Network DID Comm Implementation: The Framework for Authentication, Authorization, and Access using the QLC chain and Smart Contract

Concept Design by:

Allen Lee
Dan Kolkowitz
Chris Zhao
Susan Zhou

May of 2020

Table of Contents

Abstract

This document describes a new authentication, authorization, and access (AAA) solution called “DIBS” (Decentralization, Identity, and Blockchain Service) that incorporates the blockchain and smart contracts as its fundamental architectural components. The DIBS framework takes advantage of the strong cryptographic relationship users have with the blockchain, to prove ownership of cryptocurrencies or to prove the identity of parties in transactions. The system borrows from the well-known security system “Kerberos.” However, while Kerberos is a “federated” or “centralized” security system, the use of the blockchain allows similar security relationships to be established in the “decentralized” environment of blockchain related computing. “DIBS” is a pun, and is intended to symbolize ownership: “DIBS, it’s mine” is an assertion of ownership on the playground. Here it is done silently instead with ownership verified and established by the Blockchain.

Kerberos creates credentials called “tickets” that eliminate many of the vulnerabilities that exist around other systems that use static “key exchanges” for security. Kerberos uses a central key distribution server to distribute credentials and private keys to users and services within an enterprise environment. However, the AAA framework for the virtual private network (VPN) uses blockchain and its known security associations to build the tickets. With the DIBS framework, tickets are designed to specifically support dynamic configuration, and the configuration of network applications with smart contract capabilities on the blockchain.

By using the blockchain, the DIBS solution creates a general framework for providing network security for decentralized applications (Apps). Once the identity of the requestor is linked to a previously approved authentication process with credentials loaded on the blockchain, access to requested services may be granted. When users are authenticated in this way, client access to the hosts and services on the blockchain nodes is equally secure as transactions on the ledger.

The above methodology is used by DIBS to create a framework for hosting DApps that use the blockchain. DIBS creates “nodes,” which host clients and services on top of a fully secure peer-to-peer (P2P) network. On this P2P network, clients using the framework can communicate privately and securely. Each node is provisioned with a wallet and private key whose public key is registered in the blockchain. Nodes are added to the P2P network as they become active.

The DIBS implements its framework on multiple platforms. It produces embedded and virtual images built on the QLC Chain Node version of Linux. A software developer kit (SDK) and the set of node builds will allow internet-of-things (IoT) vendors and other providers to create applications or port to different environments using the DIBS framework. A token model (which requires a blockchain’s tokens for access to the platform and services) will apply to applications and node instances that seek to access the blockchain using the the DIBS framework.

1.0 Introduction

1.1 Blockchain, decentralized applications, and software as a service

Blockchain technology is upgrading and transforming parts of information technology (IT) from a centrally-controlled business model to a vastly more decentralized one based on the security and trust of the transactions recorded on the blockchain. For example, Ethereum and EOS are projects that aim to provide trusted environments for decentralized business applications. These applications primarily rely on the transparency and trust in the ledger—when transactions are performed and their output entered into the ledgers, all participants in the economies have agreed on the entries and accept them.

Though there is activity creating and deploying decentralized applications (dApps) with smart contracts and blockchain, some key functions of the internet that the dApp runs over haven't significantly evolved to support the security functionality for dApps that have been deployed and under development. The security required must be appropriate to the decentralized technology.

Let's consider the security that surrounds software-as-a-service (SaaS), which is the dominant licensing and delivery architecture for dApp deployment. SaaS offerings have grown rapidly with the popularity of cloud computing. The technology provides internet access to applications, which previously had been locally hosted. The convenience and simplicity of SaaS, has made it by far the most popular method of offering services on the internet.

With SaaS, applications run from points of presence in the cloud, typically using web-based protocols. The service providers manage their users' accounts, choosing how to create them and what security is required to log in. Often SaaS providers accept credentials from other trusted sites, such as Facebook and Google, for convenience since their users have these accounts, which allows SaaS providers to leverage their account names and passwords. When registering for these sites and services, the user can simply choose their Google or Facebook user ID's for registration. The user is redirected to those sites through an authentication process where they are sent to log in to their remote account and then sent back to the site of registration. The user is now trusted on the new site of registration, as Facebook or Google had previously trusted them.

Beyond not directly authenticating users, this SaaS provider acceptance of credentials from other sites has the side effect of letting Google or Facebook know which foreign sites the user accesses. This has many issues regarding privacy for the user, and does not really provide the SaaS service concrete knowledge that the actual user is accessing the account.

1.2 How HTTPS and VPN will be incorporated

SaaS access is typically web-based and runs over “HTTP” or “HTTPS” protocols, which power centralized browsers. All secure access will be performed over HTTPS, to minimize the chance of information theft from direct observation of the sessions. HTTPS provides end-to-end encryption between the client and the server, thus removing any real chance of malicious actors recording or observing the sessions and stealing the data. The security connection is established between the client and the server, and is formalized when the client reaches the server. Virtual private networks (VPN) can be established to restrict who has access to an application, but there is a large setup cost and the arrangements are typically fixed in advance because of the restrictions. The use of a VPN can isolate a server from other network traffic, but it can be difficult to provision to allow all of the desired users access through the VPN.

1.3 Blockchain and Remote Authentication

Blockchain technology can replace this remote authentication login process, where the service provider can trust the blockchain instead of another 3rd party site. The purpose of a password on an account is to verify that it is the same user and identity that has been authorized through an account creation process. Weak passwords and stolen ID’s allow malicious actors to steal accounts and commit fraud against, or with the stolen credentials. When the blockchain is used for authentication, account authentication security rises to the same level as the encryption and security technology being used.

Generally, technologies that implement encryption depend upon an establishment of security associations between parties based upon the type of encryption, and exact keys that are to be used. These security associations are formed through sets of keys that allow one party to send data encrypted to the other party.

When keys are issued, several steps are typically required to distribute and use them. Besides the cumbersome distribution and use process, the security of those exchanges is a major issue in deploying encryption. Even in the case of “asymmetric encryption” over the secure sockets layer (SSL), a certificate must be created and installed together with the private keys that support the particular certificate. The unique certificates are created in advance of the process, and represent connections to each individual service they perform.

1.4 The Authentication, Authorization, and Access (AAA) Framework

The DIBS platform for authentication, authorization, and access creates a general framework for securing dApps on demand. The blockchain allows for the registration of applications on the blockchain, which can be accessed via payment by tokens. For a listed application, smart contracts are designed to automatically execute a process when a token payment is received. The smart contract creates and delivers the encrypted keys, and establishes networking instructions for the creation of a new, secure session for the application. All transfers are encrypted with the public keys that are registered by the blockchain, in an "on-demand" and transparent manner.

The network that uses the DIBS framework is able to authenticate nodes in the same way as it is with its users. When a node is provisioned, a wallet is assigned a private key and access methods are installed as part of the node. The node can then authenticate to the blockchain and access its wallet, in the same way a user does.

The DIBS framework uses this authentication process to create a private network for secured applications through the blockchain. An authenticated node acts as the host of the applications and joins an authenticated private network. Then, users can connect to those nodes and applications through authenticating to the blockchain.

The parameters for connecting to authenticated nodes are established by smart contracts on the blockchain, which are initiated when transacting with the service through the blockchain. The authentication and authorization credentials returned are protected in the wallet to the same extent as the actual currencies being held by the blockchain.

Using the secure identities and the automatic replication of the blockchain, DIBS enables secure communication without having to exchange keys or register certificates. If the user can authenticate to the blockchain, a secure transaction can be started with a token payment, which executes a smart contract. The smart contract then delivers the required security parameters without having to use external methods or protocols to exchange keys between the cooperating parties.

An application instance (or deployed instance) registers a unique smart contract that supports the parameters required for the application. For example, the smart contract will return values like address information of the instance, and unique security credentials to access the client. The returned values will be unique to the deployed application instance.

Application instances advertise themselves on the blockchain and users transact with those instances to select applications. If a user deposits tokens in a node on the blockchain which is an application instance, a smart contract will get executed for that node. The smart contract will create a "ticket" allowing the user to access the application instance. Tickets are the security mechanism that binds the authentication and access to the application instance.

1.5 Software Development Kits

The DIBS framework distributes a set of software development kits (SDK) for the development and integration of third-party applications. It supplies the logic to implement DIBS framework and access the P2P network.

When applications are deployed, they will follow the same token rules and requirements as the DIBS framework supported dApps and technology. All applications will use public blockchains for DIBS functionality and must operate in the decentralized manner the technology supports.

To assist development in various facets of this environment, there will be two sets of SDKs: the client SDK, and the server side SDK.

1. The client SDK will consist of a reference dApp, together with build files and documentation, both for the dApp and the overall libraries. When the included libraries are linked, this will give the ability to access the P2P browse services (connect to the P2P network, connect to a service, and disconnect). The distributed libraries include support for multiple functionalities of blockchain including smart contract access and wallet support, and P2P networking integration and testing. Additionally, client code can be developed on a test environment.
2. The server side SDK will consist of a wholly runnable application, together with the build and image sources. The build will run in the major cloud environments and support the node functionality described in the document. This includes P2P networking, ticket processing, and multiple functionalities of public blockchain integration including smart contract generation and wallet support. However, the SDK will only support full nodes, not partial nodes as with embedded devices. Embedded device integration will require its own SDK and not be available as part of the public 3rd party open integration support.

1.6 The DIBS full life cycle applied to applications

The DIBS framework builds a backbone network using the deployed cloud-based images and embedded devices. When started, each image joins a DIBS-enforced, P2P network. Each node acts as a router in the P2P network, as well an endpoint if it's deploying an application. Every node that supports an end application also registers that application and its address on the blockchain.

Through the DIBS framework, when a user registers and accesses an application, their identity will be established through access of their wallet and through activation of the application via the blockchain. Once an identity is established, the connection through the P2P network will be created and the application accessible.

One such application that can utilize the DIBS framework is a dApp VPN. The DIBS framework provides the blockchain VPN-based network with the following advantages over current VPN services that run over the internet:

- Increased capacity,
- Isolation of traffic,
- Approved and managed participants,
- Minimal capital expenses of partners, and
- Maximal employee utility.

The DIBS framework will utilize an embedded device (a dedicated piece of hardware to support functionality) and a runnable system image (a software-defined device that supports functionality in a virtual environment) for users to buy and deploy. DIBS decentralized network solution will supply both a client and server. The client will allow a user to select any VPN device listed as available. Whereas, the server will run as an embedded device or as an image in the supported cloud environments.

The VPN server can be configured to support client-to-site, and site-to-site VPNs. Client-to-Site VPN's are one-sided, as a VPN client can connect through it to hosts on the server side network. However, traffic cannot be initiated from hosts on the the server side to go outside of the network. A site-to-site VPN allows users in various fixed locations to establish secure connections with one another. The VPN will be the first instance of an DIBS-enabled application that utilizes the blockchain at every security point within the DIBS framework. The VPN server network will be built (open sourced) using the DIBS framework SDK for applications and embedded devices.

The VPN client and server will consist of these decentralized nodes that have been configured and deployed. Traffic among the nodes will be secured on the private point-to-point network, which is constructed as the nodes are deployed. Any packets traversing the P2P network will be encrypted traffic from authenticated sources and destined to authenticated VPN end points.

2.0 System Overview

2.1 Blockchain Ledgers and P2P

Blockchain ledgers synchronize between all instances of the recorded ledger to derive a single agreed upon set of entries. The usability of the ledgers is directly related to their transaction speed, plus the reliability and dependability of the systems hosting them. The cryptography protecting the entries in the ledger cannot be broken by any practical methods. However, regular network attacks such as distributed denial of service (DDoS), malware, domain name system (DNS), and phishing attacks against the clients can make the environment vulnerable to identity theft and regular application attacks. Decentralized blockchain applications, which support the client access, require network improvements to provide “trusted and distributed transmission” of blockchain transactions and the protocols they support. Further, authenticating client applications to the blockchain should be implemented in such a way that access to the hosts and services on the blockchain nodes is equally secure as transactions on the ledger.

The proposed DIBS framework describes a blockchain-based, private network solution that specifically focuses on end-to-end security for individuals and businesses. With this solution, the cryptography of the blockchain is utilized to authenticate and authorize users. With users’ public and private keys, the DIBS framework establishes the ability to create secure connections between parties with no central key distribution server. The DIBS process relies on the replication of the blockchain and the accuracy of the ledger to assure that users’ resources are uniformly protected.

Users publish the availability of their resources through the blockchain. Other users can then access those services using credentials supplied by the execution of smart contracts, which have been installed and authorized by the owners. The binding between the requestor and the provider of the services is guaranteed by the security associations built through the blockchain.

2.2 The P2P Network and Application Access

The DIBS framework solves network identity, access control, and configuration problems through building an authenticated P2P network that consists of nodes that host the services offered through the blockchain. Users authenticate via the transaction/wallet application programming interface (API). Access control is determined and administered through the execution of smart contracts. As more users participate, more routes become available that can be measured for efficiency and quality of service. Efficiency and quality of service will be ranked via an algorithm to help with routing.

The DIBS framework ensures the functionality of core nodes by providing software and hardware appliances provisioned with private keys and bootstrap information to guarantee plug and play. In a P2P blockchain network, when a node attempts to connect to the network, it must locate a relay node (a.k.a. a bootstrap node) that provides a list of nodes with which it can connect. This process will assist blockchain services for networking applications for individuals and for enterprise and other business customers.

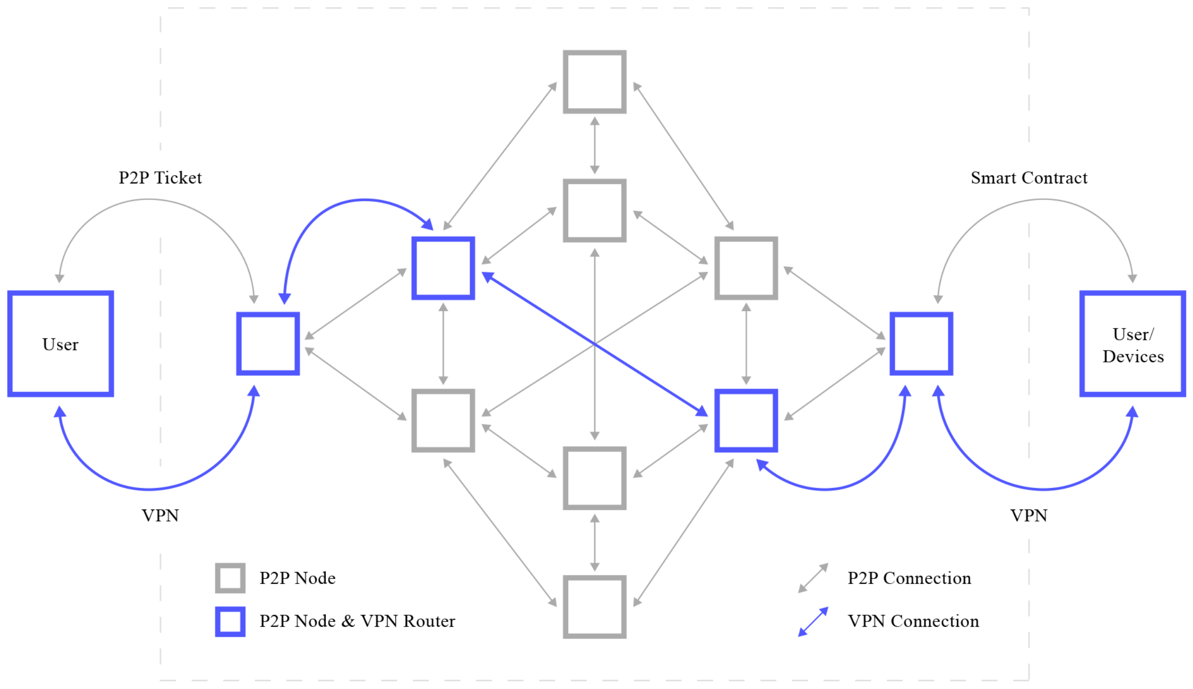

Figure 1: P2P layers within DIBS network

Within the DIBS framework supported network, two P2P layers have been designed: first, the P2P ledger layer, which is the blockchain ledger layer; and second, the secure application route layer, which is a P2P tunnel established between peer nodes. In the above image (Figure 1), the P2P ledger layer is illustrated by the blue paths, and the secure application route layer by the red paths. The application route will be the shortest path between the two nodes that route across the P2P layer.

When all the aforementioned processes are combined, the mechanics behind the DIBS frameworks are established as follows:

1) Users can join the P2P network by either running the client application to establish client-to-server connections, or connecting through VPN nodes that route traffic (the users are not authenticated to the nodes).
2) The application provider (the destination) specifies who has access. Only authenticated users can create a connection. The mechanics revolve around executing a smart contract on the blockchain.
3) When a user requests access to a remote application, the user's client application sends a request to a bootstrap node to get a "ticket" for the remote service as well as to join the P2P network. The request is made through payment to the ledger to execute a smart contract.
4) The user's request requires a token to initiate a "smart contract" on the ledger. The smart contract issues two "tickets" to the user, one to join the P2P network and the other to access the remote service. Both of these are returned to the user encrypted in the user's public key.
5) When the user decrypts the ticket with their private key, they can present the ticket to the P2P node to join the P2P network.

6) After joining the P2P network, they will connect to one specific application node on the DIBS P2P network and reach the destination through routing in the P2P network. The destination is a full-node on P2P network and has its own wallet with public key & private key.

2.3 Nodes in the DIBS decentralized network solution

Nodes in the P2P network can operate as “full” nodes or “light nodes”. Full nodes contain all the required functionality for the applications and network services, and most importantly include the blockchain to perform the full ledgering locally. “Light nodes” provide all the other functions including the networking, application services, configuration interfaces, but do not include the blockchain.

In the case a node is provisioned as part of an enterprise solution, it will contain multiple instances of routing software and related tables which allow it to isolate the packets (of data moved across the network) based on destination and the network to which they belong. The DIBS framework creates “virtual routing tables” using VRF technology to maintain multiple, completely separate, networks in a node. The DIBS framework uses “VMPLS” (a modified version of MPLS) as a tagging technology to maximize the packet forwarding using multi protocol label switching (MPLS) tags while maintaining independent routing tables and P2P connections.

When a node belongs to an enterprise, all transactions are made with its own “native” tokens. The smart contracts it executes are specific to the node and any client activity is attached to the appropriate network.

The DIBS framework will launch with a turnkey VPN and routing product built on QLC Chain Node (See Section 9.0) that will allow users to bootstrap into the network. This process will be supported both as an embedded device and as a runnable image for major cloud environments. When run, the nodes join the P2P network and route traffic across the P2P networks. The nodes can be deployed as a VPN endpoint or as a node hosting other 3rd party applications (as described below).

3.0 The DIBS Architecture

The DIBS architecture implements a secure application infrastructure that's integrated with the blockchain, so developers can deploy secure dApps without the need for central coordination of user identity registration and key exchanges. DIBS utilizes the users' wallet integration to prove their identities and to distribute cryptographic information so the user can communicate directly across a secure channel. Simultaneously, the DIBS framework develops its own secure applications to provide internet users with on-demand VPNs built through DIBS services.

The DIBS architecture establishes an authenticated P2P network, over which DIBS-enabled applications can run. The services rely on blockchain ledgers in the following ways:

- All transactions are made through payment to entities on the blockchain;
- Users are authenticated through their access to their wallets; and,
- Services are listed and selected through their postings on the ledgers.

Nodes join P2P networks as they are run and advertise their services. As a distributed service, the DIBS technology leverages the DIBS P2P nodes offering VPN services without the need for central control and management of the network. All nodes are authenticated via the blockchain, and only authenticated nodes can issue and route traffic across the P2P network. The decentralized operation of the nodes via the blockchain allows consumers to use and provide services without the need for approval or permission from a system central operation.

Below is a discussion of major components of the DIBS architecture and how the services are built on top of the architecture.The DIBS architecture primarily relies on the authentication to the blockchain and the execution of different smart contracts that deliver "tickets" to the available services on the network.

3.1 Block Chain and Blockchain Nodes

DIBS uses the blockchain to distribute authentication and access privilege's, as well as bootstrap information for joining the P2P network. A node joins the network after executing a bootstrap smart contract. The wallet which belongs to the node retrieves a P2P network ticket, as well as the location of the P2P to apply the ticket to and connect to the network.

There are 2 kinds of nodes that access the blockchain, light nodes and full nodes:

1. Light nodes contain the client logic but not copies of the blockchain and ledger. Instead they rely on an API to the blockchain to read and write transactions. Even though they do not contain the ledger themselves, they must still utilize "bootstrap-nodes" to retrieve their network and bootstrap information.
2. Full nodes contain complete copies of the blockchain and ledger. They are pre-configured to retrieve their network information from the bootstrap-nodes but they additionally perform synchronization with other blockchain nodes as they are modified by local clients.

Server nodes, nodes which hold application end points, are full nodes. For example, a VPN server will always be running on a server node.

3.2 Authentication, Authorization, and Access through Blockchain

Authentication, authorization, and accounting is a framework for establishing trust and security in traditional computer networks and systems. However, with regards to blockchain integration, the DIBS framework focuses on authentication, authorization, and *access*. Focus has been shifted to access, as the blockchain stores an immutable accounting record. Additionally, access to the blockchain necessitates an increase of procedures due to its unique requirements. A comprehensive system of DIBS establishes common identities and access controls across diverse systems. Using the blockchain allows common identities to be utilized in decentralized computing environments.

The lack of useable DIBS technology with blockchain development has held back its usefulness for general computer and network security. This framework uses the blockchain to build a system of DIBS that works in the spirit of blockchains and decentralization. This framework could potentially lay the cornerstone for DIBS in decentralized blockchain enabled environments.

3.3 DIBS through Blockchain processes

The blockchain authenticates users as they access it through the ledger protocols—as the related records in the blockchain can be updated by the agent, the identity of the owner of the related records is cryptographically proven. Since the blockchain instances are replicated, any instance of the blockchain can authenticate any user holding records within the chain. This is true for users as well as registered nodes.

DIBS nodes distribute the blockchain as well as application instances residing on the nodes. Smart contracts deliver the data to prove that a user is authorized to access services being listed and hosted via blockchain.

The authentication and authorization follows a “Kerberos-like” model of using tickets matching identities and services. Kerberos is a network authentication, authorization, and access system that provides strong authentication for client/server applications by using secret-key cryptography. The general flow for DIBS authentication, authorization, and access is as follows:

1. Users register the public keys on the smart contract.
2. All data returned to them is encrypted in their public key
3. Responses are decrypted using their private keys
4. “Tickets” are included in the encrypted responses, which will be presented to services. These tickets are encrypted in the respective service public keys.
5. Services decrypt tickets and allow access based on the validation of the tickets.

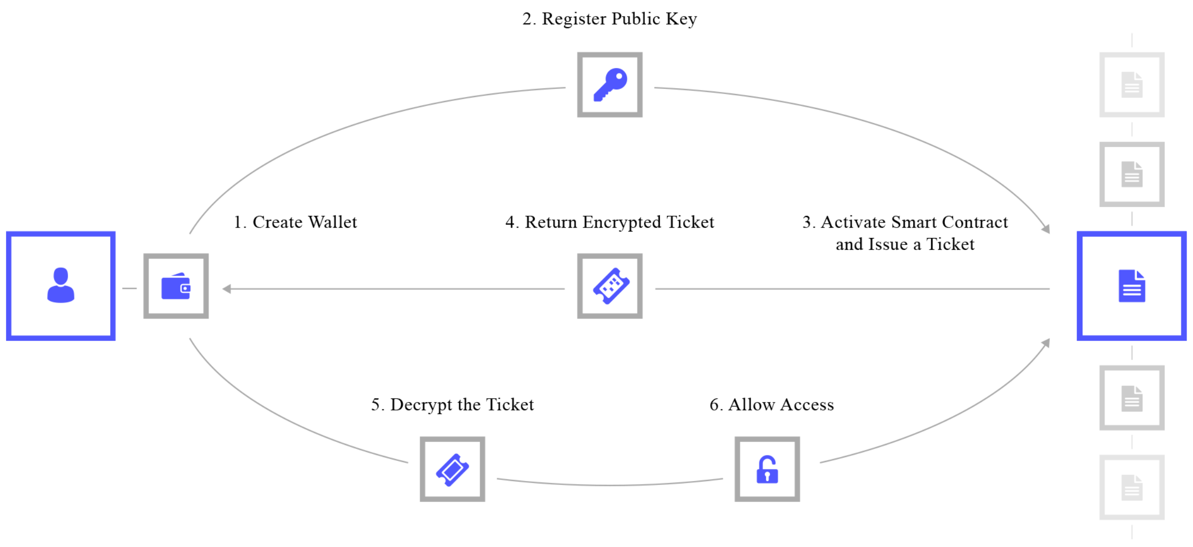

Figure 2. Authentication and authorization process

Users authenticate to services by utilizing "tickets," which get returned to their wallets when services are accessed through the blockchain. The tickets are encrypted and returned to the user, and contain authorization for the services the user can access. The encrypted tickets for the services are specific to each server and service, and are completely opaque to the user presenting them. The encrypted tickets prove that the user was able to authenticate to the blockchain and that they are entitled to use the requested service—the fact that the service can decrypt the ticket means it was specific to the requested service, and was verifiably presented to the user.

Nodes access the blockchain and use their private keys to decrypt tickets that are presented to them. The services in the nodes each have tickets and must understand the structure of each ticket presented.

Tickets consist of multiple fields that guarantee the ticket's integrity and verify the encryption. The tickets are established to protect against reuse, theft, and illegal modifications in such a way that if a malicious actor steals the ticket, it will be unusable and unreadable. The fields of the ticket are as follows:

- Time stamp: The time stamp is included in the response to the client as well as in the ticket encrypted for the server.
- Address: The address is the P2P node address of the server gotten when it attached to the P2P network.
- Service type: a constant in the ticket that indicates the type of service being accessed, which must match the type that the service is running.

The time stamp is used to compare the time when the ticket was issued, to the time the ticket returned by the client following attachment. As time synchronization will not be exact, a delta of around 10 minutes is tolerated for differences in the time between ticket issuance and return to the user.

Smart contracts are the mechanism for key generation and distribution, and replace the central key distribution authorities of federated systems. Further, instead of access rights being set and managed by the central IT organization as in an enterprise, the DIBS framework allows any parties to set the access control for their own applications. DIBS provides authentication and then the mechanism to perform the authorization and access control after access has been granted.

Instead of performing key exchanges with key exchange protocols or procedures, smart contracts create session keys and distribute them as transactions executed on the blockchain. A client that seeks to access a service invokes a transaction with an advertised service listed on the blockchain. The smart contract then uses the public keys listed on the blockchain to create the tickets for the services, with the required security parameters encrypted within the respective public keys.

When the client initiates the smart contract execution with an attached transaction, the service's smart contract creates a service ticket and lists its requirements for approval on the net. For example, in the case of a private network, this would include the P2P node address, plus a ticket that contains the authorization for the client to access the service. The possession of the ticket will prove the client was able to authenticate themselves to the blockchain, and that it has been authorized to access the service.

Represented in the image below (Figure 3), is the workflow process which represents the service ticket invocation through to the ticket granting.

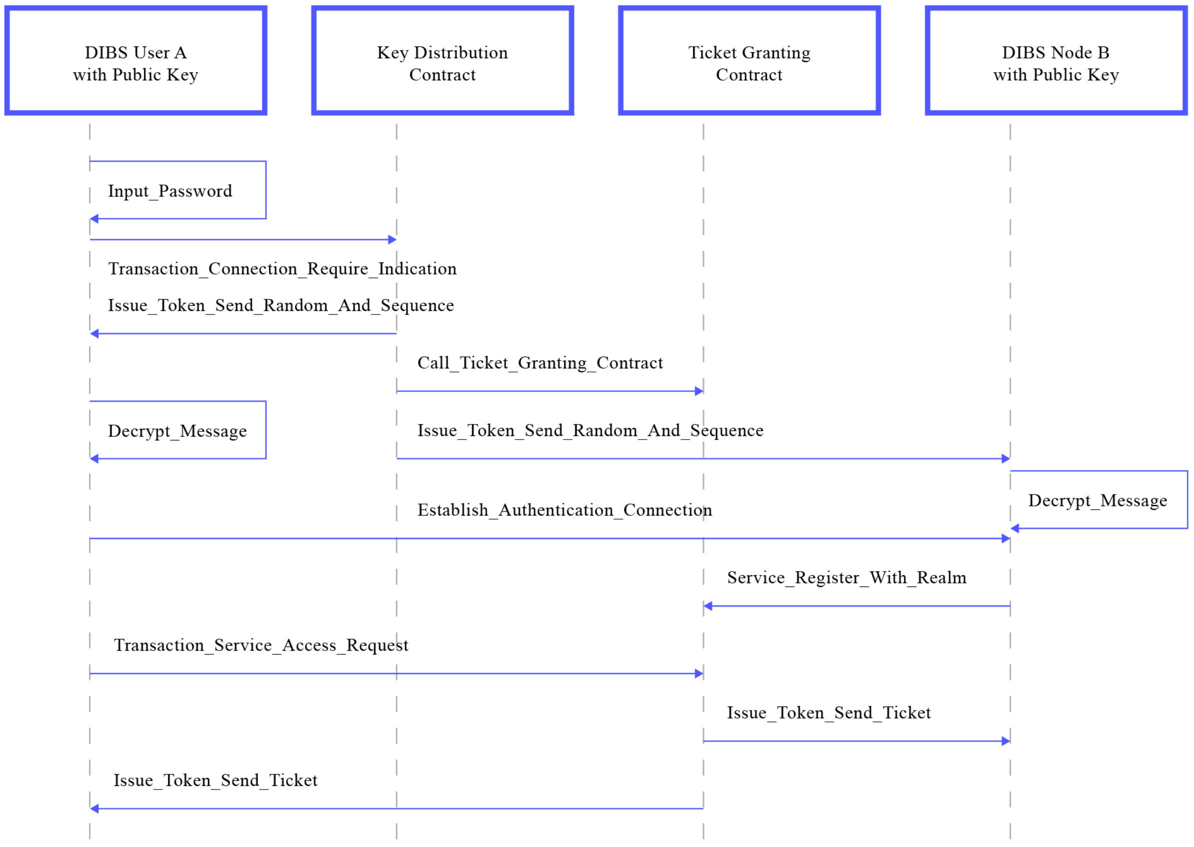
DIBS User A with Public Key
Key Distribution Contract
Ticket Granting Contract
DIBS Node B with Public Key
Input_Password
Transaction_Connection_Require_Indication
Issue_Token_Send_Random_And_Sequence
Call_Ticket_Granting_Contract
Decrypt_Message
Issue_Token_Send_Random_And_Sequence
Decrypt_Message
Establish_Authentication_Connection
Service_Register_With_Realm
Transaction_Service_Access_Request
Issue_Token_Send_Ticket
Issue_Token_Send_Ticket

Figure 3. Service request and ticket granting process flow

3.4 The P2P network with VMPLS

The blockchain ledger distributes identities and access privileges for users and nodes. DIBS applies the same concept to network routing and access. DIBS connects nodes hosting users and applications to a "blockchain centric," P2P network that only allows authenticated and authorized traffic to cross. Nodes on the P2P network have addresses that locate them in the network, and the routing tables are specific to those nodes.

The P2P network consists of all currently active nodes that have joined the network as clients or servers. Clients and servers will only access the blockchain to execute a smart contract for authorization. To connect to the P2P network, the "nearest" neighbor is selected to provide a connection. When a node joins the P2P network as a server, it broadcasts its immediate network address to the P2P network. The P2P network then acts as a routing vehicle to provide optimized and private traffic. Nodes are connected as they join and are authenticated upon connection.

The P2P network helps to isolate the environment of the blockchain and dApps from the larger network/internet environment. When the nodes join the P2P network, routing to applications on the network are done through the P2P network and not through direct internet routing. Application end points are advertised via the addresses on the P2P network, not internet addresses. The nodes provide "bridging" across firewalls, so routable addresses and network address translation (NAT) firewalls are not used in accessing the services.

The general process for bootstrapping into the the P2P network is as follows:

1. The bootstrap comes from a set of pre-defined "bootstrap nodes."
2. Transacting with the bootstrap contract gives a list of nodes to connect to in order to join the P2P network.
3. The bootstrap nodes return a list of hosts for joining nodes to connect with in the P2P network. The node issues a "ticket request" to get a ticket via smart contract.
4. A ticket presents an encrypted credential to a router to request that they allow the node to join the P2P network.
5. Once the joining nodes become part of network, their address and P2P identities are shared as part of the routing table for the P2P network.
6. The joining nodes are constructed on an "MPLS" layer that routes all traffic. Here, the DIBS process tag the packets traveling on the P2P network with a "VMPLS" tag, which just labels the packet as belonging to a particular P2P route (it is largely similar to MPLS but with no actual carrier support). All traffic between endpoints happen across the P2P network.
7. Any node in the P2P network is addressable from the private networks and can route traffic between client switch private addresses to servers which also have private addresses.

Each node in the P2P network authenticates to the blockchain to retrieve bootstrap information. Below (Figure 4) is a graphic of the bootstrap and VMPLS VPN processes, followed by their procedures.

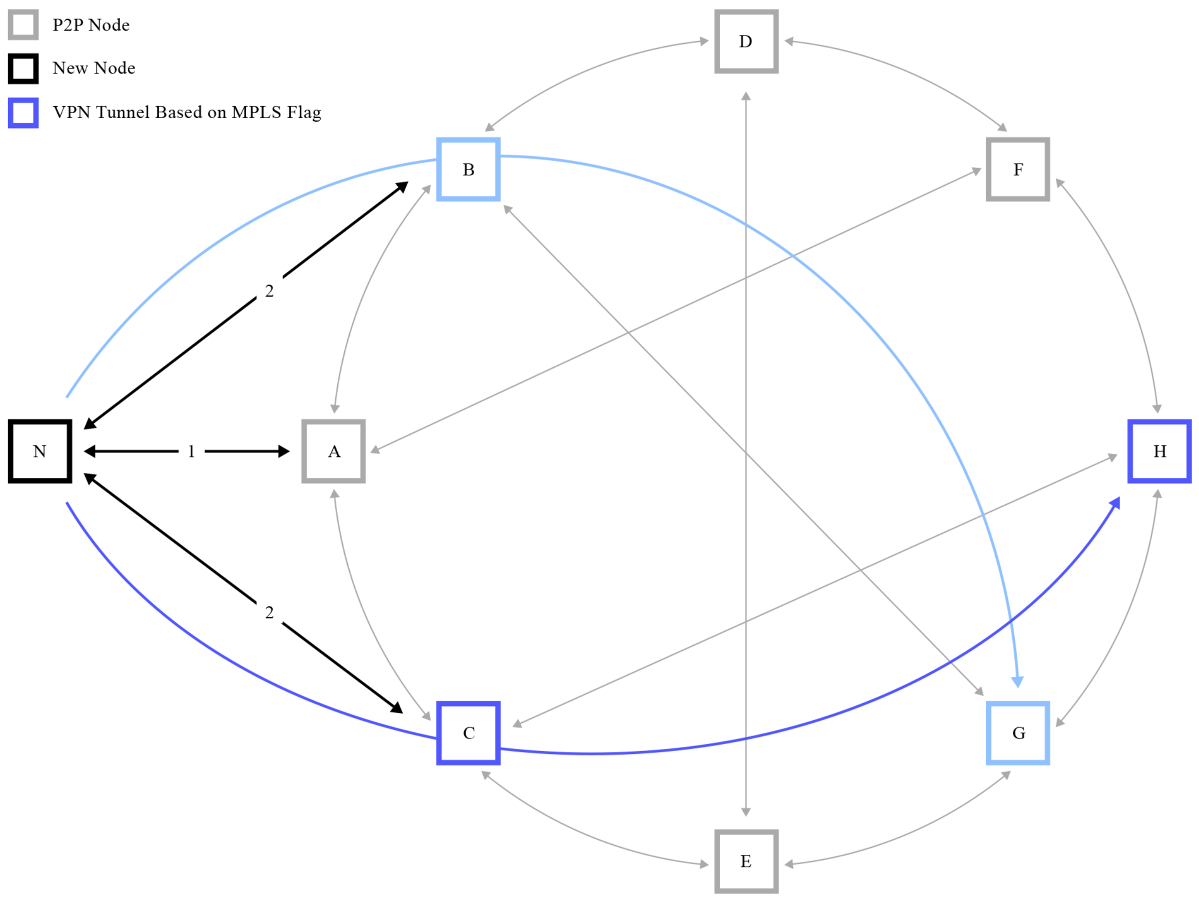

Figure 4. Bootstrapping and VMPLS VPN architectures and processes

The P2P network is built out of the nodes that host the DIBS services. When a node is added to the internet, it is assigned a private P2P address and added to the P2P network. Traffic among clients and applications that use the P2P network is routed according to P2P routing algorithms. For any packet using the network, the shortest path between two nodes in the network can be calculated on the basis of measurements of packet times. Packet time measurements are determined by observing and gauging the amount of traffic traveling through the network.

Use of the P2P network for application access has multiple advantages:

1. Isolation of traffic to only coming from authenticated entities on the P2P network.
2. Optimized routing by placing nodes and images in high performance environments.
3. Engineered networks through use of P2P measurements and VMPLS tags.

3.5 Peer-to-Peer Network for Enterprise

The DIBS framework supports organizations that maintain their own P2P networks. In this case (using “MEF Organization” as an example), when a node is added, it is identified as belonging to the network for the enterprise.

This is a network on which MEF Organization’s traffic is exclusively routed. Networks are created for an organization as nodes are added that belong to its enterprise. When a node is added, it can be attached to any node within any P2P network formed around the MEF Organization technology. Every node that is part of the new route will maintain a separate routing table for a particular route. Nodes have the ability to support multiple routing tables—the amount of supported nodes depends on how many specific enterprise networks have been added. For these private networks, the transactions with the blockchains are with tokens specifically issued for the respective enterprise.

In addition to connecting to a forwarding P2P node, a “VMPLS” tag is added to the node which indicates the organization to which the node belongs. The VMPLS tag labels the traffic associated with an organization, and allows it to use a set of routes with other packets that share that same tag. This tagging system guarantees nodes which support the same tags can be used for the traffic associated with the same tags. Conversely, traffic of data packets without the same tag, are unable to traverse through nodes approved for use by one organization.

VMPLS is based on MPLS used in packet switching for carrier networks. The use in the carriers is basically the same in that it minimizes the routing decisions for traffic carrying the same tags. At the same time, it allows customers to engineer their own traffic by assigning tags through common routes to guarantee the flow of traffic. MEF Organization uses the VMPLS tags in a similar way to isolate the traffic between customers and keep unrelated traffic from traversing enterprise customer nodes.

As blockchain distributes authentication and access information, each user is identified by their wallet and public key on the block chain. Their personally identifiable information (PII) remains private and is not utilized. Users authenticate to the network by retrieving credentials from their wallets. The user’s privileges and access methods are retrieved when they retrieve the contents of their wallet from the blockchain.

A network is built out of router nodes which access the blockchain for configuration and operation details. Each node has an identity in the block chain and can retrieve its wallet in the same way as a registered user.

The primary difference between how nodes and users access data is through the types of keys they use for the process. Users create their wallets and retrieve data, which is encrypted in their public keys. Nodes use provisioned public key and retrieve their configuration information using their private keys.

3.6 DIBS Virtual Private Network (VPN) Support

The DIBS framework launches a complete VPN solution built on the functionality described in this whitepaper. The VPN solution has the enormous advantage of automatic configuration and instantaneous deployment. When a node with the VPN server is powered on, it registers with the block chain and becomes available for client connects. The client creates a connection to the VPN in the same way as other dApps—they initiate a transaction with the VPN server and the connection is created. The use of the "tickets" and P2P network removes the steps of network configuration and key-exchange.

When VPN services are offered through the blockchain, the same processes for installation and operation apply as any application hosted and deployed in the DIBS platform. To begin, a user registers their VPN asset, which will simultaneously deploy a smart contract. The smart contract will trigger the following tickets required for the initial service:

1. Create a VPN ticket for a user to access a VPN service,
2. Create a ticket to retrieve bootstrap information (how to join private network)
3. Create a ticket to join the decentralized P2P with the ticket retrieved in step 2.

The VPN server retrieves the tickets and configuration information from its wallet. The server then joins the P2P network via the bootstrap process as explained above, and advertises itself on the blockchain. The server will then allow connections based on the tickets it has been presented. These tickets were generated by the smart contract on the blockchain.

The client chooses the service via the blockchain. On completion of a transaction, the server receives the bootstrap and ticketing information for the service. It joins the P2P network as described above. In contacting the remote VPN server, the node presents the ticket it got in the transaction, and opens the connection using the P2P routing.

3.7 VPNs built on top of P2P connection

VPNs connect clients to provide encrypted communication between two points. Frequently, the destination in a VPN is not reachable unless there is a public address that is assigned to represent the private network and addresses within that network. The P2P allows a private VPN server with a private address to connect to a P2P node, which acts as a forwarder into the private network.

The blockchain smart contract returns the address and routing information for any client when payment is received for the network service.

The VPN configurations supported are client-to-server VPN, or site-to-site VPN. In both cases, services are accessed through payment of tokens to the service providers. OpenVPN tunnels and a certified ticket allows the user to access the destination node and then create the VPN. This VPN is created and utilized for the following scenarios:

1. Client-to-server VPN scenarios:

- Users can advertise their VPN servers through the blockchain.
- When a user purchases access, they obtain a ticket to the P2P network.
- An OpenVPN tunnel is created between the user and the destination.

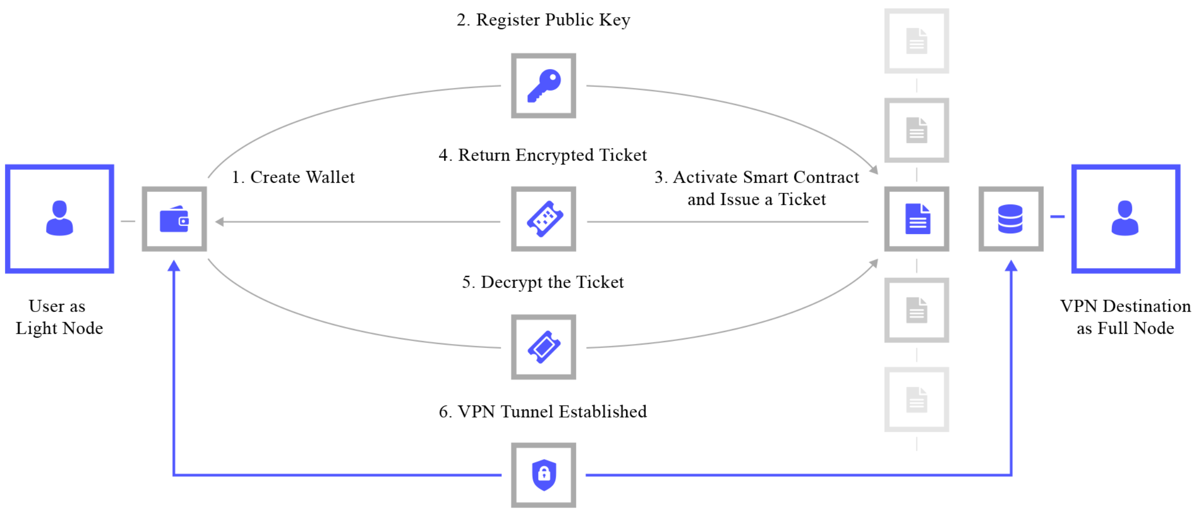
2. Register Public Key
4. Return Encrypted Ticket
1. Create Wallet
3. Activate Smart Contract and Issue a Ticket
5. Decrypt the Ticket
6. VPN Tunnel Established
User as Light Node
VPN Destination as Full Node

Figure 5. Client-to-Server VPN scenarios

2. Site-to-site VPN scenarios:

- A user can join a VPN group, to connect their VPN service with another VPN service, and create a site-to-site VPN.
- When a user chooses a VPN to connect to and another user does the same to them, a site-to-site VPN is created.
- VPN support is provided as a network
    - Both sites pay token to the tunnel

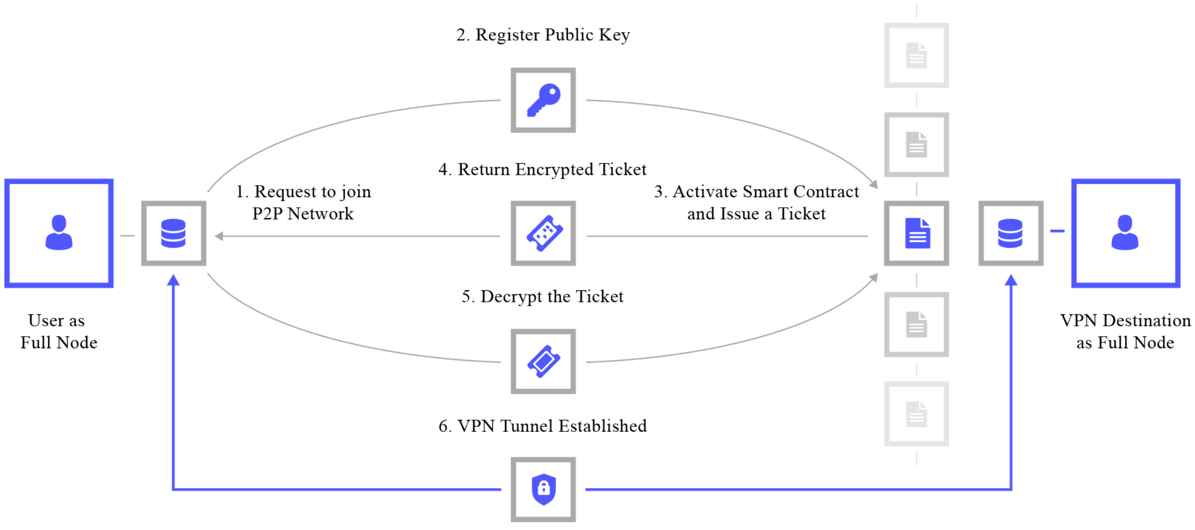
2. Register Public Key
4. Return Encrypted Ticket
1. Request to join P2P Network
3. Activate Smart Contract and Issue a Ticket
5. Decrypt the Ticket
6. VPN Tunnel Established
User as Full Node
VPN Destination as Full Node

Figure 6. Site-to-Site VPN scenarios

4.0 Building Virtual Private Network's for organizations

Through the DIBS decentralized network solution, organizations are able to create their own private provisioned networks and VPNs. The organization-based process is as follows:

1. A private P2P network is created on an enrollment,
2. The DIBS P2P routers connect to the network and route between the clients and servers in an isolated way,
3. The client's users access the network through issued tokens,
4. The service leverages the blockchain to distribute identities and privileges, and
5. The blockchain allows distribution of user public keys which are presented to access user wallets.

With regards to an organization's users, all features are the same as for the DIBS users, but using Client Organization issued tokens for identity information:

- Users log into their organization
    - When logged in, their wallet shows only tokens for that organization
    - It is up to the organization to supply tokens to their users (how do we help support this?)

- Users find services specific to their organization
    - These run on the private organization net

When an enterprise provisions their service they get the ability to invoke on-demand private networks. These networks can include site-to-site VPN, client-to-site VPN, or business-to-business (B2B) connectivity via site-to-site VPN.

5.0 Embedded devices on QLC Chain Node

The P2P network is built as a network of "full" nodes which are added as user hosted VPNs. These nodes are DIBS-enacted VPN router nodes which include the complete services required to build and operate the VPN network. Every node that is added authenticates itself to the network through accessing the blockchain ledger and then joins the existing network.

The nodes are either images which can run in a virtual environment or actual embedded hardware devices built on QLC Chain Node and customized to run the DIBS framework enabled applications. QLC Chain Node is "a Linux operating system targeting embedded devices that allows [users] to customize the device through the use of packages to suit any application."[1]

As part of the building and provisioning process of the image or complete embedded device, each unit (including the hardware nodes) has a public and private key created to represent the

[1] https://QLC Chain Node.org/

unit on the blockchain. The image includes the private and public keys to access their account via the embedded wallet logic.

The image is configured to run by default as a plug-and-play device. When booted, the device can get its network information through the dynamic host configuration protocol (DHCP), which includes its IP address, internet router, and name server. Alternatively, the device can be configured through a management interface to assign an IP address, netmask, and default router. The image retrieves its DIBS network information from a pre-configured list of “Bootstrap Nodes” from which it is able to read its initial configuration. The bootstrap node address can also be set through the management interface.

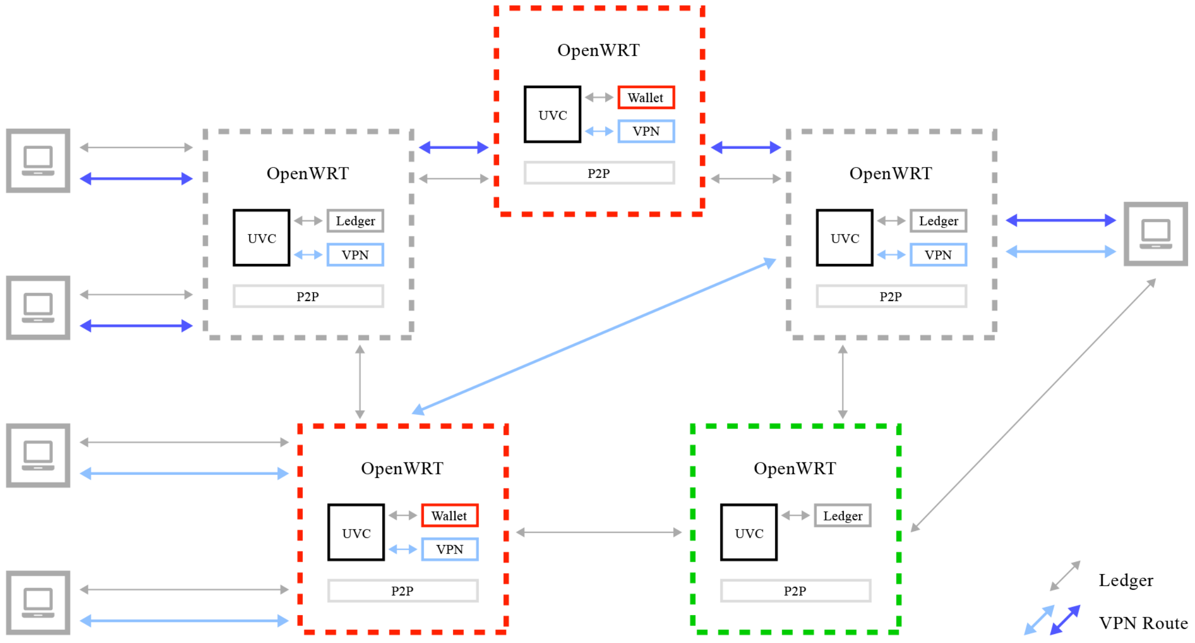

Figure 7. QLC Chain Node Network

In the network portrayed above (Figure 7), the:

- Full nodes maintain the ledger and synchronize with the other nodes.
- Light nodes can route traffic and initiate connections with remote VPN sites.
- VPN clients connect to the P2P network via a smart contract and token.
- Then, VPN client sends traffic to the P2P node as it does through the Internet (as its forwarding router).

5.1 Four Modules of DIBS-enabled VPN network built on QLC Chain Node

There are four modules of an DIBS-enable VPN network built on QLC Chain Node, which can be seen in the high level architecture for an embedded QLC Chain Node node is shown below.

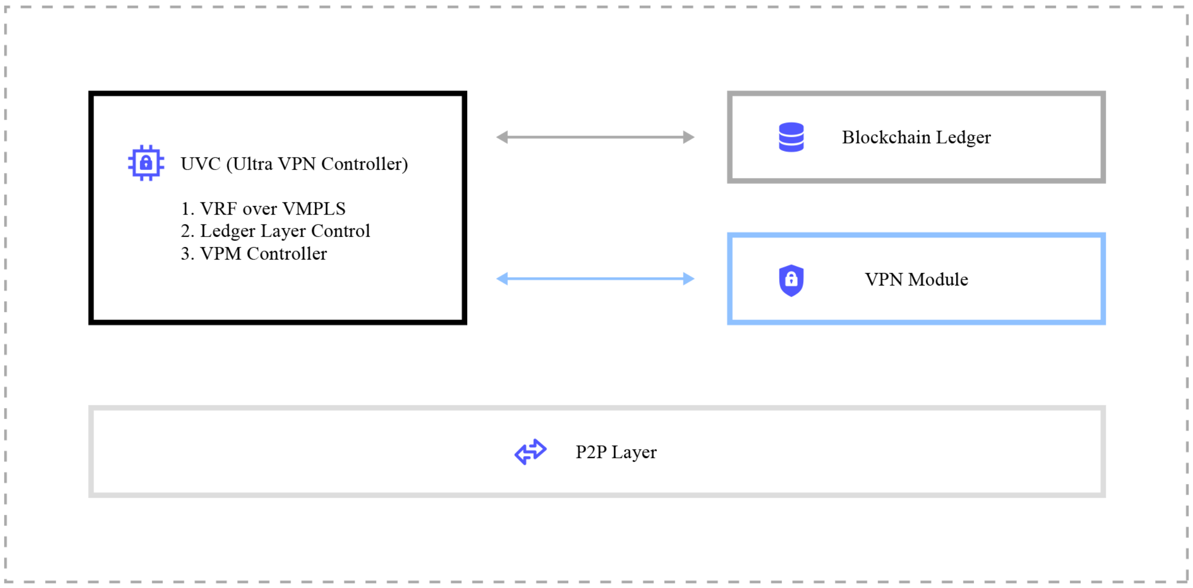

Figure 8. Four Modules built on QLC Chain Node

The modules of the DIBS-enabled VPN network include the P2P Layer, VPN module, blockchain ledger, and UVC (a module that controls the DIBS VPN).

5.2 P2P Layer

The P2P Layer includes the management of the addresses as well as the decoding and encoding tickets as the network is built. In addition, the logic for the P2P assesses the shortest paths through the P2P to reach a given address. The address tables of the P2P tracks the network addresses which are built through the VPN layers and allow forwarding through P2P as a standard routing and packet management.

Each full node that is part of the P2P network can host multiple organizations' networks. It does this through virtual routing and forwarding (VRF) functionality that allows individual networks access with their corresponding routing tables. Each instance of VRF will support a given network. Associated with each network is a VMPLS tag which is used as a forwarding vehicle within each actual VPN tunnel. The tag serves to isolate all traffic for the VPN traffic from the other traffic put through the network.

5.3 VPN Module

The VPN module builds and maintains the VPN connections between nodes. The VPN module runs OpenVPN and maintains end-to-end encrypted tunnels between the two endpoints. There is a client and server portion. The client is run from an endpoint device, and not in a full node. The site-to-site VPN is created between two nodes running OpenVPN server. The routing of the packets is handled at the P2P layer which the VPN module interfaces with.

5.4 Blockchain ledger

The Blockchain is the central architecture component. It provides the security associations between the parties and replaces the central directories and authorities used in traditional Entrprise security. In DIBS, the blockchain is used for account identification and smart contracts are used to create tickets for authorization and access to services. The DIBS system tries to be “blockchain neutral”— any blockchain that supports “smart contracts” can support the DIBS platform and application architecture. The initial launch will be on a specific set of public blockchains. DIBS provides SDKs and reference environments that will allow other blockchain providers to port and support the environment.

A common requirement of blockchain based applications is “anonymity”: the only information about user ids that is typically available is information pertaining to ownership of the transactions purely in terms of the cryptographic keys. These are typically wallet and token identifiers of the owners and assets, respectively. The applications described here require more information to allow access control outside of the identities in the blockchain. In the case of a VPN, someone who deploys the VPN wants to control who has access to the assets from a human perspective. There must be a way to map blockchain identities to user identities in a secure and useful way. Accordingly, DIBS defines its own set of initial applications that allow application providers to create the access control lists that can then be enforced while still maintaining the decentralized access of the blockchain.

The determination of identity for user or group access control is done as part of the server portion of the application deployment. A server “invites” users to join. It does this through an “invitation” application which encrypts the address of the server and a credential which can be given to the user. For example, a VPN provider can email invitations to users they want to allow to access to. These are encrypted in the server’s private key so only the users who are named can decrypt and load the packages into their applications. The user can connect to the server using the address in the invitation and then register with their unique “invitation” giving the information required by the invitation. The registration proves the user is authenticated to the blockchain, they were able to access the P2P network using the bootstrap methods, and that they received the invitation directly from the user. DIBSDIBSDIBSDIBSDIBSDIBS

5.5 Universal DIBS VPN Controller

The Universal DIBS VPN Controller (UVC) initiates the smart contract functions via the blockchain ledger. The controller interfaces to the blockchain ledger, paying for the execution of the smart contracts, and receives the tickets and tokens for establishing connectivity and the VPNs. The controller also starts the VPN server and communicates with the P2P layer with the relevant networking parameters for the VPN. Additionally, it controls the ledger layer and the VPN module to provide a VPN routing and forwarding (VRF) route over VMPLS. All of these functions are built upon the P2Peer network.

QLC Chain Node serves as the local management application that allows for the customization of the parameters in the UVC. The controller reads those values and starts the VPN server in the appropriate mode with the configured parameters. All the QLC Chain Node entities will construct together to form as an entire network.

6.0 Use Cases

The following are examples of the potential value of the DIBS framework.

6.1 DIBS for private network and traffic management

For example, a private network with identity for IoT. When we talk about connecting millions of sensors, are you comfortable with them running on the cloud?

	IoT on traditional cloud	**IoT on DIBS**
Existed provide	Cloudflare, AWS,	Network DIBS
Model	Centralized network using 3rd party security products Pre-provisioned keys and policy definitions for network security, encryption and data storage.	Decentralized network Nodes join P2P as they are run Encrypted communication, distributed authentication & network security
Cloud centric	Controls disappear outside of the cloud	Authentication and Access control throughout the network with clients and ledgers
Auditable	Service is not auditable	Service is auditable
Payment	Pay separately to the cloud provider, VPN providers, security providers	Pay as you go/on demand

Figure 9. IoT in traditional cloud versus the DIBS framework

Taking an example of IoT running inside hospitals, IoT may be part of many applications including medical devices for status monitoring, remote disease monitoring, drug management, location tracking and even alerting systems for the medical devices.

There have a few things in common:

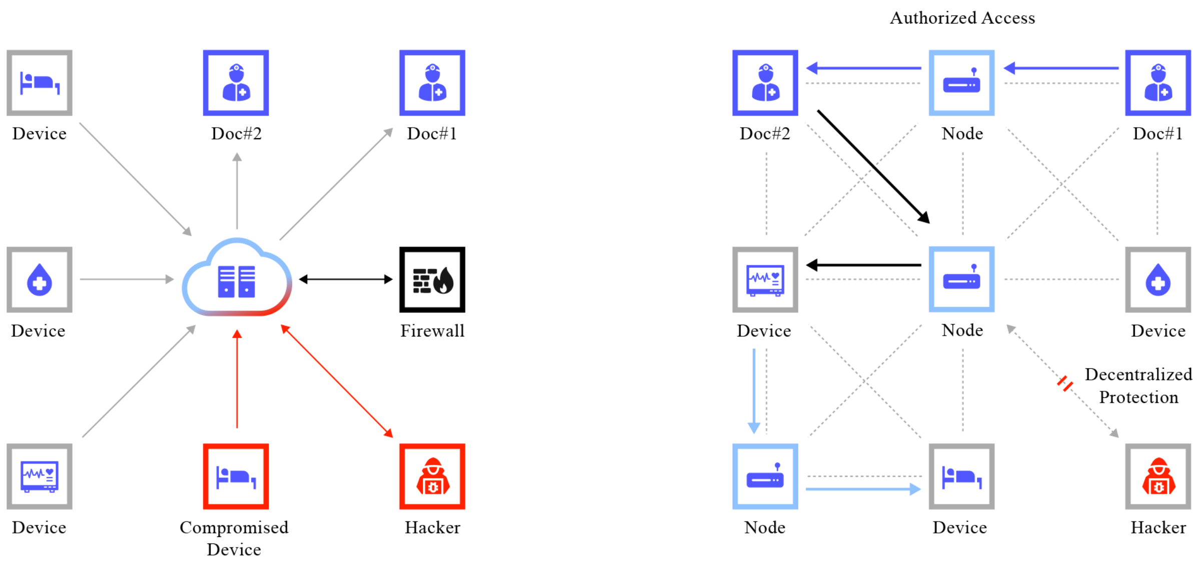

Figure 10. Cloud-based IoT versus DIBS VPN-based IoT

Using the above image (Figure 10), when an IoT network is established in a hospital, the network needs the following requirements:

1. Common network infrastructure for all parties
2. All users and devices must have access all the time so loosening of access controls
3. Security access controls are mostly through the firewall in the cloud
4. Opportunities for mistakes related to lack of security and privacy in the network

The hospital-based network is an example of a typical enterprise network which can potentially utilize the DIBS Framework for better network security.

6.2 Additional layer for network identity management

With emergence of public blockchain technology, various public chain's node provide the solution for scalability. Such projects tat could be successful in this achievement range from Ethereum, to EOS. One common requirement is that all public chain's rely on nodes.

Nodes have differing role that include block production, ledger maintenance, reaching consensus, and ensuring that the whole blockchain network works properly. Blockchain nodes are the most important guardian for maintaining the integrity of a blockchain.

Using EOS as an example, its block producers are still based on traditional IT infrastructure for communication and network security. Any network can be attacked, and blockchain nodes are no exception. The blockchain node is the centralized point of control on access, authentication, authorization and firewall.

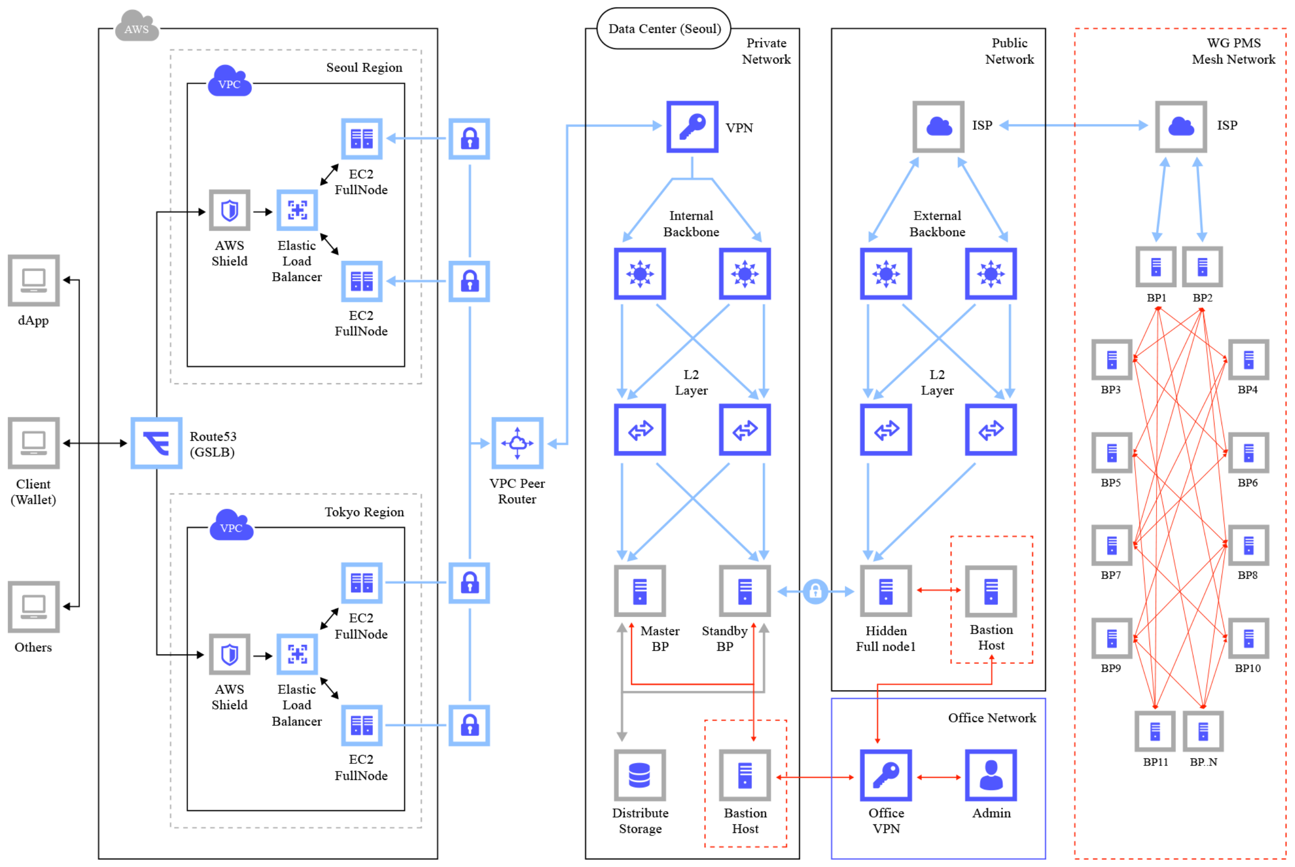

Figure 11: The EOS infrastructure

DIBS integrated into the blockchain and supporting software, transforms the network architecture for existing blockchains, by transforming it to an architecture built on the identities and security of the blockchain.

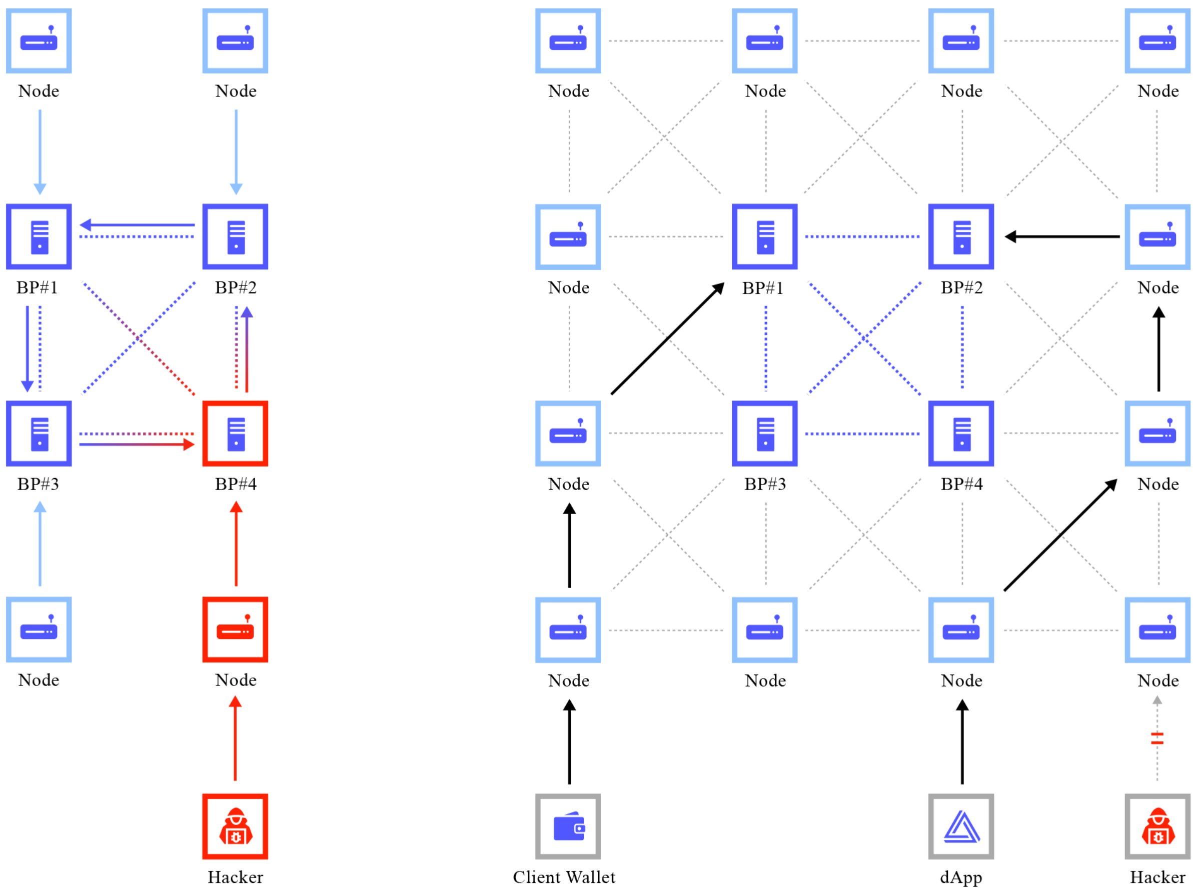

Figure 12: EOS BP (simplified) versus EOS BP running on Network DIBS

1. In accessing the blockchain the user is authenticated to the P2P network
2. No other access besides secure and encrypted access across the P2P network to the blockchain
3. DDOS traffic is not forwarded across the P2P network
4. No publicized address of the service to execute probes/DDOS

6.3 Cloudless P2P communication and Virtual Data Storage

Enterprises, investment banks, and investors often exchange investment document through 3rd party hosted virtual data room, the distribute login information to investors. Communication about these investment ideas and decision often are handled on instant messaging services such as WhatsApp, WeChat and Skype.

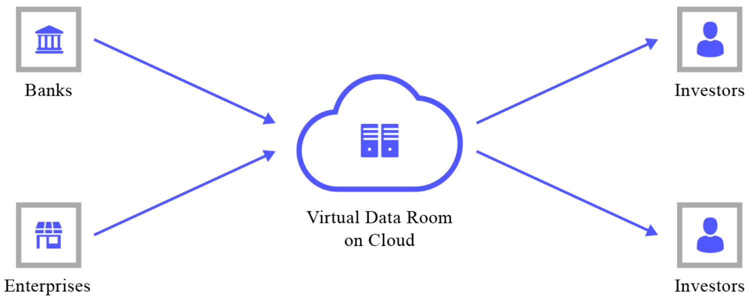

Figure 13: VDR and communication on cloud

Enterprise customers and banks will upload their information to VDR, whose services are hosted by companies like Intralinks. These services provide the investor with log in information, and ability to access information without the risk of data leaking and front-run by VDR companies.

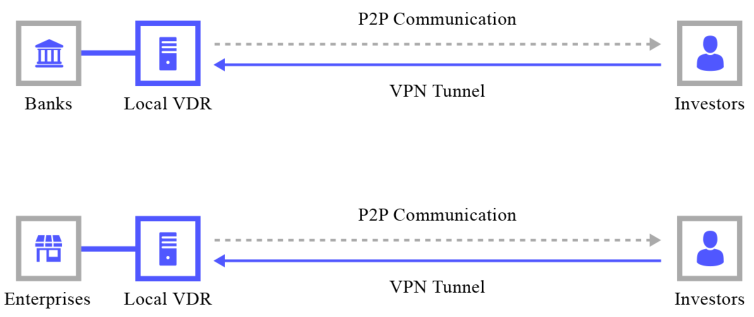

Figure 14: VDR and communication on P2P

By getting rid of cloud hosting, banks and enterprises can host their own VDR or establish secured and encrypted communication with VPN tunnels. Only authenticated and authorized investors can grant the access.

6.4 Security using NETWORK DIBS via smart contracts

DIBS support dApps to integrate with the DIBS framework SDK to enable authentication and authorization, as well as to delegate different rights to different accounts.

	dApp provided by Saas providers	**dApp authentication with NETWORK DIBS**
Existing providers	Open source and third party authentication, e.g. Dauth authentication	NETWORK DIBS
Model	Centralized account login and authentication encrypted to central services Control by SaaS providers Single points of failure	Decentralized network Authentication and Authorization independent of any individual service providers
Auditable	Service is not auditable	Service is auditable
Payment	No payment	Automatic with account and wallet creation and setup

Figure 15: Authentication via traditional SaaS providers and the DIBS framework

6.5 Replace Google login, with a new universal login for users

More and more apps use Google or Facebook for authentication. Some data is shared with Google and the app.

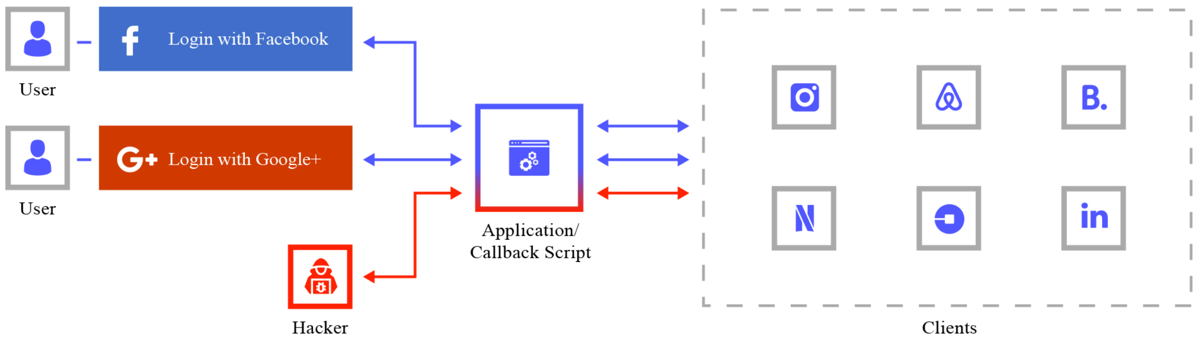

Figure 16: New Universal Login

1. Trusting security of the sites
2. Biases users towards those sites as they are used more
3. No real understanding of the use of data by those sites
4. No transparency to the site or user
5. Possible engineered fraud by inventing accounts at redirection sites

The process can also be used to replace the redirection of users to 3rd party sites for authentication (e.g. using Google OAuth).

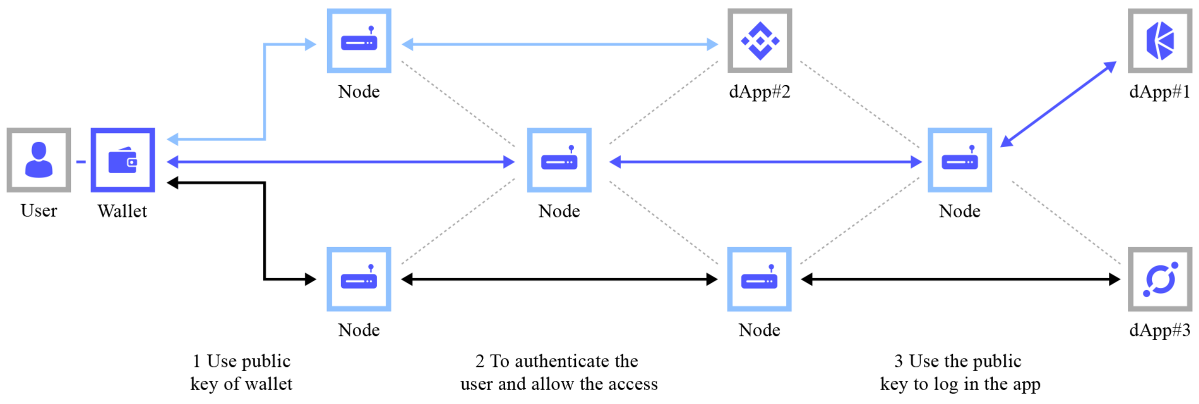
Node
dApp#2
dApp#1
User
Wallet
Node
Node
Node
Node
dApp#3
1 Use public key of wallet
2 To authenticate the user and allow the access
3 Use the public key to log in the app

Figure 17: Process for replacing 3rd party authentication

1. Identities registered and distributed via the blockchain
2. Cryptographically proven
3. Controlled by application providers
4. Verifiable identities and access via the blockchain ledger
5. All authentication is able to be audited, as transactions are recorded on the ledger

6.6 Telecom industry

Within the telecommunications industry a decentralized DIBS network solution can provide telecom operators with the ability to:

1. Securely deploy a variety of 5G services using the 5G AUSF, or Authentication Server Function.
2. Support new business models that involve complex resource sharing and roaming partnerships.
3. Use decentralized DIBS platform to support multiple wireless and fixed access technologies, including 3G, 4G LTE, 5G, Wi-Fi.
4. Boost performance and security by integrating multivendor legacy DIBS deployments in a decentralized cloud environment.
5. Enable network element inter-operability and easily accommodate different protocol implementations.

Conclusion

In conclusion, the NETWORK DIBS framework uses blockchain identities and security properties for authentication, authorization, and access in decentralized applications. The DIBS framework uses smart contracts supported by the blockchain to produce “tickets” that authenticate clients and authorize access to registered services. The capabilities described above are available to different public blockchains that support smart contract technology. The technology and security is open to any decentralized applications through APIs and SDKs, and can be used with existing P2P networks built as an integrated part of the overall solution.

The NETWORK DIBS framework provides a solution of authentication and access to the blockchain community. The hope is that the technology will be widely deployed and replace many of the proxy usages that rely on opaque security and privacy protections of popular social media sites (i.e. Facebook, and Google). The NETWORK DIBS framework provides identity verification with the same cryptographic strength as the records on the blockchains. It’s a decentralized authentication, authorization, and access that is protocol agnostic, and can be used in any blockchain.