BYZ GEN

ByzGen's DLT backed secure data management and exchange solutions

Keystone White Paper – Version 2.0
September 2020

Terry Leonard
Chief Technology Officer, ByzGen Ltd

CONTENT

ABSTRACT

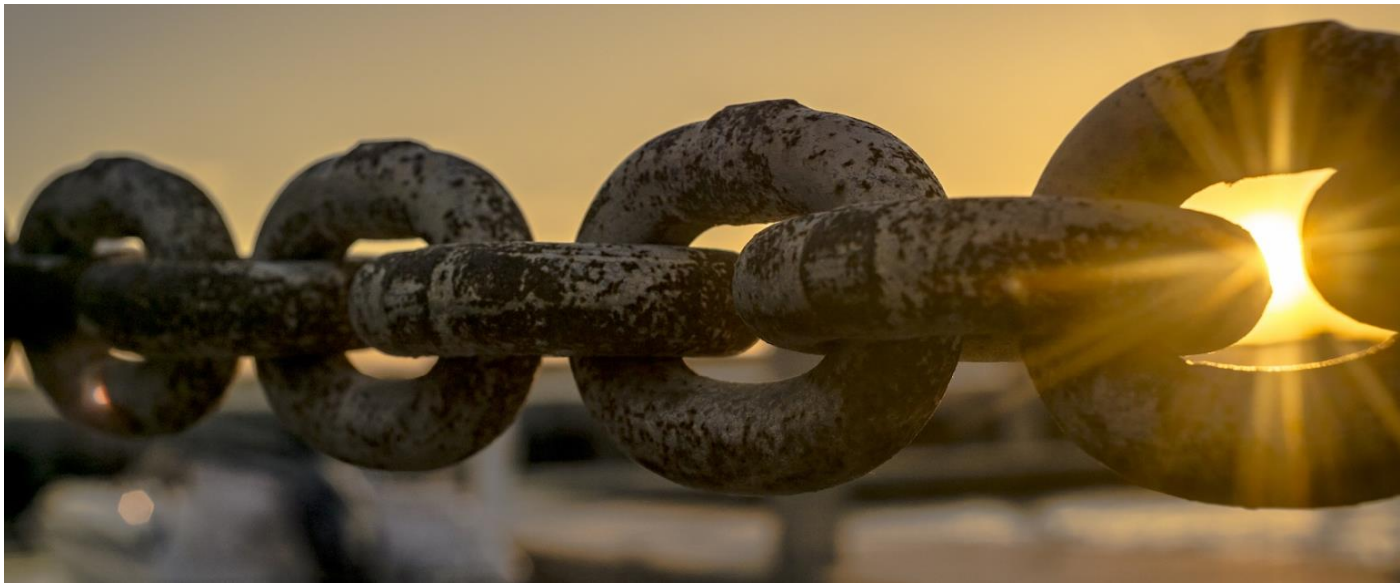
ByzGen have created a variety of bespoke DLT data security products across numerous sectors including defence, manufacturing, and insurance. The products were designed to secure and assure the user's most valuable data types within discreet business processes and data flows.

Assimilating the lessons learned from these projects with the industry challenges associated with adopting DLT. ByzGen created a fully configurable back end platform designed to integrate seamlessly with existing data management systems (e.g. ERP, PLM etc) and processes. The platform, called Falkor, is now live and Enterprise ready.

To date, the majority of DLT solution providers have focused on the Financial Service's Crypto-currency sector; ByzGen have chosen to focus entirely on the data-driven and data-dependent Enterprise market.

Falkor leverages the benefits of private distributed ledgers (replacing crypto currencies and tokens with business-critical data), using APIs and sector-leading security protocols to deliver a unique Trusted Data Exchange and Golden Record capability. The platform enables companies and organisations that have a critical need to exchange, securely and under control, their most sensitive and commercially valuable data with partners, across business units and throughout their supplier ecosystems; *inter alia* design IP, trials and test data, AI/ML algorithms and digital assets/value.

This Whitepaper represents over 3 years of product development and focuses on the overarching capabilities and functions of the Falkor platform. A series of Technical Briefing Papers have also been written, which complement this Whitepaper. Each Technical Briefing Paper is anchored by a real-world challenge that ByzGen have first-hand experience in helping to solve. As ByzGen deploy our platform to address new challenges, more Technical Briefing Papers will be released.



TYPICAL INTEGRATION AND DEPLOYMENT

ByzGen's technology is designed to solve multiple business challenges relating to data management, data exchange and the processes surrounding these activities.

Some real-world examples are described below:

- **Mutiple users from disperate parties.** Users need to exchange data and also write, read, update, link, and audit the data that makes up those exchanges.

- **Business Process Management.** Complex processes involving numerous actors produce and consume a myriad of data types and states. Users need to track this data in an assured way.

- **Use of multiple applications, infrastructures and devices.** Users occupying a complex digital and physical ecosystem need to be able to work with the same processes and data.

- **Opening Data Silos.** Legacy systems or internal organisational security procedures that cannot be economically replaced but need to be enabled to share and consume data in a secure but open way.

- **Effficient Treatment of Datasets**. Users need to exchange and work on datasets with varying requirements about how they should be secured, accessed and tracked. This differentiation must to persist before, during and after the data is exchanged.

- **Alignment of Pre-Existing Blockchains.** Users already have a discrete blockchain solution but need to add differing blockchain technology to the existing solution without creating unnessessary complexity.

- **Data Link Provenance.** Users need to prove the existence and status of ever-changing links created between datasets.

- **Decision Making and Analysis.** Users have a need to constantly analyse the latest state of data, digital assets, value and ownership; and make decisions based on that analysis.

- **Audit.** Users need to understand exactly where data, digital assets, value, and data ownership stands at a given point in time.

GENERIC DLT PLATFORM CONCEPT

Deploying Falkor allows access to **multiple** generically leveraged services and storage options, designed to be deployed in combination to provide a discrete solution to a specific challenge or use-case.

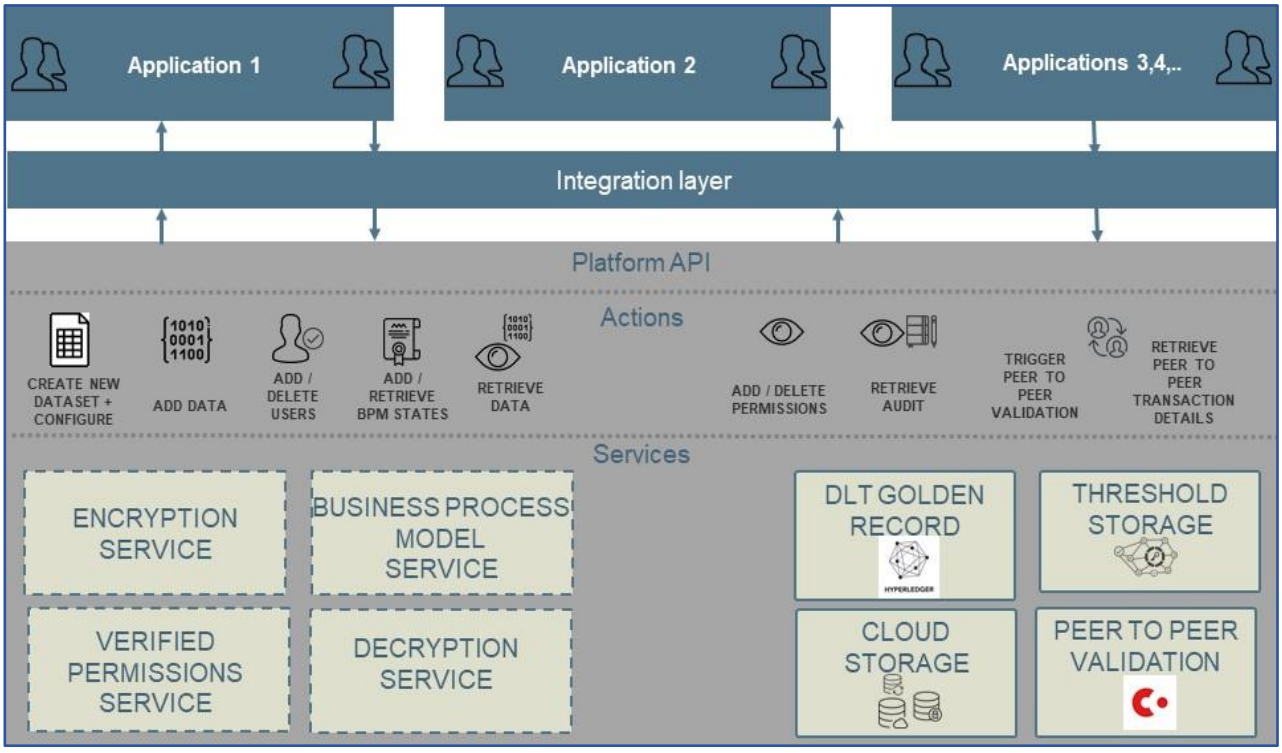
The generic platform architecture is shown below and is composed of 4 major layers:

1) The Application Layer

2) The Integration Layer

3) The Platform API

4) The Service Layer



The dev ops mechanisms in place allow services to be easily deployed individually or in combination. Each service has its own docker image and Kubernetes directory. The platform uses Kubernetes to orchestrate the images, which are hosted, built, and pushed via GitHub actions.

A particular user may wish to plainly track data on a DLT for audit purposes with no wish to encrypt or have separate read and write permissions for different datasets. This would be a simple deployment requiring only the core API and the DLT golden record.

On the other hand, complex enterprise users may wish to manage full data objects of differing types requiring: separation; high level encryption; decryption; audit and; control. They may also wish to trigger different types of data transaction (peer to peer vs golden record), between disparate parties that create and exchange within a scalable ecosystem. In this case the other services working around the core API and DLT golden record are deployed.

A single integration with Falkor enables enterprise wide coverage and scalability; meaning multiple business challenges can be addressed through a single deployment.

THE COMPONENTS OF THE GENERIC DLT ARCHITECTURE

Each of the four layers has its own components and functionality as described below:

The Application Layer

It is assumed that applications and user activity already exist in this layer and that there is a requirement to bring them together into an ecosystem leveraging some or all of the Falkor Platform's capability.

The Application Layer can have multiple disparate applications or devices. Each of these can have numerous parties and users forming a common ecosystem that needs to execute some or all of the following activities:

- Creation of datasets and data
- Control of data and versions
- Updates to data
- Reads of existing data
- The management of a business process that creates or consumes data
- Exchange of data
- Retrieval of data and process audit information
- Enforcing proper use of data
- Triggering an exchange of ownership of value, a digital asset, or data
- Validating an exchange of ownership of value, a digital asset, or data.

As part of the ecosystem's daily activity, certain datasets will be generated that need additional control, security, and validation; for example, the exchange of extremely sensitive data with a third party.

The application being used can automatically call out to the Falkor backend using business logic or ML to handle the data appropriately and enable secure data exchange.

The Falkor backend can also be activated manually if required. A user of one of the applications can create a new dataset requiring increased control, security and validation.

They set the configuration for this dataset choosing how data within it should be treated in terms of encryption level, permissions, and storage. Once the dataset exists, further data added to it is managed based on the configuration they have set.

Whether the process is automated or manual, the Falkor backend is designed to work seamlessly in the background to enable the following activity to be carried out at the application level:

- Each version of a data object will be tracked incrementally
- Links between data can be created and versioned
- Versions of data can be validated and retrieved
- Links and versions of links can be validated and retrieved without requiring decryption
- Permissions can be updated and identities added when an exchange is actioned
- Retrieval of audit for creates, updates, and reads at the version level (for data, links and permissions)
- Triggering a peer to peer transaction to validate a change of ownership of data, assets or value
- Retrieving the latest state of ownership for all data, assets or value

The Falkor Platform is scalable by design and can accept data from existing applications with the option to build and integrate new applications, flows and business logic when required.

The Integration Layer

This layer links the applications being used to the platform API. It can be accessed concurrently by multiple applications; translating application requests to formulate calls to the platform API.

The Platform API

The platform API is a gateway designed to receive calls from the integration layer and invoke the correct platform services; it is a RESTful service in the JSON data format.

The interface supports and provides endpoints for the following actions:

- Creation and configuration of datasets – options include encryption type, storage type and permissions **(POST method)**
- Creation of data within datasets **(POST method)**
- Updates to existing data **(PUT method)**
- Retrieve data by version or latest state **(GET method)**
- Create links between data **(POST method)**
- Update links between data **(PATCH method)**
- Addition of permissions for a dataset **(POST method)**
- Update of permissions for a dataset **(PUT method)**
- Addition of ID within existing permissions **(PUT method)**
- Deletion of ID within permissions **(DELETE method)**
- Retrieve audit of data **(GET method)**
- Retrieve audit of links **(GET method)**
- Retrieve audit of permissions **(GET method)**
- Trigger a peer to peer transaction **(POST method)**
- Retrieve transaction details of peer to peer transactions **(GET method)**

Each call needs to be authenticated via a HMAC generated API key. The API key is passed in the Authorization http request header for any call being made. Such a key is linked with a tenant in the platform. Internally, the platform recognises the tenant by the API key provided in the http request header.

The Falkor Platform can be used as a multi-tenant system. The concept of a tenant can therefore be a way of further separation if multiple organisations are connecting to the same instance of the platform. In this scenario a tenant ID needs to be passed in all API calls made.

The Service Layer

This layer holds the eight services and storage options the detail of which is forms the bulk of this whitepaper. They form the generic principle services that are called based on the type of action coming from the platform API.

As previously discussed, each service can be deployed separately and can sign to other services within the platform when required.

1 - Encryption Service

When creating a dataset in the platform, the Encryption Service can be activated and configured. Any data added to the data set will be treated in the same way unless the configuration changes. Several levels of encryption exist within the platform and are as follows:

No Encryption

Data added to the configured dataset is not encrypted before it is stored.

Platform Encryption

Data is symmetrically encrypted before it is stored. During encryption, the data is split into blocks with each block being encrypted separately using Authenticated Encryption (AE / AEAD).

The initial key material used to encrypt the data and linked metadata is then symmetrically encrypted and stored on the DLT golden record (validated as a transaction within a block). Note that the same authenticated encryption method is used within this step.

The 2nd symmetric key generated in this case is further encrypted via a master key and stored separately in the off-chain storage until needed for decryption.

Threshold Encryption

When reads of data within a dataset require an additional layer of security, validation and audit then threshold encryption can be enabled.

This follows a similar process to platform encryption, however once the 2nd symmetric key is generated it is encrypted via an ElGamal public key produced by distributed key generation amongst the threshold encryption nodes. This means that each read of the initial data requires distributed decryption and re-encryption to take place within the nodes (see threshold encryption section for further details).

This process generates a transaction that is validated within the DLT golden record for read audit purposes.

This generates flexibility for being able to encrypt data within a specific dataset in the most appropriate way before it is stored based on the sensitivity of data or business logic being employed. It therefore allows for multiple different dataset types to be constantly transacted at scale with the correct security applied to them.

In all these scenarios a checksum is computed over the encrypted data before it is stored. This checksum is stored on the DLT golden record and can be validated at the point at which the data is decrypted to further ensure the integrity of it.

For all symmetric encryption described here the platform uses a block cipher mode of operation that provides high speed of authenticated encryption and protects the integrity of the data until it is fully decrypted.

2 - Decryption Service

Any read request of encrypted data within the platform activates this service.

A request requiring the decryption of a dataset with full threshold encryption enabled, generates an ephemeral key pair specific to the request. Once the IDs and permissions for the unique request are validated within the DLT Golden Record (see permissions and DLT Golden Record services), the Threshold Encryption Service is signed to and is passed the ephemeral public key. The threshold encryption service then performs the decryption of the 2nd symmetric key and re-encrypts it via the ephemeral public key that it has been passed.

The encrypted 2nd symmetric key can only be decrypted via the ephemeral private key generated by the platform decryption service; once it is available it can be used to decrypt the 1st symmetric key and metadata that is being held on the DLT golden record.
Generation of the 1st symmetric key means that the data itself can be decrypted and made available to the caller of the platform API and the applications being used.

This process therefore provides validated access and audit at the exact point that the data decryption request occurs. The level of validation and immutable audit achieved is unique. This is due to the threshold encryption and re-encryption taking place; and the process itself generating a new transaction in the DLT golden record that can be retrieved.

Enabling platform encryption for a dataset generates a similar process except that the Threshold Encryption Service is not called. When a read request is validated then a decryption of the 2nd symmetric key takes place via the master key solution.

In all reads of data, the stored checksum for a given version is computed over the encrypted data that is retrieved in order to ensure the integrity of the data being read.

Similarly, reads of data will reference the latest global state held by the DLT Golden Record in terms of data, IDs, permissions, encryption keys and metadata and consider that state before triggering any decryption processes.

This service is vital for data exchange in terms of validating the data being read (down to version level) at the point of request; and the audit required in circumstances where data is being exchanged and worked on by multiple disparate users.

3 - Verified Permissions Service

Permissions objects in the platform are changeable data structures that are validated and stored on the DLT golden record.

The objects can be created at the dataset level containing concepts of owner, writer and reader of data held within the dataset. The identities of users (in the form of IDs) that can interact with the contents of datasets are also held within these structures.

Every time a permissions object is created or updated, a new transaction is created requiring validation via the DLT Golden Record. It is then stored as part of a validated block. This creates a constant “latest validated state” of the objects, that can be enforced for all write and read actions around the dataset. An immutable track of how

those objects have changed over time and who has actioned the changes is also created.

When exchanging data users can therefore maintain control over their datasets as they are exchanged. They can enforce policy around the datasets which can be ever changing with updates always being validated and tracked via the DLT Golden Record. Data owners can retrieve the immutable audit of the permission objects and also any historical changes made over time enabling better decisions about how policy should be managed.

The different concepts within the permissions data structures can be used to control and enforce policy in different ways.

Writer Concept

When there is an attempt to write something new to a dataset then the latest validated state of the permission structure is verified. If the ID of the user making the write request is present in the latest state of the structure then the write is processed, if not then it is rejected.

Reader Concept

A read of data can contain different identities to a write of data; giving policy makers flexibility to enforce which users can contribute to datasets and which users can read content only.

Owner Concept

Owners have the rights to set policy and create, update and audit the permission data structure. Regardless of concept update - a new transaction is always generated containing the newly proposed structure. The structure must be validated via the DLT golden record before becoming the latest state that all users and applications of the ecosystem are enforced against.

IDAM

This capability can integrate with existing IDAM solutions designed to manage users within the application layer. IDAM integration means that access management of users within specific organisations can continue to be managed by existing IDAM. Any existing policy will be enhanced to enforce, validate and audit based on the latest state of data, permissions structures and identities.

4 - Business Process Model Service

This service allows for business process models to be configured, stored, tracked, and executed by a combination of BPM tools and the DLT golden record.

The model can be mapped and changed separately to the DLT. Defined touchpoints to the DLT Golden Record can be established to track the status of data within the process and the DLT golden record will grow with latest and historical process data and state; this creates an immutable audit of process activity enabling trusted process automation via the Falkor Platform.

The development of this service deliberately avoided coding large, complex business processes that are stored and executed wholly on the DLT Golden Record. This kind of solution is often called “smart contracting” in the DLT space.

ByzGen have developed the Business Process Service differently because business processes can be complex and ever changing; and users do not necessarily what to make the entire process immutable on DLT Golden Record. In this case, every time the process needs to change there will be a requirement to transact the full process as code back onto the DLT. This adds unnecessary process complexity and inefficiency carrying the risk of older versions executing on the DLT (given they have been made immutable); particularly if the process is ever changing.

The approach of holding the process model element away from the DLT with enabled touchpoints to the Golden Record is a way of keeping things flexible and scalable.

5 - DLT Golden Record (Hyperledger)

The Golden Record holds all transactions that have been validated via consensus and by a threshold of peers running the ledger. Transactions are grouped into blocks and each block is cryptographically derived from the previous block thus creating an immutable chain of transactions. This “golden state” can change every millisecond and having an ongoing single source of truth is increasingly

important for data exchange - particularly when the number of datasets, data versions and third parties to be exchanged starts to scale.

For data exchange, multiple actions can generate a transaction to be validated and stored via the DLT golden record. The data itself can be stored on the DLT golden record, either plainly or encrypted depending on the configuration (see encryption service).

If data is transacted on the chain, then a unique document ID is created and stored. If a version of the same documentID is being added then the same ID will be used, but an incremented version will be transacted and stored.

If off chain storage is enabled, then meta data including a pointer for the location of the data will be encrypted, transacted, and stored on chain.

- Note that links between data is a separate concept held on the DLT Golden Record meaning no decryption of data is required to retrieve and prove those links. If a new link is created, or current links updated, then the new data structure containing the document IDs of the latest links will be transacted and stored.
- If permissions are enabled for a dataset then a changeable data structure will be transacted and stored. The data structure for permissions will include the concepts of owners, writers, and readers whereby ID's of users can be added and deleted based on policy or IDAM changes within the application layer. The new data structure that is transacted and stored is what is used to validate reads to data within a given dataset.

These transactions, their validation via consensus and their storage in sequential blocks creates a record that can provide:

- Verification at the point of data being requested to be read against the latest validated state of data before any decryption process is triggered.
- Proof of data integrity, data versions, and links between data without requiring decryption of the data itself.
- A cryptographically immutable (and immediately retrievable) audit of all data, data versions, actions performed on that data, permissions changes surrounding the data and the links and updated links between the data.

Another function of the DLT Golden Record is the signing and calling to and from other components in the platform when necessary.

For Corda DLT a secure bridge can be created whereby the result of peer to peer validations by 3rd party Corda nodes are created as transactions in the DLT Golden Record.

For threshold encryption, the DLT Golden Record is part of the process for the signing and validation of read requests. When a read request is triggered from the platform API an ephemeral key pair is generated that is specific to that request. A call is made to the DLT Golden Record with the request details; these are verified against the latest state held for the data and its permissions. Successful verification generates an approveID for the request.

The approveID is used to request for each peer to generate a digital signature certificate which are collated. A request is then made to the threshold encryption component which is verified by checking the digital signature certificate. The threshold encryption component will then perform the decryption and re-encryption of the 2nd symmetric key by the ephemeral public key for the request. This is how the material is passed back to the platform.

Note that the DLT golden record does not directly connect to the crypto nodes of threshold encryption. The platform takes care of the direct requests and signing that needs to take place between these services.

In some cases, the DLT Golden Record will also have validated and stored encrypted details that go alongside the main data object. For example, if off-chain storage is configured for a dataset then a pointer to where the data is stored off-chain will form part of the details that can be retrieved from the DLT Golden Record.

The DLT Golden Record used in the platform leverages and deploys Hyperledger Fabric 2.1. This fabric allows for a proof of consensus validation of each block of transactions by multiple peers; with

each peer carrying an identical version of the blockchain.

This type of validation is most appropriate for the generation of the ongoing Golden Record and validated state held within. If there is a data transaction type that is required to be validated into the DLT golden record, then Hyperledger 2.1 that will be used.

There are inbuilt dev ops mechanisms that allow for the initialisation of new Hyperledger 2.1 networks based on the requirements of the ecosystem.

Requirements can include:

- The number of private networks needed per tenant or project.

- The number of organisations needed to run within each private network.

- The number of peers that run within each organisation.

- The policy of each network in terms of the threshold required for the number peers from each organisation that must sign each block.

This enables flexibility maximum flexibility when considering the type of business and operational agreements that are in place for a given ecosystem; For example there could be a prime entity in the network that can be given weight in terms of the number of peers running within the specific organisation and the threshold needed to sign new transactions.

6 - Peer to peer validation (Corda)

Corda DLT is used when there is a specific data transaction requiring peer to peer validation. This type of transaction is typically triggered when there is an exchange of ownership that must be agreed for a given data object, digital asset or value; or indeed any transactions that should only be signed and validated by the parties involved in the exchange.

In this case the data, digital asset or value can be tracked continually on the DLT Golden Record. The latest state of the data, updates, versions, permissions, links, and accesses can all be tracked. If a point is reached whereby the exchange can go ahead then a peer to peer transaction can be triggered via the platform API.

When initialising a Corda network for a given ecosystem there will be a 3rd party Corda node created to map to the number of parties that can be involved in this type of transaction. As part of this initialisation each Corda node will have certificates and keys generated that allow for authentication between nodes of the network.

Each 3rd party node deployed can be separated from an infrastructure point of view, either via a separate cloud project or by being deployed in the infrastructure of the 3rd party itself. This is dependent on the ecosystem requirements and individual situation of the parties involved. A platform notary Corda node is also required to be part of the network; and is created so that transactions can be proposed and signatures from transactions can be collated.

Once the network is initialised, Corda smart contracts are used to ensure the correct flow is in place for all transactions triggered for peer to peer validation. This is based on the ecosystem's business requirements and processes.

In all cases the Falkor Platform operates an asynchronous flow. Transactions are proposed and the platform waits until all parties involved sign. A signature is generated by the 3rd party Corda node representing that party.

A typical flow for this scenario could involve a trigger from the application layer that a discreet subset of data is ready to go through peer to peer validation processing. The integration layer will then confirm with the specific peer to peer API endpoint within the platform API, what data from the DLT Golden Record should be part of the transaction (i.e. the documentID) and what 3rd parties should be involved in the validation of the transaction.

The platform peer to peer service then processes the trigger. It authenticates to the 3rd party Corda nodes that represent the parties defined as being involved in the transaction and proposes a transaction to those nodes. The platform will then wait for signatures from all parties involved in the transaction before any further action takes place. The signatures are sent to the platform notary node which collates responses.

At this point there is an option for a separate business process for accepting the transaction to be inserted into the flow. An ongoing list of transactions can be made available to a party via the application layer based on the fact the system always knows the latest state of ecosystem transactions.

Transactions can therefore be extracted and made machine readable via an application. From here a specific party can explicitly accept the transaction via the application forming the trigger for the associated 3rd party Corda node to sign and send the signature to the platform Corda notary node. Once the platform notary node receives all signatures required, then the transaction is fully validated on the platform.

Validated transactions are stored in the local state data base on the 3rd party Corda nodes involved. No other Corda node in the network will be aware that the validation has taken place.

The next step in the process is a call out from the platform peer to peer service to the core API. A new document is created within a specific dataset that confirms to the DLT Golden Record that the item of value has exchanged ownership.

The secure bridge created between a Corda transaction and the DLT Golden Record means that any user that wants to view the latest state of the item of value can do so without being involved in the acceptance of the transaction themselves.

The secure bridge can be further enhanced by the other platform services described in this document in the following ways:

Threshold Encryption

The documents used to process the result of a Corda transaction on the DLT Golden Record can have full encryption enabled adding additional security and auditability.

Permissions

Datasets containing the results of a Corda transaction can have permissions enabled and a permission structure enforcing read/write permission to defined users enabling rigid management of who can/cannot view and/or alter the results of sensitive transactions.

7 - Cloud (Off-Chain) storage

Off-chain storage can be enabled when data objects should not or cannot be stored directly on the DLT Golden Record. Note that with off-chain storage enabled and a data objects final resting place is off-chain; there will still be transactions around the object that are validated onto the DLT Golden Record that enable the overall capability e.g. the latest and historical state of versions, permissions, links and the pointer to where the data is stored.

Off-chain storage is typically configured for one or more of the following reasons:

- *Size*

  Transacting multiple large data objects can bloat the chain and impair performance.

  If the size of the data is over 2KB the recommendation is for off-chain storage to be configured.

- *Personally Identifiable Information (PII)*

  Under GDPR legislation you must be able to delete personal data on request from the owner. If you are transacting the full data object on DLT you are making it cryptographically immutable and it cannot be deleted in accordance with the legislation.

  The solution is to separate out what the blockchain should be transacting and the storage place of the data itself.

  Personal data objects can be encrypted and stored within the configured off-chain storage option. The DLT Golden Record is only used to transact documentID, versions, links, latest permissions etc.

  All transactions that need to be validated and audited still take place on the DLT Golden Record, but the data object is separately stored and can be deleted leaving a full history of all data transactions around the object.

- Commercial Sensitivity (e.g. Intellectual Property)

As with the PII sensitive data there is good reason to encrypt and store data off-chain in the commercially sensitive dataset space.

Ultimately though the decision to enable off-chain storage sits with the ecosystem requirements and the users within it. The admin API allows new datasets to be created at scale with different configurations including the storage option.

Users can configure a new dataset upon creation so that any data object added will be stored within the off-chain storage component.

This enables significant system flexibility at scale as the number of datasets grows. The ByzGen recommendation for data that is sensitive (either personally or commercially) is to configure the dataset to have off-chain storage, threshold encryption and permissions enabled.

Off-chain storage can be any cloud or on-premise storage solution (or combination of solutions) that the platform can be configured to authenticate against.

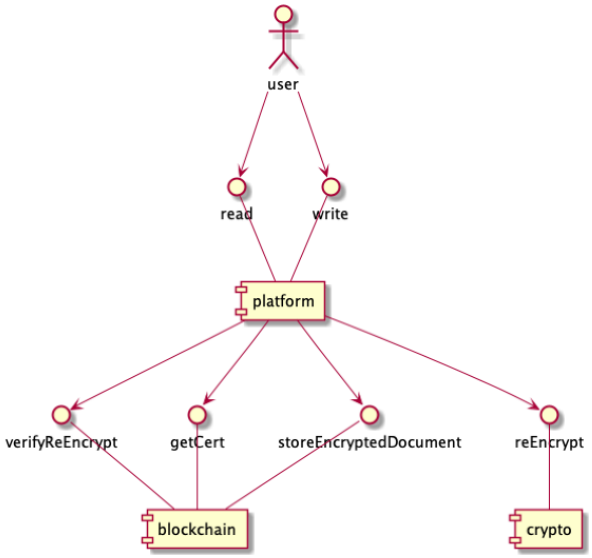
Configuring a cloud-based bucket for this option is the preferred option for many enterprise customers. This reduces costs significantly and means that if there is a certain dataset a user wants to move to the cloud, the platform ensures the data is properly treated (encryption level, permissions etc.) as it makes that transfer.

In this context the Falkor Platform facilitates the secure movement of datasets to the cloud.

8 - Threshold encryption (storage)

When enabled this service provides an additional security, access validation and audit layer for datasets. The service runs its own set of distributed crypto nodes. These nodes are separate from the DLT Golden Record and Corda nodes and there is no direct connection between them. The platform co-ordinates requests to and from these services and ensures that the correct processing and signing is taking place to realise the overall capability - and provide the bridge between them.

The diagram below shows this interaction between the services:



As discussed previously it is the DLT Golden Record that is responsible for storing encrypted data (keys, pointers etc.) and providing the validation necessary for a full decryption to take place.

When access to data is validated, the DLT Golden Record issues certificates that are used to perform operations on the threshold encryption services and the crypto nodes that work within it.

During read request processing, certificates are passed from the issuer (DLT Golden Record chaincode) to the threshold encryption service and the platform is responsible for the requests and the processing of this workflow.

Writing data with threshold encryption

If enabled, the threshold encryption service will be called as part of a write request to the platform.

This service will be called once the data object has been symmetrically encrypted and a checksum computed over the encrypted data object. A second symmetric key is generated to encrypt the first symmetric key with the encrypted data object and other encrypted material being stored based on the configuration of the dataset. Note that the 1st symmetric key will always be stored encrypted on the DLT Golden record.

The 2nd symmetric key is then encrypted further via the threshold encryption service. The service will generate a public ElGamal key (with Curve25519) for threshold encryption. This key is generated via a Distributed Key Generation (DKG) protocol via the crypto nodes of the threshold encryption service. Once encrypted via this key and stored, the 2nd

symmetric key cannot be decrypted unless the crypto nodes of threshold encryption reach a threshold to generate the private element needed to decrypt it.

To decrypt the data a threshold needs to be reached by multiple distributed nodes. Therefore, no single entity or node can be responsible for decryption; ensuring the additional level of security provided by this service.

This is further enhanced by the requirement that any read request being made to the threshold encryption service needs to pass certificates issued by the DLT Golden Record peer nodes. These belong to multiple organisations, so it is not possible to gain access to data without getting control of the majority of organisations participating in the network.

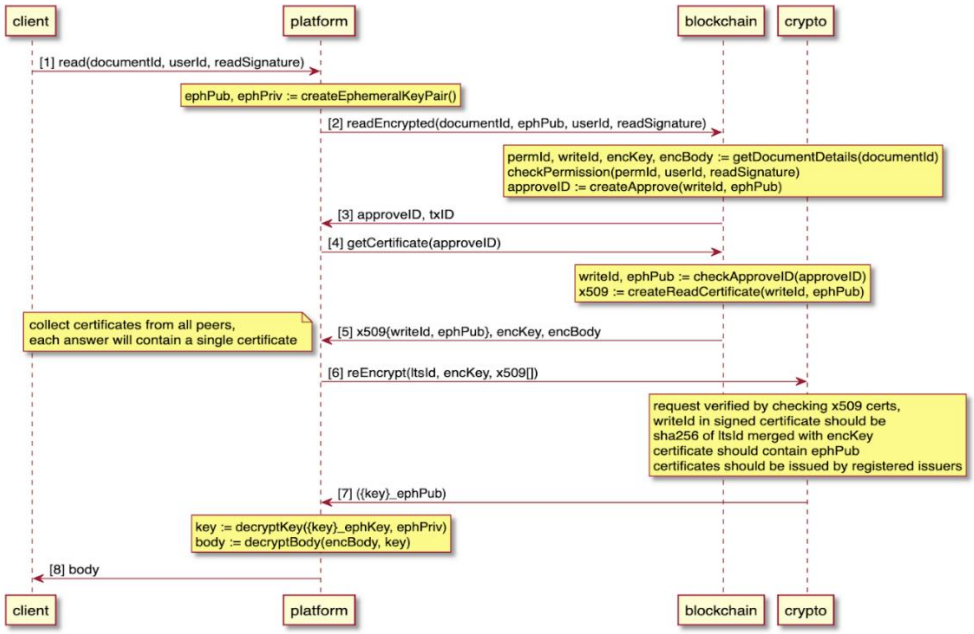
Reading data with threshold encryption

The threshold encryption service is called during a read request for any data that has threshold encryption enabled. As discussed, this data will have a 2nd symmetric key that is encrypted via the ElGamal key generated by the crypto nodes.

Note that before the service is called during the read process, some steps have already been completed including; the generation of an ephemeral key pair unique to the read request, a validation against the latest identities and permissions held on the DLT Golden Record and the request for certificates to the peer nodes of the DLT Golden Record in multiple organisations.

The service is then called and the crypto nodes decrypt the secure data (in this case the 2nd symmetric key) and simultaneously encrypt it by the ephemeral public key delivered as part of the request.

The process of decrypting and re-encrypting data is performed in a way that guarantees that any single crypto node cannot reconstruct the original data. To get this information, the private ephemeral key for the given read request is required. This private part of the key is never sent to crypto nodes.



The re-encryption flow that involves the platform, the blockchain (DLT Golden Record) and the crypto nodes is displayed in the figure above. It begins with a read request from the user side via the platform API containing the documentID , a version of the data (if applicable), the ID of the user trying to read the data and a signature which can prove the origin of such request.

When the read attempt is received by the platform a new ephemeral key pair is generated that is specific to this individual read request. This key pair is used only for this unique request and is then discarded by the platform.

The ephemeral key pair generated is an asymmetric elliptic curve crypto key in ED25519 form. The private key of this pair will stay in platform memory and will never be stored.

Furthermore, it will never be transferred outside of the platform to any of the other services. At this point the platform sends a readEncrypted request to the DLT Golden Record containing a documentId, dataset, version, tenantID, ephemeral public key, userID and signature.

This starts the DLT Golden Record validation processing. Each peer node receives the same input (readEncrypted request) and each of them verifies the latest read permissions to the data being requested by the user.

When the read attempt is successfully verified each peer node is creates an approval for this request by creating an internal document (an approval of the re-encrypt operation). All peer nodes (or a threshold of them based on policy) create this approval in the same way and all of them propose it as a transaction for a next block in the chain.

The response from the peers contains an approveID and the encrypted document body and symmetric key which can be used later to perform a full decryption of the data. The platform sends the approve ID to all peer nodes. The chaincode on the DLT Golden Record peer node verifies the approve ID and creates a certificate, which will be used by the crypto servers of threshold encryption. This is a regular x509 certificate containing a writeID and the ephemeral public key previously generated by the platform. Both values are stored as x509 extensions and each DLT Golden Record peer node holds its own private key (and self-signed root CA certificate). Each peer plays the role of certificate authority and each of them can issue an x509 certificate. All of this logic is handled in the platform created chaincode running on the DLT Golden Record. In the end the platform collects several certificates issued by the DLT golden record peers; each certificate will contain the same writeID and ephemeral public key and each certificate will be issued by a different peer. These certificates are valid for only a short period.

The collected certificates, the additionally encrypted key and the ID of the threshold encryption service installation is sent to that installation of the service. The Threshold Encryption Service then verifies the request and propagates it to the crypto nodes. The crypto nodes then verify the certificates (i.e. verification of the certificate issuer, writeID, validity time). Depending on the threshold configuration, a minimum number of certificates are required for the crypto nodes to accept the re-encryption request. When this verification is successful the crypto nodes convert the encrypted 2nd symmetric key to the key encrypted by the ephemeral public key.

As described the Falkor Platform is the only holder of the private element of the ephemeral key pair, so only the platform can decrypt material received from the Threshold Encryption Service. Once the symmetric key is decrypted it can be used to decrypt the material for data held on the DLT Golden Record. Each re-encryption will generate its own transaction within the DLT Golden Record that can then be retrieved via an audit request.

In summary, this service and the processing and transactions it generates is used in data exchange to provide an additional layer of security and audit. If a user has a particularly sensitive dataset (personal, commercial etc.) the recommendation is for Threshold Encryption to be enabled.

This is particularly pertinent when data is going to be exchanged and used by disparate 3rd parties whom not everyone in the network may know or trust.

HOW BYZGEN WORK & FIT INTO THE DISTRIBUTED LEDGER SECTOR

There are some key values and approaches that ByzGen hold to when developing solutions in the DLT and primarily blockchain space. These have shaped our platform capability and associated client engagements:

- Investigate multiple solutions in terms of DLT and blockchain and have no motivations to back just one of them and be too dependent and inflexible as a result. The route taken has been to gain knowledge of what each potential solution could bring to the platform and integrate those as part of the wider capability if they can provide a real differential for use cases and clients of the platform. These solutions are integrated in a way that they should work alongside the other components of the platform and be deployable via the central dev ops mechanisms of it. This has led to us working in the academic research space and leveraged our threshold encryption service from that and we have also integrated the ever-maturing open source solutions of Hyperledger and Corda.

- Each of these bring different and required capability to the platform. The threshold encryption component uses the concept of a distributed network for validation but looks at it through the lens of multiple nodes being involved in the encryption, decryption and re-encryption of sensitive data and therefore provides a unique level of security, validation and audit around sensitive datasets. If this level of capability is available, can be leveraged in a non-complex way and is performance ready then there would need to be a question of why it would not be used by a client. The Hyperledger component gives us the proof of consensus validation and block creation that provides the benefits of the DLT golden record described in this document. The Corda component gives the option for triggering a differing type of data transaction whereby an exchange or change of ownership is required to be validated by only the parties involved in the transaction.
These individual components are brought together via in-built platform development and are under the control of the platform rather than any third parties e.g in the scenario where corda needs to transact results in the DLT golden record, the golden record needs to provide proof that a read request is valid before generating approval certs for use in the requests to the threshold encryption component.

- A primary approach is to have the platform be a non-complex gateway to multiple DLT solutions, the differing capability they bring, the bringing together of them and the running and deployment of them. This means a client does not need to be burdened by the complexity and undertake development for the integration to each solution. They can instead leverage the platform API in a generic way and serve multiple projects and use cases they have enterprise wide. This is in particular relevance when you see the different data transaction types that are required to be supported and the need to leverage different DLT solutions to provide the golden record versus the pure peer to peer validation for certain transactions.

- Build non DLT components that can work agnostic to DLT solution that are proven to be required (outside of generic blockchain) during the build of real use cases and solving of real business challenges. Examples of these are to have encryption and decryption processes before and after data is stored, dataset configuration that is scalable and flexible, enhanced security and audit for reads to the data, being able to create, update and store links between data and other properties without needing to decrypt the data itself.

We believe that generic blockchain solutions only are not a sole solution to many use cases in themselves and these other non-DLT components need to

operate around them. Also, not one generic blockchain solution provides all the capability and therefore if multiple of them are to be leveraged then these other non-DLT components will need to deal with requests between them and the secure bridge of those requests and data that passes between them.

Part of this is always to question what processes and data should and should not be transacted on DLT. Once you have transacted data onto DLT you have made it cryptographically immutable and this can be of great benefit when you want to validated against a latest state of data, permissions, track versions / links and produce an audit for those. However, in the scenario where there is personally sensitive data (under GDPR legislation) involved then transacting that data directly to a generic blockchain solution makes that data immutable and therefore it is unable to be deleted which goes against the legislation. Also, in the case of a business process model there is care required when attempting to code the whole process and make it immutable via smart contracts on a blockchain, this can cause problems when that process changes and there is a potential that an old version of the process could execute.

With these non-DLT components the platform can also be used to solve the challenge of different organisations who have chosen and implemented certain blockchain solutions and are therefore potentially creating more data silo challenges when trying to make data exchanges with 3rd party organisations who have made a decision to implement another blockchain solution. As proven with the processing between corda and the DLT golden record through the platform then these components can be used to bridge the gap between use of different blockchain solutions.

- We work for our clients. This is multi-faceted in approach and means that platform development is based on proving against real use cases and business, there is no bias on blockchain solution, focus on taking complexity away from the client, to prove and ensure new functionality can be deployed agnostic of cloud infrastructure and produce generic platform functionality.

- Be an enterprise wide capability and therefore always consider scale, flexibility, performance, and deployment when developing new capability.

- Continually research and innovate around new capability that can be tested, proven and integrated with the platform and make them readily available