Luther Case Study

Resemblance

Detection of fraudulent similar claims with data privacy constraints across insurance carriers

AXA AIA Luther

1. Introduction

The Reinsurance group of America found that 3-4% of all claims are fraudulently resulting in fraud-related losses, equivalent to 3 - 6% of total expenditure[1]. Fraud and abuse take place at many points of the insurance process and the costs of such an issue have been estimated by the Coalition Against The Fraud (CAIF) as around $80 Billion per year[2]. The result is not only a higher cost of care but more importantly higher insurance premiums for all consumers. Fraud and abuse take place at many points of the process and the prevalent two consequences are:

- Higher billing - bill services not rendered, unbundling services
- Similar claims issued

What makes similar claims hard to detect is Insurers cannot legally share commercial data with competitors or others and disclosing personally identifiable information presents a privacy violation. The absence of a collaborative and automated product for similar claims identification with data privacy protection has been a real barrier for AXA & AIA to address this problem. This is why AXA & AIA have partnered with Luther to build this product for a safe and privacy-wise collaboration.

The **Resemblance** product can:

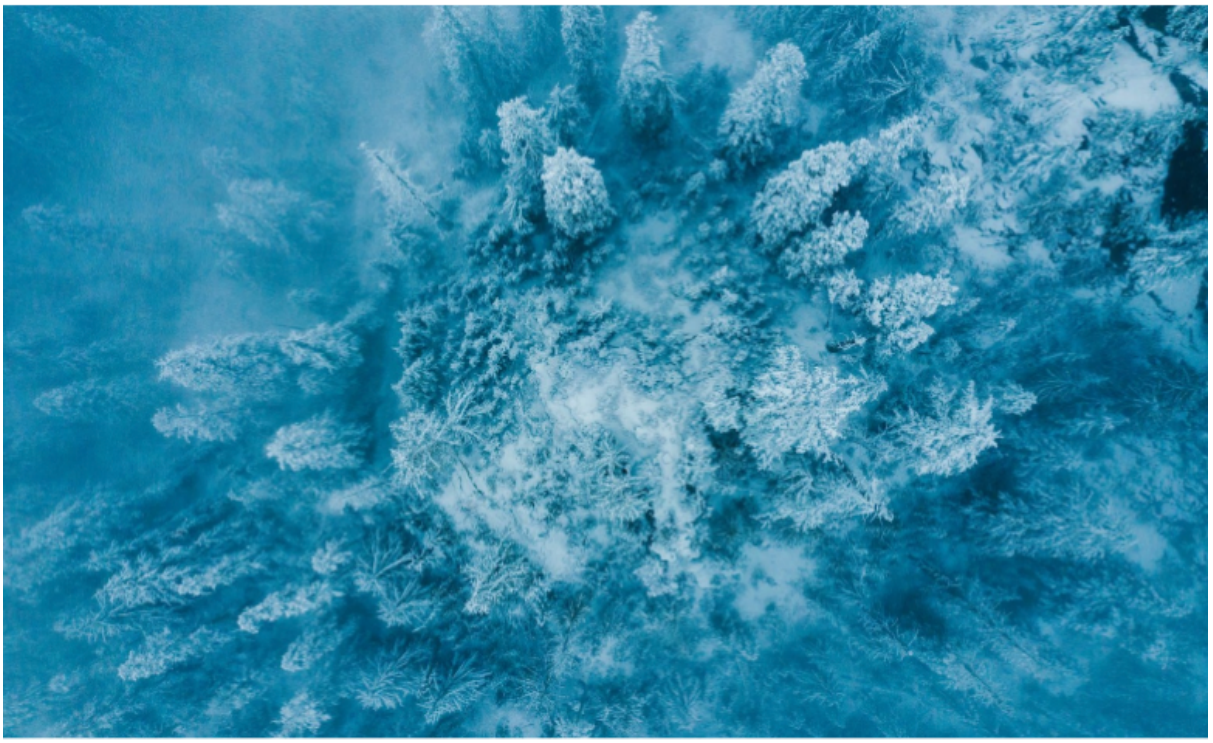
- Identify similar claims
- Ensure data privacy between all the participants of the network
- Ensure data storage is GDPR Compliant
- Identify similar claims in less than 1 sec
- Create a network that is easily scalable to add new insurance companies

There are similar use cases in many industries which require a network of participants to detect similar documents across each participant's private repository. For example, private watchlists containing the profile of suspicious individuals are maintained by separate land, sea, and airport authorities. Further examples include procurement, mortgage, tax, motor finance, and insurance coordination of Benefits. Resemblance is a technology that was initially developed for Claims fraud, and it allows the match detection between documents using a novel "Multi-signature hashing" method. It achieved the following results and is a foundation for collaboration with privacy and security constraints.

$2.7 M costs reduction for fraud	**50%** Operating costs fraud audit	**90%** less processing time	**10X** estimated ROI

[1]For more data: https://www.rgare.com/knowledge-center/media/research/rga-2017-global-claims-fraud-survey

[2] For more data on insurance fraud - https://www.iii.org/article/background-on-insurance-fraud



2. Existing similar Claims Detection Process

There are no viable solutions available on the market.

3. Problem

A network of participants requires the ability to detect matches on documents across each participant's private repository. In the case of multiple insurance providers, the need is to detect fraud by determining if the same or a similar claim was filed with another insurer.

An example that can help to understand the importance of such a tool. Jeff has an accident in Germany during his holidays. Jeff is insured by company A in Spain and his wife Kelly is insured by company B in Spain. When they are back they can file two claims with their two insurance companies, neither of them will know that they are reimbursing the same incident. This is a type of Fraud that with a similarity check can be detected and properly addressed.

In many cases, the participants desire to keep the contents of their repository private from the other participants. Additionally, the document checked against these private repositories should also be kept private. The documents themselves sometimes contain Personal Identifiable Information (PII) and are subject to data storage and processing requirements to ensure user privacy (compliance, regulation, sensitive data). The documents across the repositories may have slight differences yet still be considered similar to another, which requires systems to include a document similarity metric that detects similar documents. When a similar document

is detected, then the system needs to determine which participant contains the similar document to perform subsequent processing (e.g., notify the corresponding participant's fraud department or coordinate payments).

These applications must be secure, meaning that malicious participants and external actors must be detected on the network, their attacks must be mitigated, and they must be promptly removed from the network. To avoid deception, these applications must prevent participants from discovering documents that they do not own. In particular, the system must provide safeguards to ensure that querying participants can only query for authorized documents (e.g., documents similar to those that are already in their possession), to prevent external participants from determining the contents of the private repositories. Our solution design provides an ideal tradeoff between two extremes in the design space, explained below.

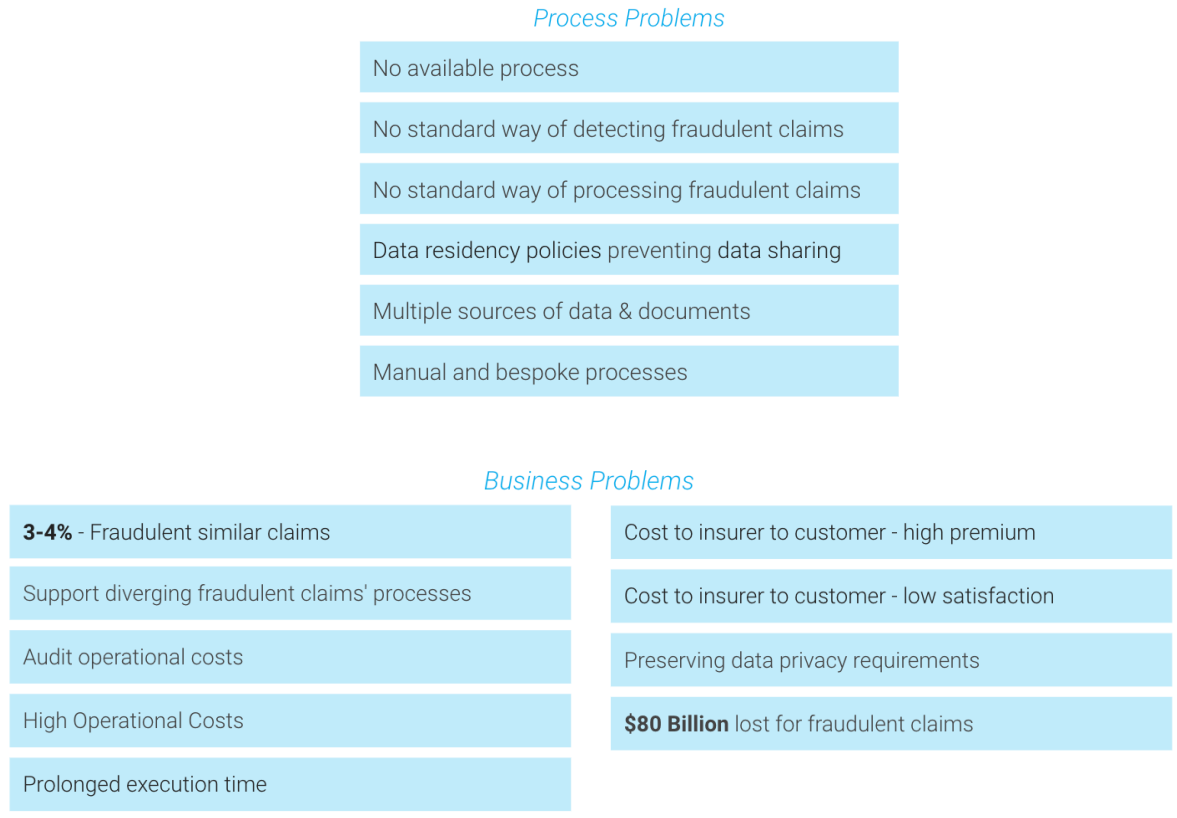
One extreme design is to share all the raw insurance data:

- Very strong fraud Detection
- Zero Privacy

Another extreme design is to not share any insurer data (as is today):

- No fraud detection
- Maximum Privacy (no one sees anything)

The solution provides a powerful algorithm for fraud detection while also providing a high degree of privacy.

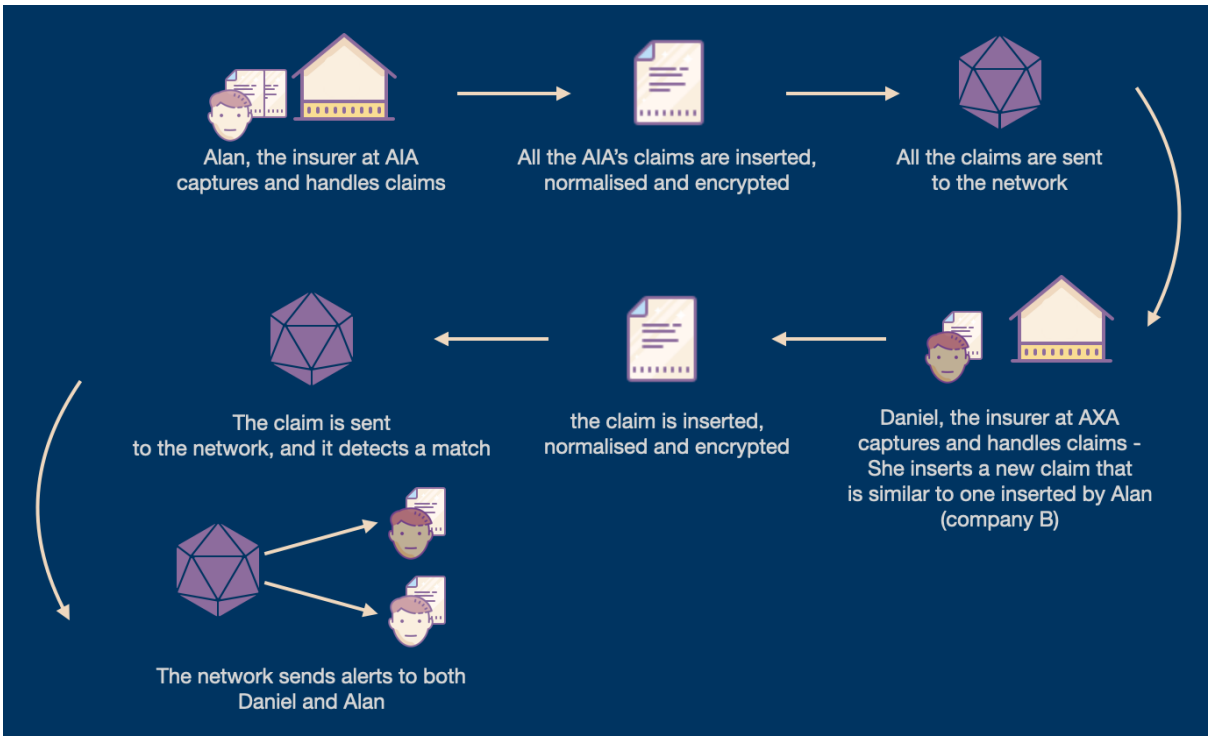


3.2 The newly introduced automated process

Together with AXA and AIA as partners, Luther drafted a new process to address and automate the problem. The detection of similar documents provides a safe and useful tool for insurance companies to decrease the number of fraud claims and improve customer services, and ultimately provide lower insurance premiums. These policies are usually sold by individual Insurance companies for any incidents stipulated in the policies. Below is the description of the process between 2 specific Insurance companies.

This process can happen between any 2 of the companies across the network:

- **Insurance companies handles and submit claims (company B in the diagram below)**:
    - All the companies submit claims
    - The solution will normalize and encrypt the data locally within the insurance company.
    - All the encrypted claims are sent to the network in real-time, where they are saved for 90 days.
- **Insurance company A captures and handles the claim (The case of similar claim)**:
    - Once a claim is submitted the insurer will start to handle it.
    - The system will normalize the data and encrypt the claim
- **Resemblance Network**:
    - Once the claim is encrypted the system compares it with the other encrypted saved claims in the system. If a match is found then the system flags this to the company who filed the claim.
- **Both Insurance Companies A and B**:
    - The insurers who issued the two similar claims receive, review, and approve the allegations of matches, and decide how to handle the flagged claim.


Alan, the insurer at AIA captures and handles claims
All the AIA's claims are inserted, normalised and encrypted
All the claims are sent to the network
Daniel, the insurer at AXA captures and handles claims - She inserts a new claim that is similar to one inserted by Alan (company B)
the claim is inserted, normalised and encrypted
The claim is sent to the network, and it detects a match
The network sends alerts to both Daniel and Alan

4. Objectives

The objective is to create an object match detection process usable across industries, which is fast, standardized, simple, and low-cost to operate. Align all the insurance companies across the world, by decreasing the number of fraudulent claims through high standard safety levels and privacy-respected collaboration networks.

This requires:

- Identify similar claims
- Ensure data privacy between all the participants of the network
- Ensure data storage on the Blockchain is GDPR Compliant
- It can identify similar claims, in less than 1 sec
- Create a network that is easily scalable to add new insurance companies

5. Solution

5.1. Solution overview

Deep Process Automation using the Luther common claims repository.

Building the world's best international common claims network, best for insurers and best for customers, required automation that could handle the inherent complexity of a multi-insurance process involving hundreds of different Insurance companies and processes. It is equally essential that the new network could be built scalable to easily adapt to an always broader audience. This was far beyond the scope of traditional workflow automation technology which provides process automation for workflows with tens of tasks and one or two separate operational participants.

Furthermore, connecting and coordinating multiple instances of workflow automation presents several challenges, including much longer development time, far less efficient code, limited permissions for different users, limited visibility, and maintenance overhead. This was further far beyond the scope of Robotic Process Automation which is effective when processes only have a few steps and involve very few participants. This is why AXA and AIA have partnered with Luther Systems to utilize its Deep Process Automation technology to build this product for a safe and privacy-wise collaboration on Luther's platform.

Deep Process Automation is a new automation technology for automating complex enterprise processes with multiple separate participants involving hundreds to thousands of tasks and logical rules. Luther's platform provided the operating system to run the process while providing the rails for orchestrating, executing, and monitoring the complex workflows. It also gave the development tools to achieve rapid development times. With this new platform, similar claims detection across all the operating entities of each Insurance company can collaboratively detect and be alerted of all claims, even international cases, directly inside the product.

5.2. Demo

Here please find a Demo of the product

5.3. Resemblance Network overview

To describe the required functionalities of the product, Luther worked with AXA and AIA, subject matter experts, to develop detailed process maps and identify all the participants involved and the particular fields which can be key in the detection of matches between claims.

5.4 How does it work

Luther uses its similarity detection module to create a network of semi-trusted and untrusted participants. Each participant runs software, known as Oracles, that translates their private claim data into an encrypted form which the oracle places on the permissioned Blockchain network. The original data will never be shared in the untrusted participants' network. The Blockchain network safely and reliably executes a smart contract that examines this encrypted data to inform the participant of whether or not the original claim is a duplicate. The system compares the encrypted claim with all of the other encrypted claims saved in the network.

The encryption process uses Luther's proprietary multi-signature hashing method, which constructs several encrypted signatures from a claim such that:

- the original claim is kept private/confidential
- slight modifications of a claim (e.g., changing a name or amount) are detected.

When two claims are similar between two different insurance companies it can be a specific sign of fraud. Typically, fraud detection algorithms are measured via sensitivity & specificity (False Positive & True Negative).

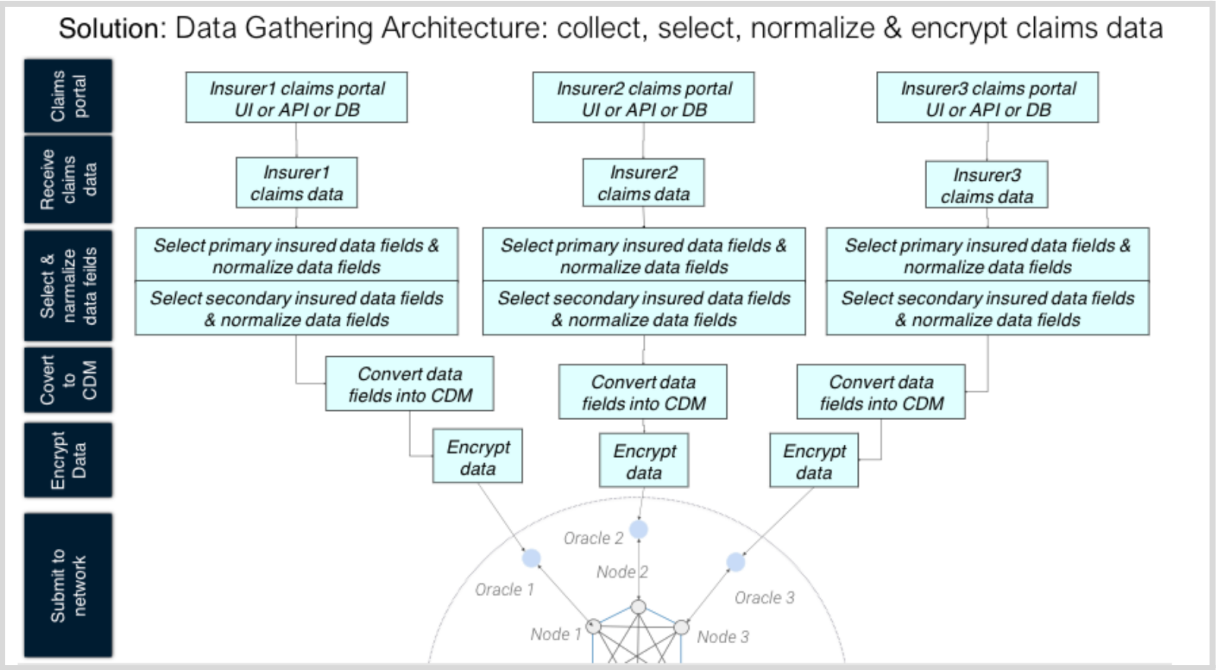
In this case, sensitivity & specificity do not apply in the usual sense, rather

- False Negative: if two claims DO match in at least 8 of 10 fields and the system DOES NOT capture that as fraud,

- False Positive: if two claims DO NOT match in at least 8 out of 10 fields and the system DOES designate this as fraud.

Based on these assumptions, the process is designed to:

- Normalize data (elastic Search Matching)
- Convert and Standardize data
- Compute multi-signature Hashes over data permutations
- Encrypt with key derivation functions
- The data are then submitted to the Network
- Rapid search with match thresholds on Common Repository
- Real-time Alerts



The data from each insurance company needs to be normalized and the key fields need to be extracted from each claim. The list of key fields is the following:

- Passport number
- Complete name of the patient
- Complete name of the doctor
- Name of the Hospital
- Amount of the services submitted for the claim
- Dates - consultation, admission, and discharge.

Once the data are extracted for each claim, they are selected and normalized.

The Luther solution uses a normalization process to convert fields from the document into a structured standard (canonical) representation. For example, a medical claim that has a hospital name field is normalized by finding the closest hospital name (Levenshtein distance) in a list of known official hospital names. To perform this matching, we use the Elasticsearch database system which executes

flexible search queries (including fuzzy matching) against a large document corpus. The solution also supports standard normalization techniques, including converting timestamps into a standard timezone (UTC), and money values into a standard format (to pence). With all the data properly organized, it is time for the computation of the multi-signature Hashes over data permutations. The solution extracts sets of the normalized fields from the document and computes cryptographic hashes ("multiple signatures") over these sets. Until this point, the data are stored in each insurance company's proprietary data space.

The hashes are subsequently submitted to the blockchain network via the Oracles. The number of hashes that are generated is a function of the business rules of the application and the required similarity metric. For example, a medical claim that includes a passport number will have an exact match requirement of the passport number, claims with different passport numbers are considered as not similar regardless of the values of the other fields. In this case, the passport field is always included in every generated signature. In cases where the field can be different (after normalization) yet the documents are still considered similar, then the system generates all subsets that both contain and do not contain that field value.

To prevent attackers from successfully decoding hashed records using brute force techniques (known as "preimage" attacks), Luther's solution leverages computationally difficult hash functions known as "Key Derivation Functions" that are tuned to require a large number of resources to compute. Specifically, the solution uses the S-Crypt hash function and sets the parameters to be as strong as possible while still meeting the performance requirements of the system. Furthermore, no information about the original Insurance company is saved in the untrusted participants' network. It takes an attacker an average of 5.7e9 years to "decode" a single claim given the solution's data fields (there are about 1e34 combinations). If a similar claim is detected, the network will send a real-time alert to the insurance company that issued the claim.

5.5. Luther platform architecture benefits

The choice of a blockchain-based architecture sets the platform on a strong footing for the future:

- Enables a single, secure, scalable shared claims database
- Provides real-time event-based architecture with multi-organization support
- Reduce the cost of processing and identifying similar claims
- Enable different organizations to work together to tackle growing fraud
- Enforces strong integrity protection of each Insurers data in the network
- A federated architecture allows easy integration with existing infrastructure
- New participants can be added over time while preserving process and data integrity, fast scalability, and data privacy.

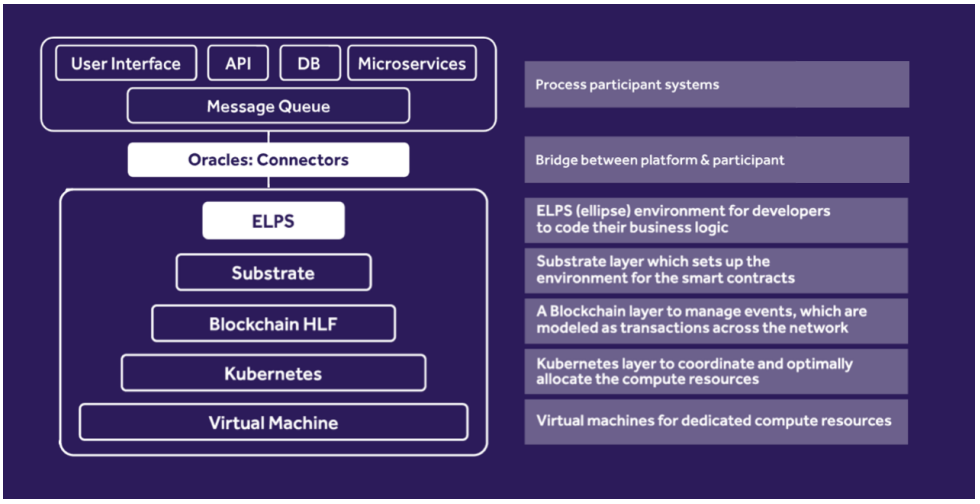
6. Technology Insights

The network developed with AXA and AIA is a collection of Virtual machines, which run Kubernetes on AWS Cloud. The network uses Hyperledger Fabric to establish a distributed ledger, which manages transactions. Luther's platform runs the business logic held within smart contracts which orchestrate and execute the detection of similarities in claims.

As underlined above, Luther uses its similarity detection module to create a network of semi-trusted and untrusted participants. Each participant runs software known as Oracles that translates their private claim data into an encrypted form which the oracle places on the permissioned Blockchain network. In this chapter we will go through the technical

6.1. Luther stack

Below is a closer look at the technical components that make up this Hyperledger Fabric network starting with an overview of the software stack.



The new product consists of a layered design where modules between layers communicate using APIs. It is built on the Luther Enterprise Infrastructure Architecture (LEIA). LEIA is a layered-microservice architecture that is built from the ground up on blockchain infrastructure. Multiple nodes owned by separate companies connect directly using permissioned blockchain protocols to form a decentralized blockchain network. In this way, processes spanning multiple companies interconnected using blockchain as the underlying orchestration and data sharing infrastructure. Please refer to Luther's offerings and website for more information on the LEIA platform.

6.2. Platform & technical specifications

The product is built on modern architecture, combining technologies including enterprise Blockchain Technology and microservices. Luther's platform provides AXA and AIA with:

- environment setup, infrastructure, platform & integrations with existing systems
- operating system (script) to orchestrate and execute automated operations

For the product, 6 blockchain nodes were deployed along with several microservices. Each insurance company has their own URL for the application.

6.3. Kubernetes to manage compute resources

Kubernetes, used here, is the standard cloud-native container orchestration platform. The network runs on the managed Kubernetes offering by AWS, Elastic Kubernetes Service (EKS) driving the following benefits:

- Resilient and scalable container (docker) execution
- Seamless integration with AWS services including EC2, API gateway, and EBS

- Low maintenance effort to stay up to date with the latest Kubernetes releases

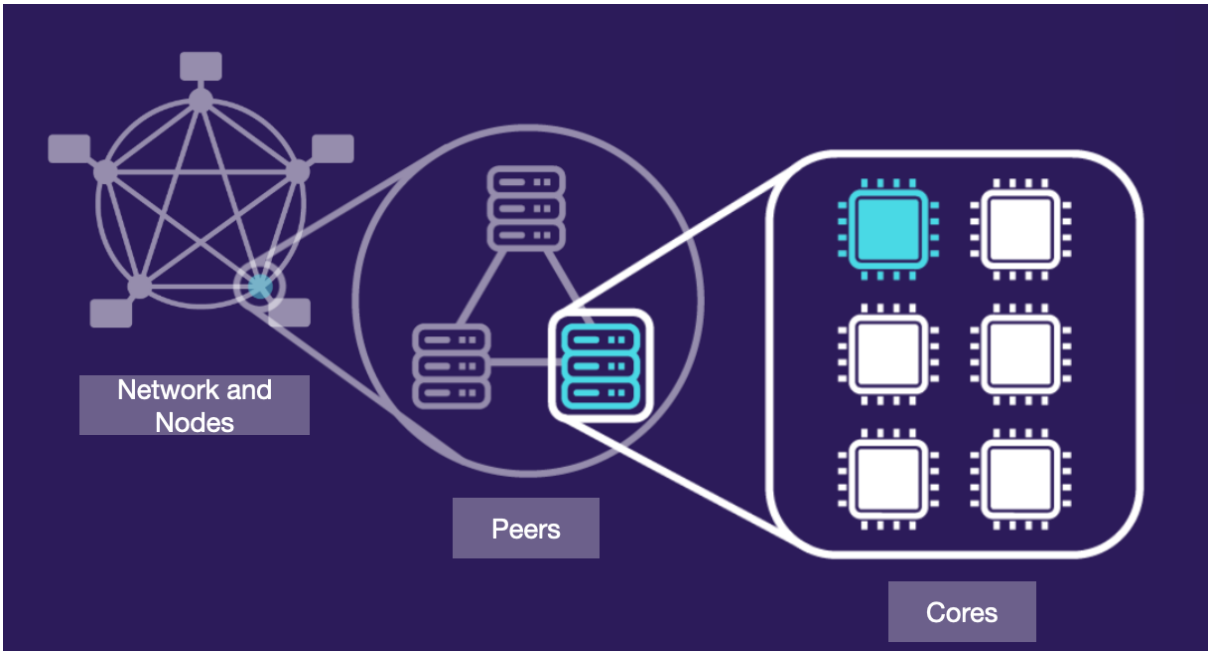
6.4. Blockchain to manage process execution events (transactions)

Each organization runs peer nodes and off-chain services known as oracles. In the UAT, claims were submitted by participants only to the oracles belonging to the submitting participant. These oracles processed the claims and then used a blockchain client to connect to their organization's peer and submit the multi-signature hashes across the blockchain network for matching by the smart contract. In the UAT, Luther ran infrastructure on behalf of AIA and AXA, where each org had:

- 2 replicas of oracles - AWS instances
- 2 replicas of BC peers - AWS instances
- 3 cores for each peer.

For improved operations, Luther also ran 2 monitoring instances, a grafana health dashboard, a VPC bastion, and an ordering service.

- Organization:
  - Across the network, each Insurance company is a fabric organization
  - There are a total of 2 fabric organizations, AIA and AXA.
- Peers
  - Each organization has at least 2 fabric peers, based on the traffic run the number of peers can be increased
  - There are a total of 4 peers.



The Blockchain network has 2 parts:

- Network operations

    - Creating, sharing, validating events as transactions across the distributed network, and storing the events (transactions) on the distributed ledger
    - The network uses private data collections (PDCs) to ensure only participants relevant to a transaction can see the associated data
    - An ordering organization that uses Raft consensus
    - The Distributed Ledger Technology (DLT) provides a standardized process execution & data sharing layer across the network
- Smart contract
    - Standardized orchestration, execution, and validation of the process steps.

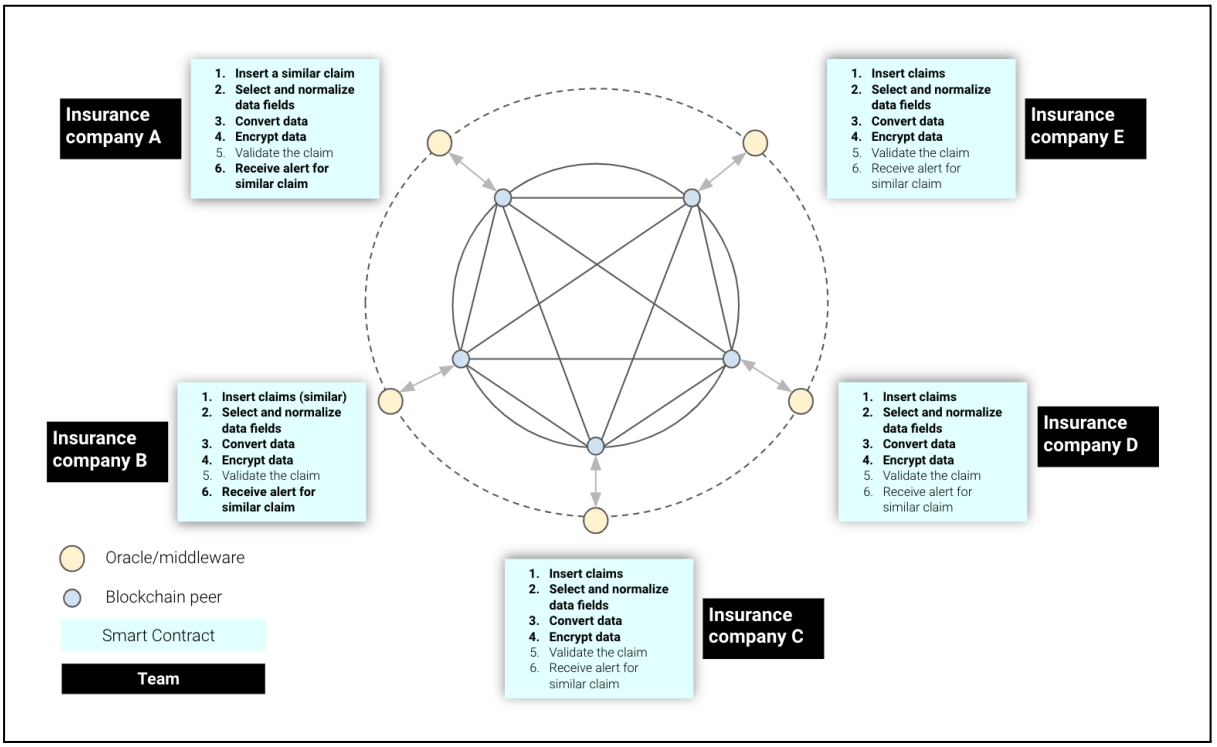
6.5. Smart Contracts to orchestrate and execute process steps as events

In the solution, process steps are referred to as events and every event is executed and stored as a transaction on the blockchain network.

The smart contract is the script that:

- At this point, the Smart contract will validate the claim by searching for a similar claim
- And send an alert to the insurance company that issued the last claim involved in the detection of similar claims

The image below, by way of illustration, is a representation of the network and the interactions. Insurance company A inserts a claim. The claim inserted by company A is similar to one of the claims handled by company B. In the case of a match, all of the companies with a similar claim receive an alert.



6.6. Oracles to manage interactions with external participants

The Oracles use clients (SDKs) to communicate with external systems and achieve a set of functions such as:

- Retrieve data from SOAP/XML or other services
- Trigger alerts
- Connect to a local identity management system to authenticate users

Finally, the oracles include a DLT client (SDK) to initiate transactions that read and write data from the ledger. This allows:

- Ingestion of data from external systems into the blockchain network for subsequent smart contract processing
- Response to events triggered by the blockchain network through smart contract processing

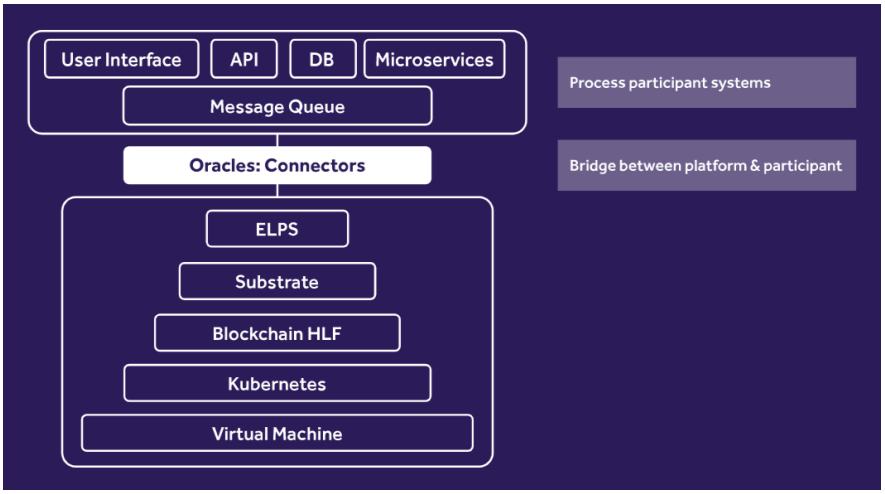
6.7. Microservice Architecture

The application is broken down into individual components based on function. These components are then packaged into containers and provide microservices with REST APIs. Each microservice API:

- Is defined using OpenAPI specification.
- Is deployed and managed via extensive automation and infrastructure as code
- Has a fully automated CI/CD pipeline for automatic deployment to the integration environment

The entire ensemble of microservices is deployed using pipelines for fully versioned deployments to staging and production. Search and Reporting Microservices - CQRS (Command Query Responsibility Segregation) - have also been developed to perform three broad functions:

- Real-time even streaming to off-chain databases for search services and reporting
- Repayable event streaming with push and pull flows to ensure reliable in-order delivery of every event to downstream consumers
    - Push flow uses RabbitMQ (Queue) message broker for immediate event processing
    - Pull flow uses DLT events for reliable event processing and failover
- On-chain event generation per transaction, with off-chain consumer microservices for an off-chain population of relational databases enabling fast off-chain indexing for search and reporting.



7. Discussions on Technical Architecture

7.1. Why not use event streaming technologies like Kafka?

Event-streaming technologies including Kafka do not provide key features necessary to meet the application requirements.

- Single org only:
    - Kafka uses a conventional consensus (zookeeper) and file system technology
    - Although these technologies are distributed for performance and availability purposes, they are centralized in that they are operated by a single administrative team
    - Kafka does not provide an architecture where multiple independent entities participate with their infrastructure (i.e., it is not federated)
    - Additional services are necessary to reconcile data across multiple organizations e.g., Hyperledger Fabric with Kafka consensus
- No logic execution capability:
    - Kafka is ultimately a publish/subscribe system
    - It reliably and efficiently delivers unprocessed messages from a single publisher to one or more subscribers
    - Any business logic or data transformation code is executed outside of the Kafka cluster and done directly within an application
- No data compliance capability to control data placement:

    - Global applications have data residency requirements where data must not leave certain organizations
    - Kafka messages are stored in topics. Topics are divided into partitions where each node (broker) hosts partitions
    - No out-of-the-box mechanism to restrict messages on the topic to specific regions or otherwise restrict them to specific nodes while maintaining message order.

Since these key features are missing, it requires developers to build additional ad-hoc services on top of Kafka. Leading to the following conclusions:

- This is bespoke, expensive, error-prone, and time-consuming
- The bespoke architecture developed will have considerable overlap with Blockchain architecture
- Blockchain technology (hyperledger fabric) is compatible with Kafka (fabric orderer)

7.2. Why HyperLedger Fabric over Corda

Corda uses an Unspent Transaction Output (UTXO) architecture which is suited to high transaction volume, low transaction logic complexity, and participants enforce validation rules on a per-transaction basis (corDapp).

On the other hand, HyperLedger Fabric (HLF) uses an Execute Order Validate (EOV) architecture which is well suited for the general execution of standardized business logic across a network. It is good for medium transaction volume with high transaction logic complexity where the Fabric network enforces standard validation rules across all transactions via smart contracts. Luther, therefore, selected HLF for its architectural benefits that are well suited for automating complex processes using smart contracts.

7.3. Scalability

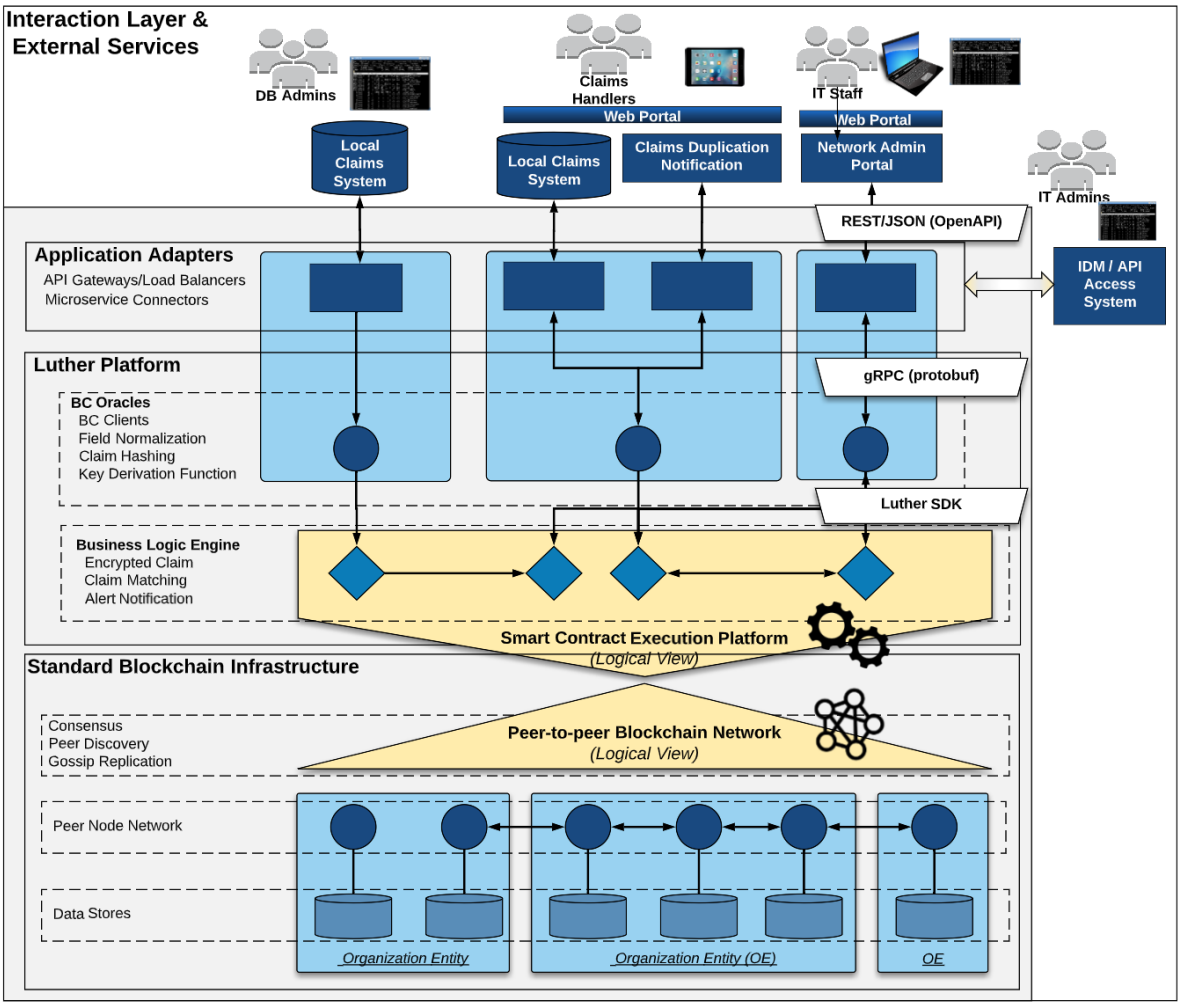
The Luther platform and infrastructure is managed entirely using Infrastructure as Code (IaC), for the rapid scaling of resources. It provides full container orchestration via Kubernetes, allowing automated and low-friction scaling of computing and storage resources across nearly 400 microservices. Moreover, gossip protocols are tuned for unique network configurations to readily support several peers.

As a result, the solution achieves real-time processing of transactions within <1s within a production scale network.

7.4. Best technology fit for the use case

Luther has considerable experience building and operating large-scale DLT networks in production.

Luther's platform was a perfect fit for the distributed nature of the process: a distributed problem that requires a distributed solution. HLF as an underlying DLT was the right technology for standardizing the complex process across all the insurance companies and enforcing strong standards.



8. Technical key performance indicators (KPIs)

8.1 Detection / Privacy KPIs

The dual goals of project Resemblance are to:

- Detect matches (possible Fraud) between submitted claims
- Ensure claims details are not visible to other platform participants or outsiders.

Luther systems compared the baseline and the developed system using several metrics and KPIs. These metrics are practical in that our customers require a solution that must provide a certain level of performance guarantees while maintaining data privacy and security.

The two KPIs for measuring the performance of the system against these goals.

- Goal 1 is a binary classification problem for which there are standard KPIs, including Specificity / Sensitivity analysis.
- Goal 2 is less clearly defined as there are no well-defined security KPIs. Below are a few options for measuring “Privacy Performance”.

8.2 Detection Performance

Typically, for Fraud detection, claims are already classified as Fraud/ Not Fraud and a detection algorithm is designed to classify new claims based on their features. In this case, once the algorithm is designed in practice, we measure its sensitivity and specificity (False Negative & False Positive) which are measures of the algorithm's performance.

8.3 Options for Privacy Performance

There are a large number of hashes on the repository at any given time. The goal of an attacker is to "decode" the hashes and learn about the details of a claim.

As measures of the algorithm to protect privacy we underlined the following:

- Expected number of "hashes" that need to be generated for a hacker to "decode" (read) any claim in the repository.
- The expected amount of time it takes a hacker to "decode" (read) any shared claim in the repository.
- Probability of one organization being able to deterministically read a claim from another organization.

These quantities are driven by the "threat model", which is defined by:

- who is the attacker
- how much data do they get access to
- how much do they know about the inner workings of the system

8.4 Options for threat model

In the options for the threat model Luther has considered:

- Other insurance companies attempt to systematically "decode" the hashes on the common repository.
- Other insurance companies attempt to "learn" about claims by submitting synthetic claims to the common repository.
- Any node in the system is externally hacked and the hackers ``decode" the hashes. The hacker knows the exact hashing function and multi-signature hashing process we use, as well as the exact data fields, the range of values that the field can assume, and the relationship between the field values.

8.5 Options for Decoding Stolen Data Methods

There are numerous methods for attempting to decode the stolen data. Below we describe a particular method for decoding the stolen data.

- Systematically select each claim field claim from a range of values
- Combine the fields to create the "combined data fields"

- Creates the multiple hashes required for the MultiSignature hashing, using Luthers's hashing function
- Compare each generated hash with all hashes on the database for an exact match
    - Any exact matches between hashes mean that the original claim data (before hashing) and the randomly generated fields are an exact match
- This process is repeated until the attacker finds an exact match

For this set up the hacker will need to create “all” combinations of field values. Given the fields we have in the product, this is approximately 1e34 combinations.

8.6 Average Number of Hashes Before the Attacker Finds a Match

Taking the “weakest Link” approach we need to find the most vulnerable hash (the smallest search space) and estimate the search space size for that hash. This is assuming that hash is the easiest to break and so an attacker will target that hash. This will be a hash of 8 of these fields together.

Keeping in mind that Passport Number is included in all Hashes, the smallest Search Space Hash is the hash of the combination of the following fields:

- Passport_number, Patient_first_name, Consultation_date, Payment, Admission_date, Discharge_date, Doctor_first_name, Hospital_name

This results in a search space of size 4.5e24, which is the number of all the hashes that can be generated. At any given point in time, there will be ~250,000 claims (hashes) on the Blockchain, which we assume the attacker has “stolen”. The attacker will generate the hashes one by one and compare each newly generated hash to the list of ~250,000 hashes she has stolen, all at once.

Given all this, it takes an attacker on average 4.5e24 / 250,000 hash generations to “find” a matching hash between her randomly generated hash and the list of 250,000 hashes stolen from the Repository. While strictly technically speaking this is an approximation, the orders of magnitude are exact.

We will conduct the following analysis for a

- “Slow Hash” which is what we use
- “Fast Hash” which is a typical hash function (for example SHA-256).
- With the Slow Hash function, each hash takes ~10ms long to compute.
- The speed at which the Hash function is computed is a parameter that we set. This parameter is set considering there are 36 hashes that need to be generated and processed all in under 1 Second, according to the requirements of the product.

- For the Slow Hash function this takes the attacker on Average 4.5e24 / 250,000 * 1e-2 seconds. This is on the order of 1.8e17 seconds. This is approximately 5.7e9 years.
- With the Fast Hash function, each hash takes ~4e-7s long to compute.
- For the Fast Hash function this takes the attacker on Average 4.5e24 / 250,000 * 4e-10 seconds. This is on the order of 7.2e11 Seconds. This is approximately 227 Years.

8.7 Detection vs. Privacy Tradeoff

On one extreme both claims' data sets are visible to a detection algorithm, which can be used to design a powerful algorithm (Sensitivity/ Specificity scores), however, this algorithm has Zero Privacy as the contents of all claims are visible to the algorithm. On the other extreme, there is the current "algorithm" (approach) of not attempting to capture Fraud (matches and points of connection), in this case, fraudulent claims go undetected, so the algorithm has very poor detection performance (Sensitivity/ Specificity scores), however, this algorithm has Perfect Privacy as the contents of one parties claims are not visible to the other party at all.

Our solution provides a tradeoff between the two extremes, the algorithm provides fraud detection while also providing a high degree of privacy.

8.8 Further Considerations:

Luther also considered the following points:

- If attackers don't have a particular person in mind, Luther's analysis is for the detection of a single claim, while it is important to protect against this, the attacker will likely want to decode multiple claims for any meaningful malicious use
- Hardware enhancements and quantum computing may not destroy this approach, using a quantum-resistant hash function. While quantum computers provide vast computational speed-ups in certain types of computations and tasks, it is possible to simply replace existing hashing functions with "quantum-resistant hash functions" which can neutralize the speed-up provided by quantum computing
- Not assuming an advanced persistent adversary (APT) who has continuous access, they only download DB once and then try to crack. The assumption in this analysis is the attacker simply downloads all the data present on the common repository once and attempts to decode the data. We do not assume the attacker has continuous access to the common repository in which case she can collect much more data and make inferences based on submission patterns. That threat model needs to be analyzed separately depending on the exact persistent threat model.

9. Results

The solution proposed in this document and the developed product address the problem with a scalable, flexible, and privacy-protected solution. Furthermore, the multi-signature hashing function, the strong hash function selected, and the distribution of data in several structures leave the information safe and not available to any malicious attacker. This chapter is for exploring the results on a business, commercial and technical level.

9.1 Product results

The solution is valuable not only for the newly introduced process but for the extensive automation that allows the process to complete a check of matches in less than a second.

Examples of automated steps are:

- Standardized common format claim

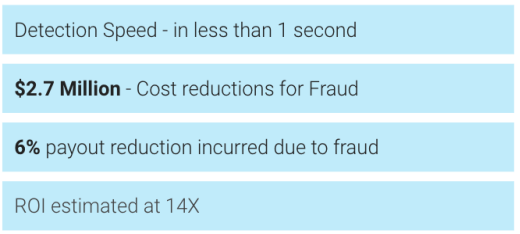
- Automated encryption of claim submission to the network
- Automated detection of similar claims across the network
- Automated alarm trigger

The system also includes a document similarity metric that detects similar documents. The system not only works on exact matches of claims but also extracts matches between fields even if one claim is not an exact duplication of the other.

The solution also keeps privacy protection as a focal point. The solution enables a safe and privacy-protected environment where collaboration has never been easier. Furthermore, it's easily scalable, which means multiple insurance companies can join the network and keep their processes and data as is without renouncing the ability to collaborate to tackle fraud.

9.2 Commercial results

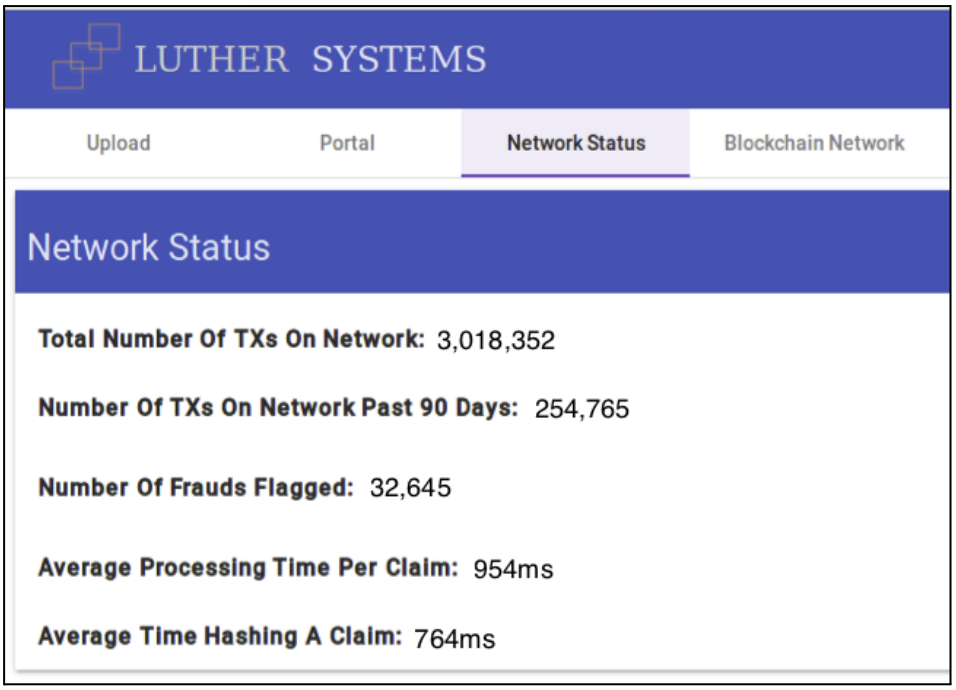
We have rapidly developed a world-leading matches detection product to tackle fraud. Thanks to this transformative initiative, similarities in claims are detected before and resolved faster and cheaper with a sharp increase in customer satisfaction.



The detection of similarities between two claims requires less than 1 second. The detection triggers a flag to the entities of a single insurance company that will require further investigation.

The flag is estimated to deliver $2.7 Million in cost optimization for insurers and a 6% payout reduction incurred due to fraud.

9.3 Technical results



9.3.1 Detection Speed: detect document similarity in less than 1 second

The solution leverages Luther's scalable and efficient platform to detect similar claims in less than 1 second. This performance enables the system to be part of existing business process workflows that must process documents in real-time.

For example, it allows an insurance claim to undergo fraud detection immediately at the time of claim submission, providing immediate feedback to the insurer and preventing subsequent processing errors and payout.

9.3.2 Document Active Period: keep docs active for a long period of time

The joint repository is capable of storing & comparing active documents for long periods, in its current version the documents were kept active for 90 days.

9.3.3 Number of active documents in the joint repository: over 1 billion docs active

The joint repository can store and compare over 1 billion encrypted active documents. This enables the solution to detect similar documents across multiple organizations in near real-time dating back long periods.

9.3.4 False positive rate: 0.1% false positive detection rate

Luther's solution achieves great accuracy with a 0.1% false-positive detection rate based on constructing signatures for each document.

9.3.5 Security & cost to decrypt a document on the repository

Luther's solution relies on signatures that are resistant to brute force attacks.

It takes an attacker over 6 months and over $10M to decrypt one encrypted doc.

We specifically tune the hashing function to be as close to impossible to break as possible, while maintaining the average detection delay requirement. Our estimates indicate that even if an external party gets access to the repository by breaking the security layers in accessing the repository, it will take a highly specialized software dedicated to the decryption task, over 6 months, and over $10m in computing cost to decrypt the document and gain access to the data.

10. Expansion

Think of multiple similar processes solved by the same product. We need to make fragmentation across complex processes a thing of the past. It is like building a railroad network where we have just positioned the first rail track. Several industrial processes can benefit from the introduction of a product like Resemblance. It can securely share the information in a network where silos visibility is creating economic impacts on business.

Hereafter are some examples of how Resemblance can solve other matches and points of connection issues:

Coordination of benefits: If you are health insurance and one of your plan members issues a claim, you need to know if the plan member is registered as a secondary beneficiary of another insurer's plan. Resemblance can flag the possible presence in both policies of the same plan member, without sharing private information of either the plan member or the Insurance company.

Open lines of credits for customers. If you are a construction company or a supplier and you need to keep track of the customers' credit lines and the number of entities and offices makes the check a little bit difficult. Resemblance can instantly recognize if there is already a line of credit for a specific customer and settle all under a specific threshold.

Inventories and customer management. When you have several business lines and several customers and several entities, it is difficult to check and track all of them. The actual result is that you can have 30000 customers when you have 2000. Resemblance can instantly reconcile your data and flag the presence of a similar customer or product in your system. This will help you tailor your offer to the customer.

Billing liabilities. When you must check, as a healthcare provider, billing liability based on Personal and private information, you have several offices, and the transmission of data can lead to privacy, hence the reconciliation is difficult. Resemblance can solve the problem by adding the right level of shared information always security and privacy-wise.

Patent and trademark offices. There are always more products that, under regulations, cannot be copied. If you are a music provider or a pharmaceutical company, how do you track that your patent and trademark are protected? Resemblance can do this for you, check several databases and extract and flag any unauthorized reproductions of your product, without the need to expose your private information.

Rapid expansion company - **target evaluation**. When analyzing a merger or acquisition it is difficult to understand the synergies and the costs associated. Resemblance can underline similarities and differences between the two companies and enhance the process of evaluation of the companies, keeping the information secure.

Striving to make your life better and safer

11. AXA Company & Offerings

AXA S.A. (styled as AXA) is a French multinational insurance company and one of the leading insurance companies around the globe. The AXA Group operates primarily in Western Europe, North America, the India Pacific region, and the Middle East, with a presence also in Africa. AXA is a conglomerate of independently run businesses, operated according to the laws and regulations of many countries. It is a component of the Euro Stoxx 50 stock market index[3].

AXA offers a variety of products and services to protect:

- Properties, which include the insurance of personal property (cars, homes) and liability (personal or professional)
- People who encompass both savings and retirement products, on the one hand, and other health and personal protection products
- Asset involves investing and managing assets for the Group's insurance companies, as well as for third parties, both retail and institutional clients.

AXA has **149,000 employees** committed to better protecting customers around the world, **50 countries** where they deliver the same quality of service and dedication around the world, and **95 million customers**. It has revenues of **Euro 9.9 billion** and a Net income of **Euro 7.3billion**[4].

[3] https://en.wikipedia.org/wiki/AXA

[4] https://www.axa.com/en/about-us/key-figures#tab=social-data

For a century, we have served millions of people and generations of families all around Asia.

12. AIA Company & Offerings

AIA Group Limited, known as AIA, is an American-founded Hong Kong multinational insurance and finance corporation.

It is the largest publicly listed life insurance and securities group in Asia-Pacific. It offers insurance and financial services, writing life insurance for individuals and businesses, as well as accident and health insurance, and offers retirement planning, and wealth management services, variable contracts, investments, and securities.[5]

AIA is the largest independent publicly listed pan-Asian life insurance group – with a presence in **18 markets across Asia**. AIA is committed to helping you meet your financial needs and goals with a wide range of life and health protection and long-term savings products for individuals and businesses.[6]

AIA achieved strong financial results for this year with a value of new business (VONB) of **US$3,366 million**, underlying free surplus generation (UFSG) of **US$6,451 million**, and an embedded value (EV) operating profit of **US$7,896 million**.[7]

[5] https://en.wikipedia.org/wiki/AIA_Group

[6] https://www.aia.com/en/about-aia.html

[7] https://www.aia.com/en/media-centre/press-releases/2022/aia-group-press-release-20220311.html

Accelerating the advent of the automated enterprise

13. Luther Company & Offerings

13.1. Who we are

Luther Systems is a software company and a pioneer in Deep Process Automation: the business of automating, orchestrating, and managing complex enterprise processes.

At Luther, we build the next generation of enterprise computation technology for use by organizations with processes that have remained out of reach for prior automation platforms.

Through our platform, we enable organizations to reimagine the way they operate and unlock unparalleled levels of automation in a world where collaboration and flexibility across disparate organizations, geographies, regulations, or standards are more important than ever.

13.2. Luther's platform for automation

At Luther, we recognize that enterprise processes of today are complex and challenging to automate. They require orchestration across multiple participants, hundreds to thousands of tasks as well as non-standard systems and datasets. Their execution is filled with reconciliation, rework, delays, and costs that have been unavoidable until now.

Luther's unique proposition lies in its ability to take on this complexity through a distributed technology architecture: a distributed solution for a distributed problem.

With our proprietary LEIA platform, we provide our customers with:

- Enterprise developers' tools to automate their applications rapidly
- The operating system orchestrates and executes their automated processes

2.5X faster application development

10X Total Cost of Ownership reduction

7X process execution

10X ROI

Fully automated compliance by design

Highly scalable

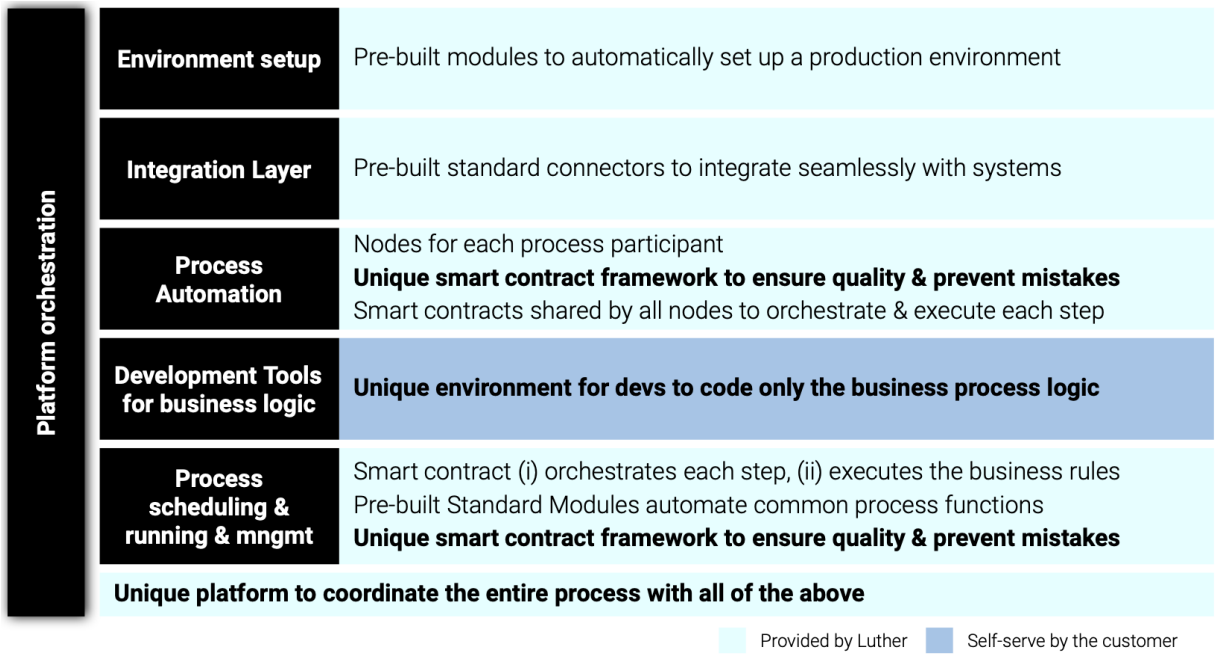
Improved customer experience

Reports from the field have been staggering, validating our vision for the future of enterprise computing. Our customers span multiple industries and use our platform today to orchestrate complex processes such as Claims Settlement, Mortgage Sourcing, Asset Issuance, or Customer 360 Views. Their execution cuts across siloed functions, teams, or even organizations performing thousands of independent steps across UIs, APIs, databases, applications, workflows, and Robotic Process Automation (RPA).

13.3. Luther's offerings

Luther's unique architecture combines and coordinates multiple layers of technology which enables enterprises to (i) develop enterprise-grade automated processes and (ii) orchestrate & execute the automated processes in production.

Below is an overview of Luther's stack. Luther's LEIA platform automates and provides the majority of this stack so that enterprise developers can exclusively focus on developing their business process logic.



The Luther Platform is built around Luther's breakthrough insight that virtually all complex processes can be seen as a set of "smart contracts" between steps or participants in a process. Smart contracts are the rails over which the Luther Platform orchestrates, executes, and monitors processes in real-time. With the LEIA platform, our customers can ensure that multiple steps across the entire process are executed and orchestrated in a way that follows a predefined & agreed-upon business logic. This enables the Luther Platform to easily automate complex processes that were previously highly manual and non-standardized.

Luther's distributed platform also provides developers with the tools to achieve rapid development times and keep them in total control of the automation process. The LEIA Platform is designed to make the complex simple and can be used by developers with only a few weeks of training.

Luther's platform can be applied to numerous complex enterprise processes across industries.

For more information about Luther's platform please visit our website.