MARCH 8, 2019

akaChain

WHITEPAPER V2.0.

AKC TEAM

Contents

1. Terms and abbreviations

#	Term/Abbr	Description
1	AKC	AKACHAIN
2	HPL	Hyperledger Fabric
3	TCO	Total cost of ownership

2. Introduction

2.1 Akachain

Akachain (AKC) is a ***permissioned business blockchain*** based on HyperLedger, one of the most popular blockchain project in the world.

Akachain's mission is to enable enterprise private transactions. With smart contract, it's never a problem to manage complex agreements cross entities over a variety of assets.

Compare to state-of-the-art blockchain systems, Akachain is designed for better privacy, integrity, scalability, governance capabilities and TCO. Akachain provides off-the-shelf software infrastructure which need for commercial uses from enterprise perspective.
Akachain minimizes the effort to integrate with existing enterprise IT architecture.

Our goal is to become a global ecosystem with hundreds of blockchain token built on Akachain for different business purposes from finance, retail, manufacturing, insurances, health services to government and other public sectors.

Additionally, Akachain provides blockchain-as-a-service that helps anyone from developers, entrepreneurs, retailers to large enterprises to issue easily their own tokens to conduct their business in the token economy.
For examples, an organization will be able to raise fund from trusted investors with Akachain to conduct fair and reliable crowdfunding. They can use the same platform to tokenize various products such as loyalty points for consumers.

2.2 Advantages

AKC is a **consortium blockchain system**. Such architecture offers a number of advantages over a traditional centralized, cloud-base system.

- AKC now provides **multiple transparent, immutable “ledgers”** that record transactions among enterprises.
- AKC blockchain system allows **complex business models** that requires **strong consistency, high security and data transparency**.
- AKC **decentralized** nature eliminates system single point of failure, both in term of service availability as well as security compromising.
- AKC is a **privacy-preserving** system, meaning that customer and enterprise private data does not leak to other parties without authorization.

And many other benefits.

3. High level architecture

3.1 Actors

We introduce 3 majors entities/actors that involve in the Akachain ecosystem, namely the host, the vendor and end users. They cover all interest aspects from those who operate the blockchain and essential services (the host), those who directly use the system (the vendor) and the end user that do not directly use the blockchain but benefits from the result of it.

1. **Host**: refers to entities who provide infrastructure and resources to deploy, monitor and maintain the Akachain system.
2. **Vendors**: Organizations, enterprises or individual companies that want to participate in the system to use blockchain in their services.
3. **End-User**: Developer, Contributor, who want to build / develop DApp in Akachain system.

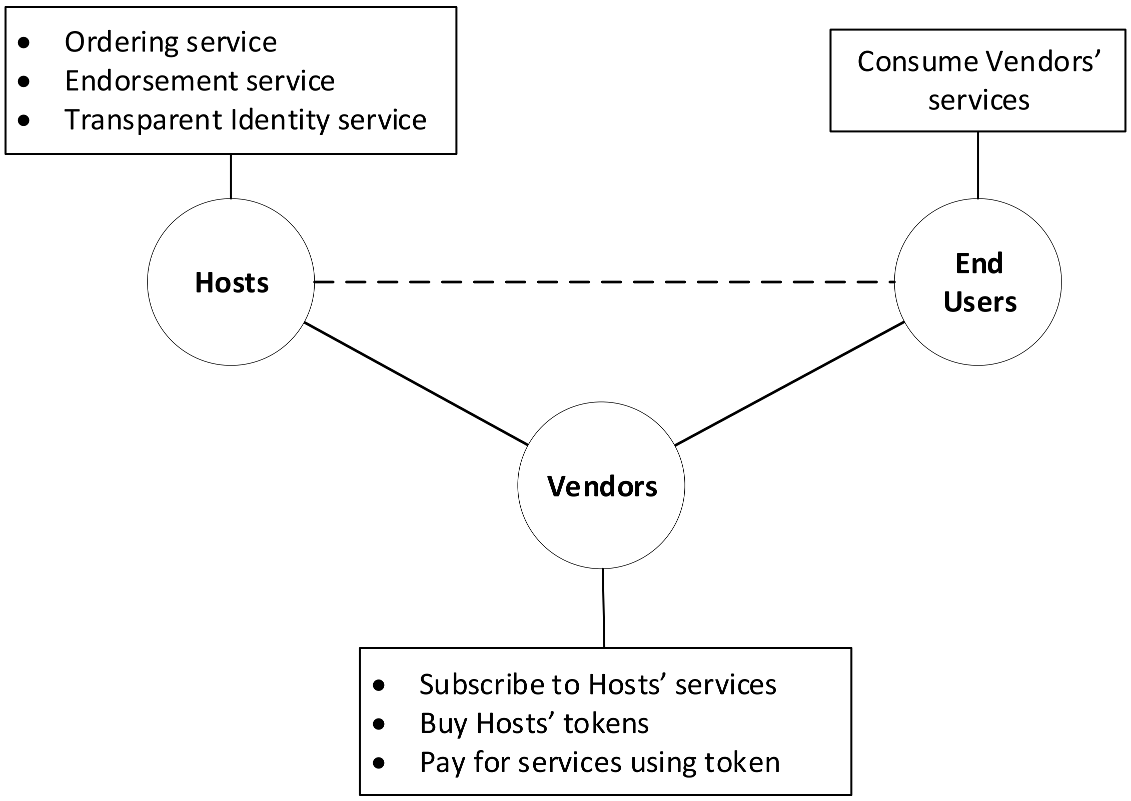

Ordering service
Endorsement service
Transparent Identity service
Consume Vendors' services
Hosts
End Users
Vendors
Subscribe to Hosts' services
Buy Hosts' tokens
Pay for services using token

Figure 1 – Actors relationship and sample available actions for each actors

Figure 1 shows the relationship between the 3 actors. The Hosts provide AKC infrastructure such as the ordering service to deliver messages in total order. It also provides essential services to the ecosystem such as the transparent identity service, the endorsement service and any other add-on service. We will explain in detail those components in the following sections.

The vendors, being the main target users of AKC, subscribe to the hosts' services by buying tokens issued by the hosts. Tokens are simply cryptocurrency token similar to Bitcoin or Ethereum with the sole purpose of paying the hosts for their services. A hosts can also simply offers subscription services and allow the vendor to pay them in standard currencies as well. However, the private token issued by the hosts are not affected by the trading market. Thus, vendors do not have to worry about sudden price change in their service fee.

Finally, we have the end users who consume vendors' services. Normally, there is little the blockchain can offer to the end users as most of the time the end users only connect to the vendor service backend. However, as the backbone of the system are connected by the blockchain, it is possible for the hosts to provide extra services to end users by letting them query directly the blockchain without going through vendor services. We can think of this as a transparent auditing service as the end users now can double verify their data downloaded from the vendors are actually existing on the blockchain. Yet, this is optional and depend on the actual implementation of the blockchain. Thus, we are only presenting the feasibility of such idea.

3.2 Transparent Identity services

Similar to Hyperledger Fabric, each peer in AKC is represented by a digital certificate in X.509 standard. Every message that a peer sends out must be signed by his private key so that others can validate using his public certificate. Fabric proposes to use a PKI model with a centralized Certificate Authority (CA) to issue certificate for participants. Here, we propose that AKC can use a hybrid transparent solution that combine CA with Trusternity like architecture.

Figure 3 presents the architecture of AKC Transparent Identity Services (TIS). The Host provides a Root Certificate Authority server to accept registration and issue certificate for other peers. It is noted that there can be multiple Root certificate from different hosts. However, their certificate must be cross signed to guarantee authentication. Similar to a Trusternity server, the Root CA database is organized into a Merkle Tree data structure and the CA software publish its signed tree root every epoch to the AKC blockchain. Alternatively, we can also implement the CA database on the AKC blockchain in a chaincode as well similar to EthIKS. The reason is that there are not a large number of peer in the network as in the case of Trusternity.

Similarly, in case there are vendors who want to run intermediate CA to issue certificates for its peer, the intermediate CA also follows the same procedure as the root CA. The intermediate CA either publish its STR every epoch to the chain or use the blockchain to store its issued certificates.

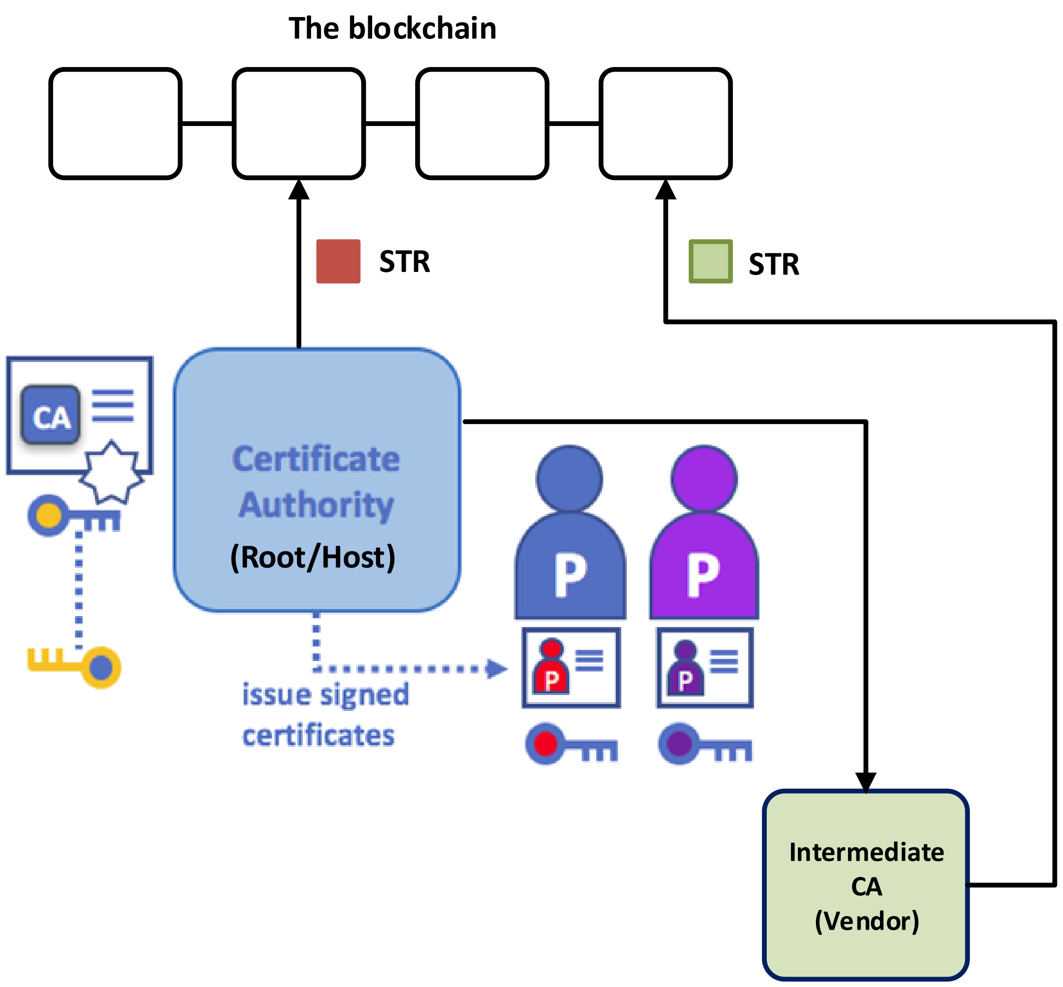


Figure 2 - Transparent Identity Services using PKI and blockchain

3.3 High level Blockchain design

At a high level, AKC is a hybrid, permissioned, multi-chain solution. We depict a sample setup as follows.

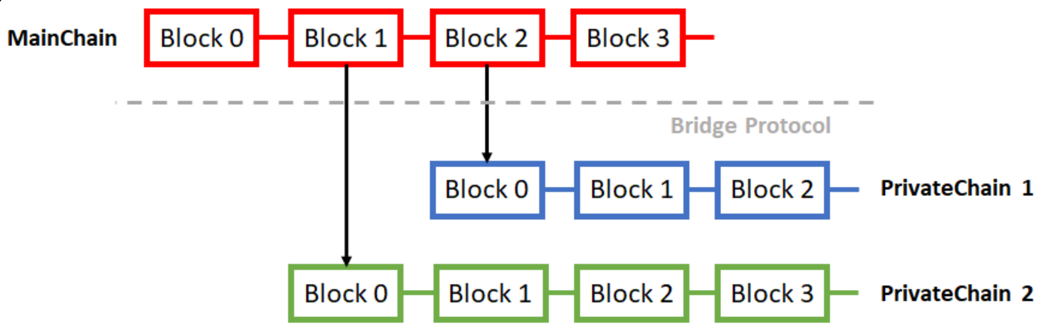


Figure 3 – Blockchain design

There is **one Public MainChain** that functions as a normal Public Blockchain. Mainchain is the core of AKC, it links every transaction from every parts of the system together to forge an immutable ledger. There are **multiple PrivateChains** that serve as private channels for different DApp *consortiums*. Here, we interpret a consortium as a group of Enterprise who build their own DApp.
We introduce a **Privacy-Preserving Bridge Protocol** to link all PrivateChains to MainChain. The intuition is that, while private transactions details are distributed securely over PrivateChain channels, the Bridge Protocol abstracts those details and create cryptographic verifiable proofs on

both chains to ensure system transparency and security.

3.4 Eco system

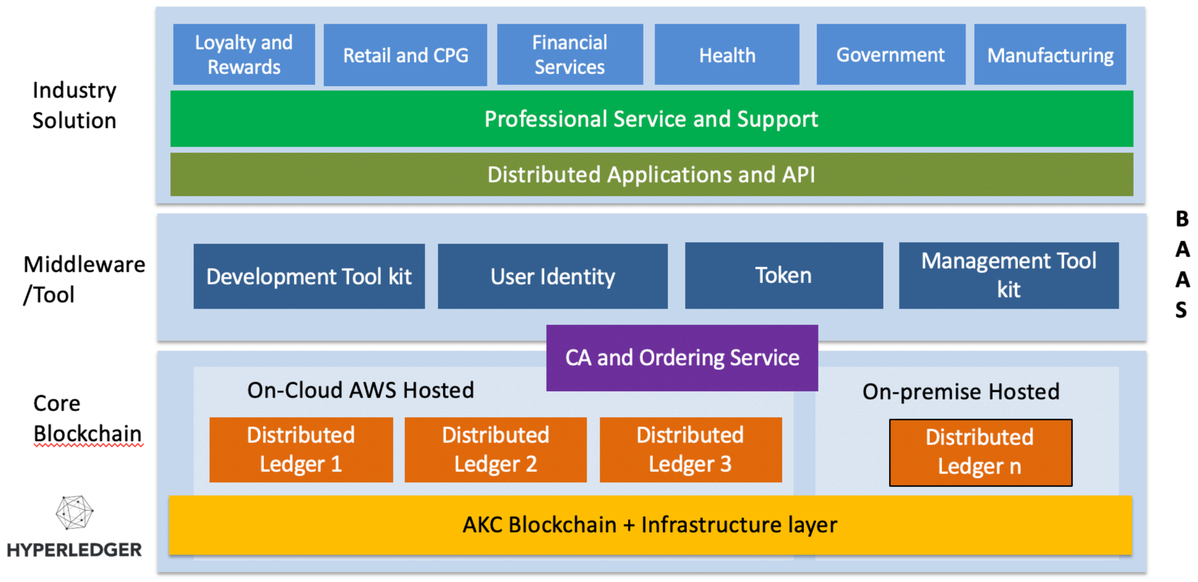


Figure 4 - ECO System

We now show a rich picture briefly describing components of the AKC eco system.

The host is depicted as a cloud center that provides backbone Hyperledger services

- Ordering Service: Instead of traditional P2P communication like Bitcoin or Ethereum, Hyperledger Fabric offers a *centralized* ordering communication. All peers/clients send messages to the ordering service and
- CA Service: all Merchant peers are required to obtain valid certificates from AKC Certificate Authority service in order to participate in the system. This requirement, however, does not apply for investor on MainChain.
- AKC Peers: As a part of the blockchain eco system itself, AKC Peers participate in both MainChain and PrivateChain. AKC peers take part in the consensus and policy enforcement of the chains.
- Explorer: The host provides a web portal allowing enterprise connect and view all transactions in their channel.
- Smart-contract: AKC allow customer to develop, deploy their smartcontract.
- Pre-built Middleware: AKC provide the pre-built layer as SaaS for enterprise.

3.5 Infrastructure

We designed AKC infrastructure based on AWS cloud. This architecture can make AKC high availability and ready for 2000 transactions/second. Below is the diagram:

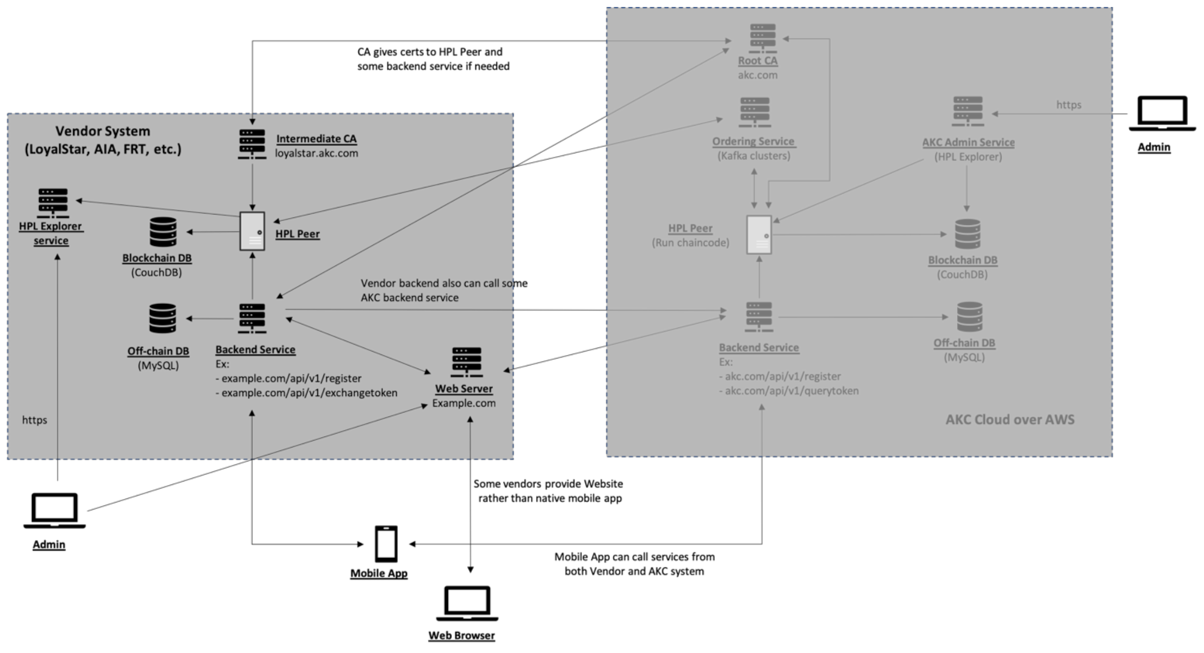

CA gives certs to HPL Peer and some backend service if needed
Vendor System (LoyalStar, AIA, FRT, etc.)
Intermediate CA
loyalstar.akc.com
HPL Explorer service
HPL Peer
Blockchain DB (CouchDB)
Vendor backend also can call some AKC backend service
Off-chain DB (MySQL)
Backend Service
Ex:
- example.com/api/v1/register
- example.com/api/v1/exchangetoken
Web Server
Example.com
https
Admin
Some vendors provide Website rather than native mobile app
Mobile App
Mobile App can call services from both Vendor and AKC system
Web Browser
Root CA
akc.com
Ordering Service (Kafka clusters)
AKC Admin Service (HPL Explorer)
https
Admin
HPL Peer (Run chaincode)
Blockchain DB (CouchDB)
Backend Service
Ex:
- akc.com/api/v1/register
- akc.com/api/v1/querytoken
Off-chain DB (MySQL)
AKC Cloud over AWS

Figure 5 - Infrastructure

3.6 Why not Bitcoin or Ethereum?

We design AKC mainly based on a hybrid model. We are not currently considering building the system on an existing public blockchain like Bitcoin. There are several reasons:

- Public blockchain is expensive: Bitcoin and Ethereum transaction cost is extremely high, even with the promise of "lightning network" that is still on paper.
- Public blockchain is slow: The cryptokitties event shows that Ethereum is not ready for high amount of transactions. AKC must be able to handle thousands transactions per day.
- Public blockchain means the system depends on public community. We must at least be able to control the system with regards to security, software update or network data flow.
- Privacy: Putting something on public blockchain means it is replicated across all clients. While there are ways to abstract away the detail via encryption/hashes, it is not straightforward and may put customer/merchants at risk of leaking private data.
- Legal: Cryptocurrency is rumored to be considered illegal in Vietnam. I'm not sure how DApp on public blockchain will be treated.

While a private blockchain solves all those issues, a private chain cannot be used as a public crypto currency - a vision that AKC wants to pursue.
For those reasons, we propose to use a hybrid architecture of **partially decentralized consortium blockchain**. An excellent reading of this matter can be founded here

3.7 Why Hyperledger?

There are several popular consortium blockchain platforms such as Corda, Hyperledger, BigchainDB, etc. However, many popular one that we can find on the internet is quite immature and definitely not ready for production.
Hyperledger is definitely the most active blockchain for business platform. It is heavily backed by numerous top enterprises such as IBM, Microsoft. Whether Hyperledger is ready for production or not is still a question since we have not seen any successful use-case with the technology. Yet, it is our current safest choice for the job.

4. MainChain

4.1 Peers

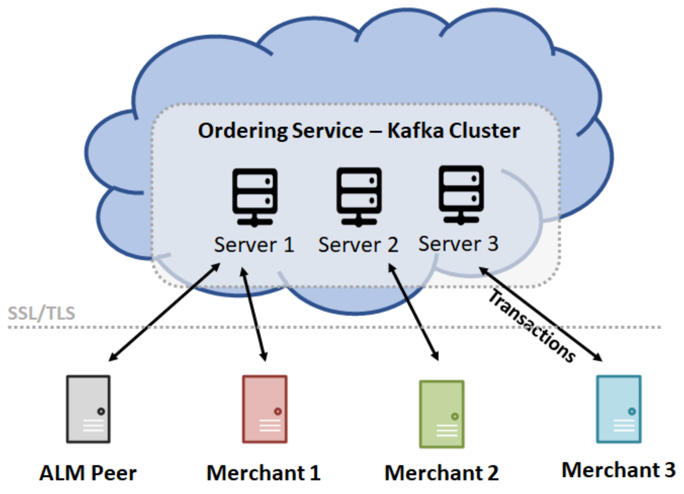


Figure 6 - Peers design

As a HPL blockchain, MainChain consists of ***Peers***. Unlike BitCoin or Ethereum, HPL peers connect to a **centralized Ordering Service** to exchange messages/transactions. In particular, our ordering service uses Apache Kafka to enable seamless, high throughput, real-time data exchange among peers. Kafka has built-in **crash fault-tolerant** achieved by internal Raft consensus, **extremely fast** and **horizontally scalable**. Thus, the architect scales well to millions transaction exchange asynchronously.
MainChain peer composition includes the **host peer** and **all merchants**. Blockchain data is then replicated over those peers via the ordering service.

4.2 AKCoin

We introduce ***AKCoin***, an internal crypto-currency to use inside MainChain. AKCoin is used by merchants to pay "transaction fee" in their corresponding PrivateChain. Obviously, a traditional method is to charge merchants to pay-per-transaction or apply subscription model, similar to paying for computing power on cloud service provider.

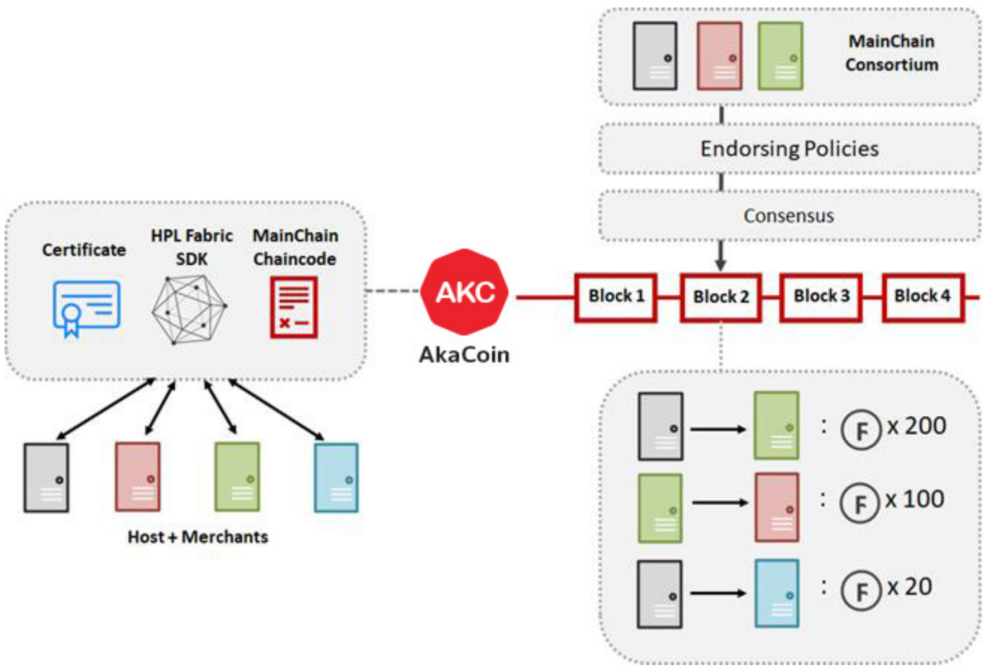


Figure 7 - AKCoin design

However, as a blockchain system, AKC allows merchants not only to *use* the system, but also *contribute* back to the system by *validate* and *endorse* other merchants' transactions. Such unique property incentivizes merchant to participate into AKC system, at the same time, increase security and trust among merchants as a transaction can be validate/endorsed by different parties. In that direction, we propose 2 ways of using AKCoin.

1 When a Enterprise creates a transaction to another Enterprise, he must pay a pre-defined amount of AKCoin. This is also called ***Transaction Fee***. This mandatory fee is what makes AKCoin has real value to the system. It also prevents merchants to flood the network with unlimited micro transactions.

2 When a merchant participates in the consensus/endorsing process of a PrivateChain or MainChain, he should be rewarded with a certain amount of AKCoin. This is also called ***Endorsing Reward***. The rate of consensus reward should depend on how much effort the participant spends in the process.

4.3 Consensus

Before going into detail, an excellent reading on consensus on Hyperledger 1.0 is available here. We now briefly describe how our system perform consensus.
First, as we currently use Hyperledger Fabric, there are 3 phases: Endorsement, Ordering and Validation.

1 **Endorsement** is driven by policy (e.g. m out of n signatures) upon which participants (host and merchants) endorse a transaction.

2 **Ordering** phase accepts the endorsed transactions and agrees to the order to be committed to the ledger.

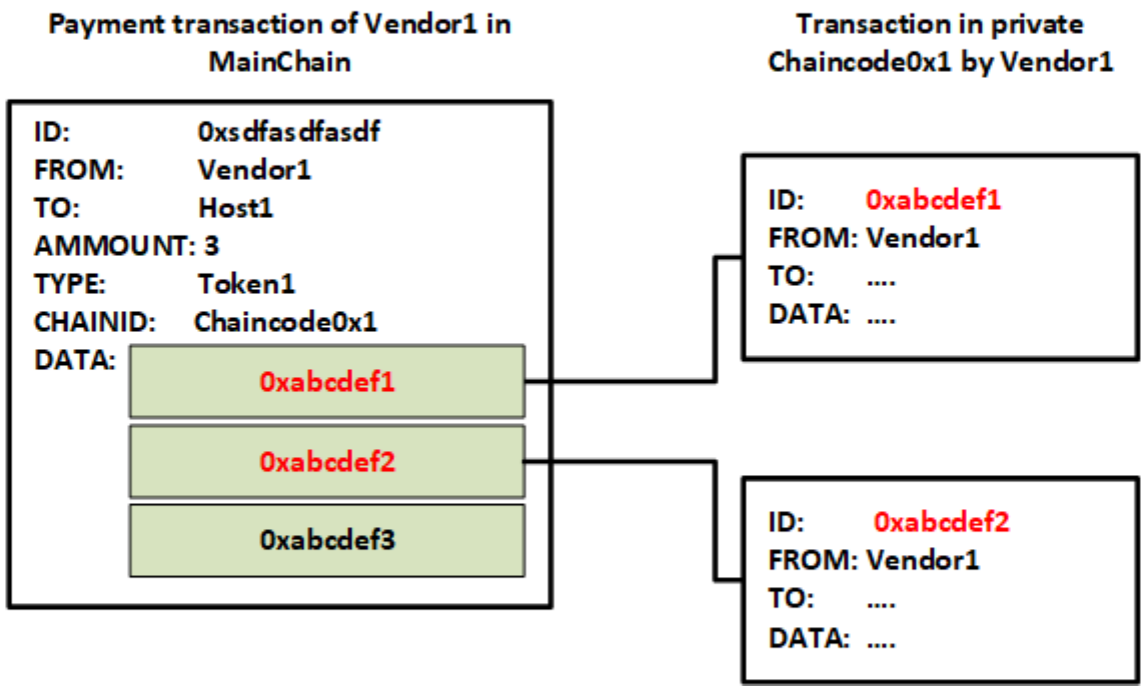
3 **Validation** takes a block of ordered transactions and validates the correctness of the results, including checking endorsement policy and double-spending.

Currently, as an out-of-the-box solution from Fabric, MainChain uses Raft as the underlying consensus algorithm. The Hyperledger Architecture paper referenced above provides excellent insight on advantages and disadvantages of different consensus algorithm. In this case, Raft is extremely fast and crash-fault tolerance whilst not be able to tolerant Byzantine behavior. In the future, further improvement towards Byzantine tolerant can be implemented. For example, BFT-SMART is a great reference that we want to follow in the future.

5. Bridge Protocol

We require a merchant to pay a small amount of AKCoin in order to issue transactions in a private chain. This fee is fixed (exact numbers are TBD) and the same for all transactions. Since AKCoin only circulates in the MainChain, the merchant needs to pay the fee in the main chain before it can issue transactions in the private chain. Below we will describe the detailed processes of how a merchant can issue a transaction in a private chain and how a peer can verify the transaction.

5.1 Transaction issuance



AKC require a vendor to pay a small amount of token in order to issue transactions in a private chain. Since the token only circulates in the MainChain, the vendor needs to pay the fee in MainChain before it can issue transactions in a PrivateChain. Below we will describe the detailed processes of how a vendor can issue a transaction in a PrivateChain and how a peer can verify the transaction.

Let us consider a scenario where Vendor1 wants to create some transactions on PrivateChain with id Chaincode0x1. Vendor1 must follow the following procedure:

1. The Vendor first issues a *payment transaction* in the main chain, which sends *fee* **n* Token1 coin to the issuing Host. Now, this is to simulate the real life process when a person pay money to the shop keeper. However, actual implementation depends on the application developer. One can reverse such process by allowing the Vendor to pay by real currency via a payment gateway and issuing a transaction from the Host to the Vendor rather than following this pattern. However, as long as we have a transaction that contains a certain amount of Token between Vendor1 and the Host, it is enough to fulfill this step.

2. In the Payment transaction, the vendor and the host must declare those following information:
    a. The PrivateChain id: Chaincode0x1
    b. The Id of the vendor: Vendor1
    c. The type and amount of paid token
    d. The Transaction data that includes arbitrary meta data depending on the application developers and, most importantly, a list of unique *ID hashes.* Those ID hashes are the transaction ID that the Vendor can use in the future to create transactions.
3. After the payment transaction has been accepted to MainChain. The Vendor can use the ID Hashes in the data field to create transaction on the corresponding PrivateChain.

We now present the procedure for a peer to accept a transaction issued by the peer on a PrivateChain. The key part is in the endorsement process. In Hyperledger Fabric, when a peer send a transaction to an endorsement peer to get their endorsement, the endorsement peer just simply executes the transaction to check its validity and sends back the signed data blob. Now, AKC bridge protocol requires that when an endorsement peer receive an endorsement proposal, the peer must check if the transaction ID is actually exist in a payment transaction of the MainChain for that particular PrivateChain. It the result is positive, the endorsement peer sign the proposal bundle and return, else it ignores the proposal. This simple process allows us to link any transaction on a private chain with a data field of a payment transaction. Thus, making sure that the sender pay fee for the host.
A vendor can also try to perform double spending attack on the bridge protocol by reusing the hash ID for multiple transactions in PrivateChain. Basically, he can send the different transaction with the same ID to different endorsement peers. This is acceptable as the peers are not aware of each other works and gladly endorse for those double spending transactions. However, when all signed transactions are sent to the ordering service and redistributed to other peers in the network, they can quickly detect the attack since the transaction ID cannot be reuse for one PrivateChain. Thus, we obtain an undeniable of a peer who try to perform double spending and can easily blacklist them from doing so in the future.
We can also see here that other than endorsement peers, other peers in Mainchain cannot not learn anything about the nature of transaction in the particular PrivateChain that they do not have access to. This type of bridge protocol is mainly address to the payment of service rather than actually linking transactions in the PrivateChain. However, it can be extend in the future with more advance cryptographic protocol to enable more functionalities such as allowing cross PrivateChain transaction without knowing the detail of each other’s.

6. Transaction verification

To verify that a merchant has paid for a transaction, a peer needs to do the following checks:

1 There exists a payment transaction in the main chain with the ID included in the transaction.
2 The payment transaction must contain exactly the public key of the merchant and the ID of the private chain.
3 Assuming that the payment transaction contains enough fee to pay for *n* transactions, there are less than *n* transactions with the same payment ID in the private chain.

7. Open Blockchain as a service

Finally, we present an open architecture of an open blockchain as a service model that encourage service providers to participate as the Host. Through our model, we show how multiple hosts can work together to share the workload and benefit from the system whilst increase the decentralized trust in the AKC system as a whole. Let us consider a service providers δ who wants to be a host of AKC. We propose a blockchain as a service delivery model for δ as follows.

- **Ordering service cluster**: AKC use Apache Kafka as the ordering service. δ can simply host an Apache Kafka cluster to share the workload with other hosts to provides the ordering service for AKC. When a vendor pay the transaction fee in MainChain, δ can receive a cut from the fee depending on how much δ provide for the ordering service. For example, if there are 2 hosts α and δ each provide 1 Kafka cluster for the ordering service, the transaction fee for ordering service will be divided in equal for both partners.

- **Transparent Identity Service:** Each host can provide his own TIS server where vendors can register. However, they must agreed with each other and cross sign their root certificate. The fee for TIS is then collected separately.

- **MainChain Token:** δ can provide payment method for Vendor to buy token on Mainchain to use in their PrivateChain. We are aware that the Token is mainly use in the endorsement process. Thus, if a vendor purchase Token from δ, the endorsement policy of the PrivateChain must also include δ in the required endorsement list. δ can also gives Token as a rewards for other hosts or vendors who participate in the endorsement process.

- **Automation tools**: δ can provides automatic tools for vendor to upload and instantiate Chaincode on their PrivateChain. This part can be considered as an additional service that is unrelated to the blockchain data itself.

- **Others**: monitor the network, anomaly detection, development tools, etc.

Another question that may arises is that does the role of third party service provider is necessary, given that a group of merchant can just simple host their own Hyperledger Fabric network (e.g. remove MainChain, only run PrivateChain). While it is certainly more straightforward, it requires more effort from merchant to learn and monitor the blockchain by themselves. It also remove the benefit of any future enhancement in the bridge protocol such as allowing cross-domain reference transaction using MainChain.

8. High-level Security Assessment

This section provides a high level security assessment of AKC eco system as a whole. Security is the utmost important characteristic that we want to preserve. However, AKC is an eco-system with a lot of software components and actors. While each software component deserves thorough security analysis, this page only provides a high level analysis of the system as a whole.
We analyses the system using a general classification of threats towards computer system. Each class of threat is considered with all AKC components.

8.1 Cryptographic Operations

AKC uses industrial standard (PKCS11) cryptographic algorithms throughout every component in the system. Without further configuration, we use those default algorithms:

- Each Peer possess a X.509 certificate.
- Asymmetric Crypto keys: ECDSA
- Symmetric Crypto keys: AES-256 in GCM mode
- Secure hash: SHA-3
- Key derivation function from password: PBKDF2
- Communications between any 2 parties are always encrypted in the transport layer using TLS

8.2 Spoofing of user identity

We want to prevent an adversary to impersonate something or someone else in the eco system. We propose several methods:

1 **AKC Organizatio**n: All communication with AKC Backend Service from the Internet use HTTPS. Thus, adversary cannot spoof himself as AKC.
2 **Hyperledger Peer**: All Hyperledger Peers have their own X509 certificates signed by AKC Certificate Authority, which in turn is signed with PKI. Thus, outside adversary cannot spoof their identity without compromising the CA service.
3 **Merchant Backend Service**: use the same certificate as Hyperledger Peer.
4 **Mobile Authentication**: A customer mobile client stores user credentials to quickly gain access to remote back end service. For remote authentication, we provide OAuth2 compatible authentication service. Customer can login to AKC service using either the combination of email/password or using social network (Facebook/Google) account.
5 **Mobile Access Token**: We also require merchants and organizations to provide at least one secure authentication service to their backend. After accepting user credentials, the corresponding remote service returns unique access token for the user. This token is stored on the phone using the most secure appropriate with each device. For example, shared preferences on Android.
6 **Mobile Long Term Key**: Each mobile client has access to a long term master keypair used for encrypt various payload on the application level. The private key is stored using KeyStore on Android and KeyChain on iOS.

As we can see, the only way for an adversary to imitate as another party on AKC system is to compromise the most securely protected AKC CA service. We will provide further assessment on securing the CA service during the development phase.

8.3 Tampering

An adversary wants to modify data stored on Peers, mobile client or data on the wire in transmission among entities. We propose several countermeasures as follows.

1 **Data on the wire**: Secure encrypted communication channel over SSL/TLS. All peers and backend services are required to provide signed certificate by AKC CA. Thus, the risk of Man-in-the-middle attack is greatly reduced.
2 **Block chain data**: All blockchain related data is validated and stored by Hyperledger Fabric SDK. Furthermore, all transactions are signed by corresponding peers. The adversary cannot freely modify blockchain data without private key from sending peer.
3 **Mobile data**: AKC mobile client is in fact a *light client*. That is, the mobile application does not perform any blockchain related functionality but act as a front-end layer to invoke remote services. Thus, we do not store much information except for user's credentials or private key on the mobile device. As pointed out above, those private information is protected by their device secure storage. Unless the adversary can root/jailbreak the device, there's nothing an

adversary can do to modify user’s data. Even when this situation happens, AKC can blacklist the device or perform remote data wipe when there is request from customers.

8.4 Repudiation

What if a legit party (merchant/AKC/Customer) wants to abuse the system and later on want to deny his action?
Fortunately, a blockchain system provides immutable records of **every successful transaction** in the system. There is no way for any party to deny his action after the transaction is confirmed in AKC eco system.
With this model, anyone who observe the flow of transaction can detect if there is abnormal behavior. They can provide an undeniable proof of the malicious behavior to AKC administrator to have corresponding action immediately.

8.5 Information disclosure (privacy breach or data leak)

As all communication on the wire are over SSL/TLS, if we assume endpoints are not compromised, we do not have to worry about data breach while exchanging among parties. There are, however, other source of information that we must protect from data leak.

1. **Peer's credentials**: Long term private keys are the most important assets that a peer must protect. AKC provides sandbox environment (docker/VM) and guidelines for organization administrators to setup and protect those data.
2. **Blockchain data**: Blockchain data may contain sensitive customer information such as social security number or history of service usages. AKC protects that information by only allowing peer from authorized merchant to install PrivateChain and access ledger data. Even when a peer has access to multiple PrivateChain, those data are kept separately in their own container.

8.6 Denial of service (D.o.S)

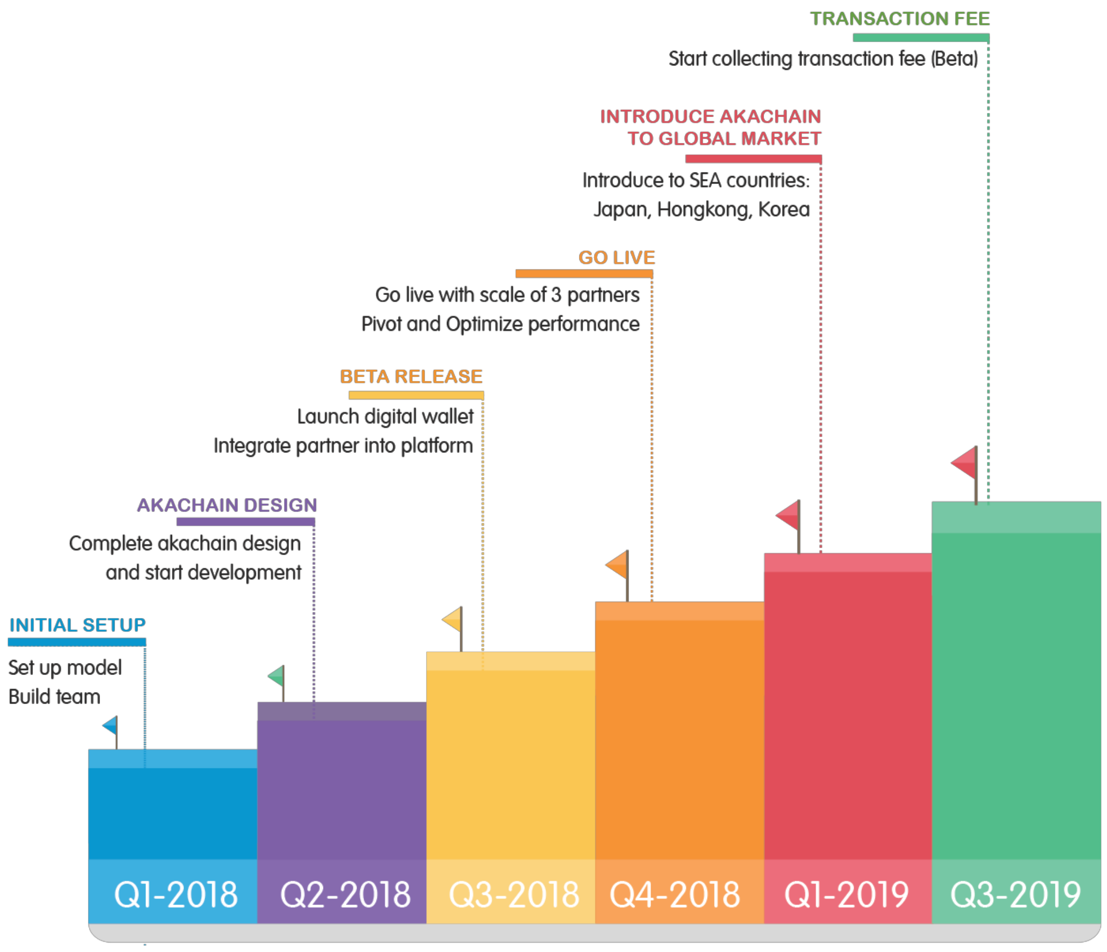
Similar to any Internet service, AKC must provide method to deal with DoS attack. We propose several countermeasure as follows.

1. **Remote DoS attack**: AKC back end service can limit the request rate from each mobile client. Thus, this greatly reduce the threat of DoS attack by rogue mobile devices.
2. **Sybil attack**: As all peers are registered with either AKC CA or delegated organization CA. If there is abnormal situation on the number of peer for a certain organization, AKC administrator can have immediate action to black list those peer by issuing certificate revocation and contact with the organization administrator.
3. **Transaction spam**: an adversary that have access to a legitimate entity in the system (merchant/customer) can create many transactions at the same time to sabotage the network. As we collect fee from each transaction, eventually the adversary will run out of fund to carry on the attack. We also employ a large number of Kafka cluster as ordering service to deal with the sudden spike on the number of transaction.

8.7 Elevation of privilege

A major concern with organizations and merchants is the integration of AKC back end service to their system. A misconfigured AKC client that exposed to the internet can easily be targeted to execute arbitrary code to elevate its privilege to gain access to the merchant system. We prevent this attack by providing a sandbox environment for the Merchant Organization Backend Package. A merchant peer is kept inside a complete sandbox environment, either in Docker, Virtual Machine or a separated dedicated server. The Peer Backend Package is exposed to mobile client and other peers to connect to. Yet, this package only communicates with the merchant existing system via an internal REST API provided by the Enterprise module. Details on this REST API is specific on each merchant and we cannot public this information. However, we can prevent the risk of privilege elevation by strict sanitization of the REST API input.

9. Product Roadmap


TRANSACTION FEE
Start collecting transaction fee (Beta)
INTRODUCE AKACHAIN TO GLOBAL MARKET
Introduce to SEA countries: Japan, Hongkong, Korea
GO LIVE
Go live with scale of 3 partners
Pivot and Optimize performance
BETA RELEASE
Launch digital wallet
Integrate partner into platform
AKACHAIN DESIGN
Complete akachain design and start development
INITIAL SETUP
Set up model
Build team
Q1-2018
Q2-2018
Q3-2018
Q4-2018
Q1-2019
Q3-2019