption of the data, the system checks for the signature of that particular IoT device, and compares it with the IoT inventory (Proof-of-Signature).

The key aspect to be specified is that device data won't go out of the AWS IoT due to the fact that blockchain peers (nodes) in our solution will be deployed on Amazon cloud services, as well (e.g., Amazon Managed Blockchain). The Amazon system itself uses encryption certificates that provide a security layer for the whole ecosystem. The process of sending decrypted data from the Amazon to HyperLedger will be secured by AWS.

**Providing Certificates for In-Transit Data Encryption with Amazon EMR Encryption**

To encrypt in-transit data, we will use a security configuration with the following options:

1. It is possible to manually create PEM certificates, include them in a zip file, and then reference the zip file in Amazon S3.
2. It is possible to implement a custom certificate provider as a Java class. It is crucial to specify the JAR file of the application in Amazon S3, then provide the full class name of the provider as declared in the application.

Amazon EMR will automatically download artifacts to each node in the cluster, and later use them to implement open-source, in-transit encryption features[3].

[3] https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-encryption-enable.html