Big Data + Blockchain in HealthIT

Building a solution with Bluemix's Blockchain and Big Data

Introduction

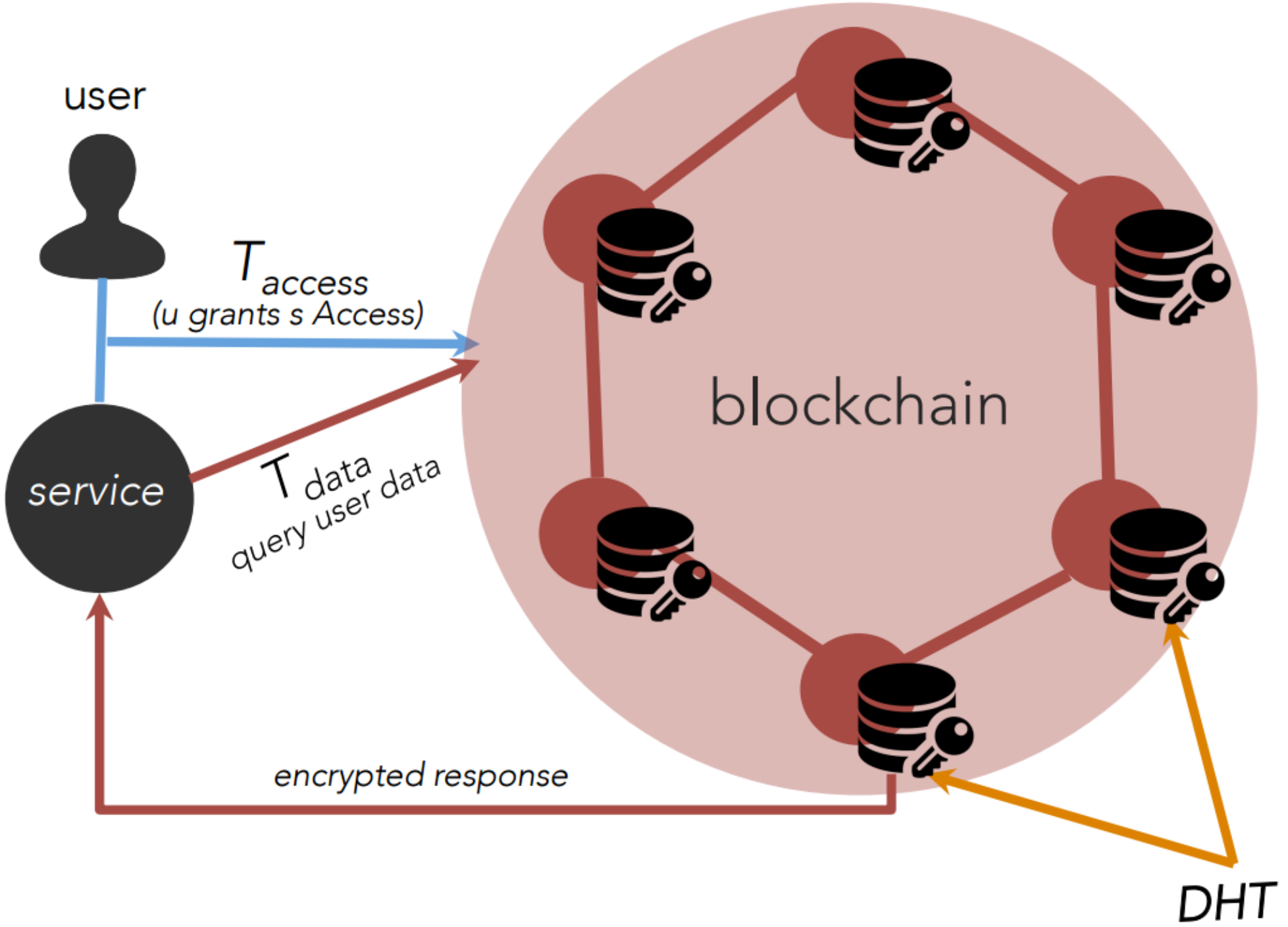
- This project started during the Blockchain in Healthcare Code-A-Thon in Washington DC for the 2017 Blockchain in Healthcare Summit.
- Overall, the goal was to develop a solution that enhances interoperability and demonstrates the potential to seamlessly incorporate blockchain technology into existing HealthIT systems.
- Specifically, I took the Metadata Tagging and Policy Expression track, to demonstrate the use of of blockchain for security metadata and tagging to manage access, provide auditing and provenance information.
- I took my lead from the Blockchain For Health Data and Its Potential Use in Health IT and Health Care Related Research paper,

  > Our proposal involves the use of a public blockchain as an access-control manager to health records that are stored off blockchain.
  >
  > ….
  >
  > There are currently no open standards or implementations of blockchain that utilize this approach but research supports the feasibility of the proposed solution.

- My goal was to provide an implementation that could be used by corporate IT departments in the Health space with an eye to Innovation Risk Management.
  - Don’t change more than you absolutely must
  - New technologies fail in the enterprise because of Governance standards issues rather than technical issues
  - Don’t let the perfect be the enemy of the good

Current Architecture

- According to the Blockchain for HealthcareIT paper, any blockchain for health care would
  - Need to be public
  - need to include technological solutions for three key elements:
    - Scalability
    - Access security
    - Data privacy
- According to the MIT paper on decentralizing privacy, the off-blockchain storage needed for scalability should be on a similar data store following the Kademlia specification for a distributed hash table (DHT).
- The overall architecture would look similar to the diagram on the following page with services as a proxy for corporate health IT departments and a public blockchain and, presumably, DHT.


user
T_{access}
(u grants s Access)
service
T_{data}
query user data
blockchain
encrypted response
DHT

Perspective

- Private blockchains are technically far inferior to public blockchain implementation. If it was only a technical discussion, we would be able to move on. But technical will always take a backseat to the perception of regulatory requirement. Best to take a page from Supply Chain Management and EDI and make the specification, not the implementation, public.
  - Prefer publicly available standards to publicly available systems.
- Try not to introduce a new technology to an IT departments with a new programming language. R3 spent $53M to produce a framework in Kotlin. Solidity, Ethereum?
- Regulated IT departments distrust external data the way mice distrust tigers. They will launch a POC, but it’ll stay a science project.

Proposal

- We’ve never had a data revolution and we’re not getting one now.
  - When we needed more flexible data models, we added relational database in addition to mainframes.
  - When we needed to aggregate enterprise data, we added data warehouses.
  - When we needed to store internet-scale data, we added Hadoop clusters.
  - When we needed to react to device-scale data in real-time, we added Spark.
  - Now we need to manage trust externally at scale.
- How do we make this merely additive?

TaLIa

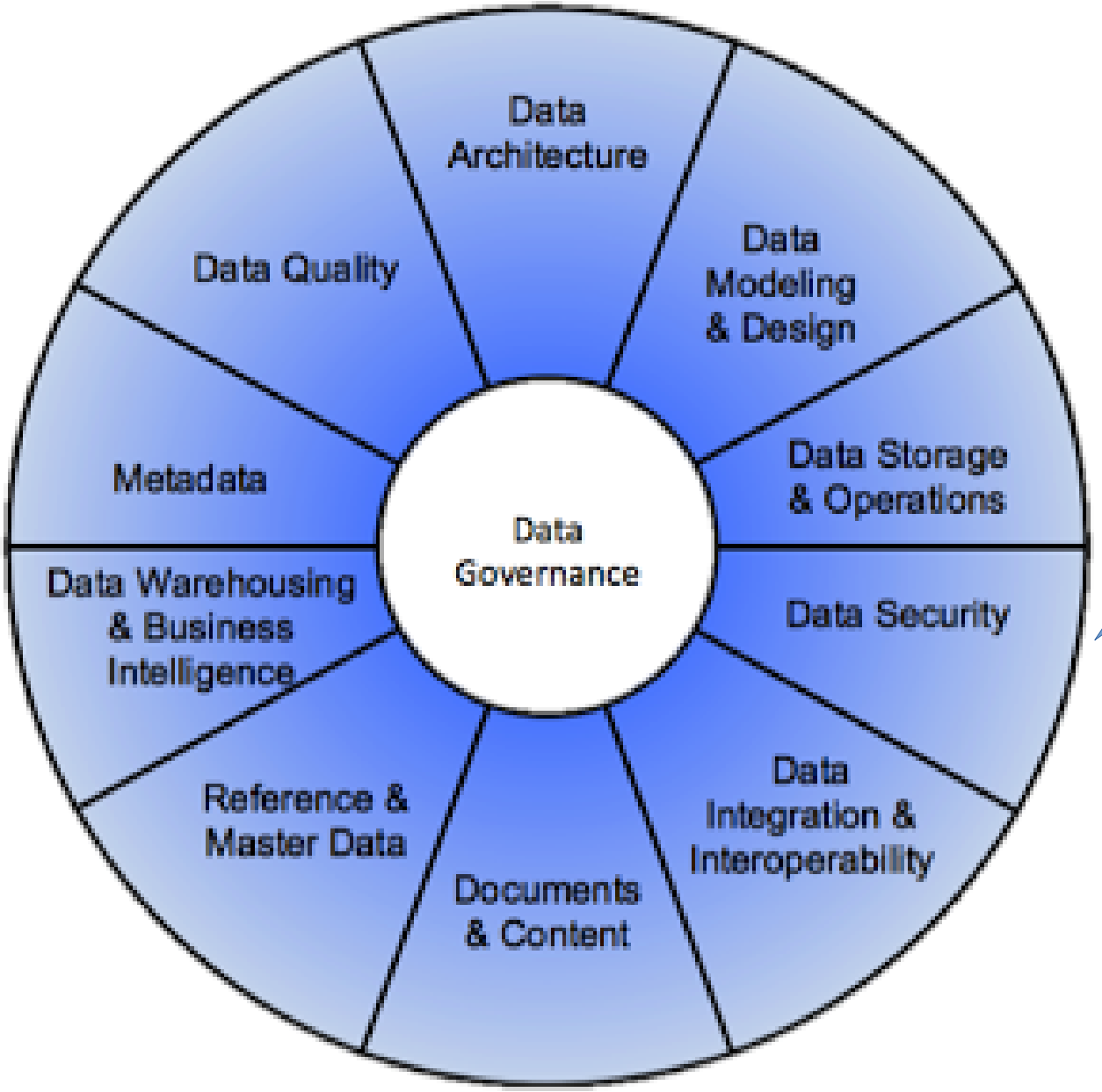
Trust **L**ayer **I**njection

The voice of the patient
emboldened by blockchain

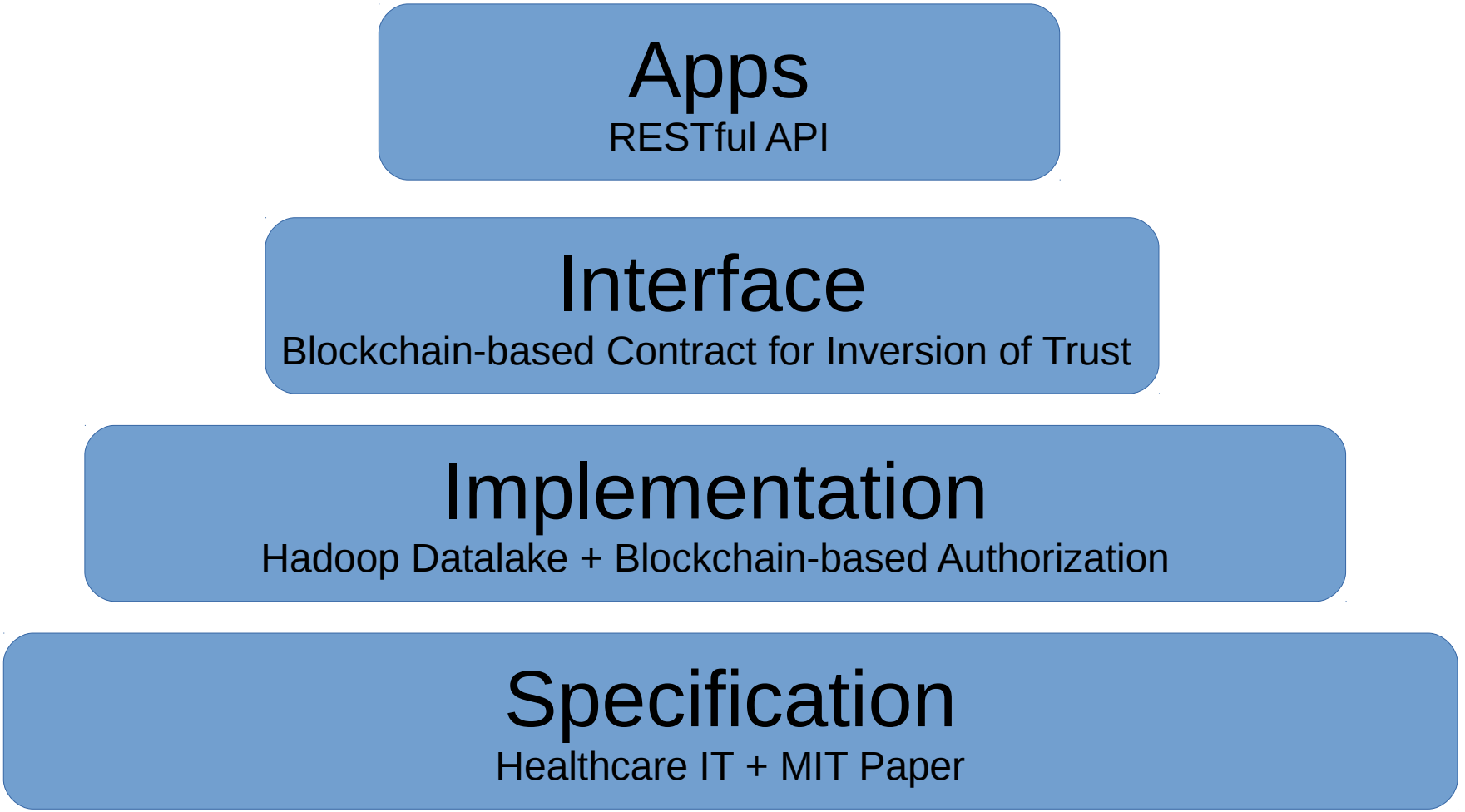
Blockchain Model for Healthcare

- I am going to suggest a blueprint that hopefully follows the path of least resistance through your enterprise data governance stage gates.
- I am going to make a few datacenter assumptions:
  - There is an onsite Hadoop data lake
  - The data lake is secured
    - Authentication – Apache Knox
    - Authorization – Apache Ranger
    - Audit – Apache Ranger
    - Data Protection – Apache Ranger KMS
  - You have the tools in place for a standard Lambda architecture (Hbase for speed, HDFS for batch, Hive/Impala for reporting). Hbase 0.98+ is really the only mandatory component, though.
  - There is an ability to consume and serve public RESTful APIs, but they have to go through a some sort of integration layer
- We will be building the chaincode using IBM's Hyperledger Fabric, so hopefully your organization
  - allows Apache-licensed open-source apps
  - primarily codes in JVM languages (Java, Scala)
  - having an open mind to Google's Go is a really, really nice to have.

Innovation Risk Management



High Level Architecure



Application Layer

- At the API level, the MIT Enigma project provides the guidance we need to build our APIs.
  - Note that in all of these examples, it is assumed that the user has previously created a profile in the company's existing identity and access management system.
- A user installs an application to access their EHR from the provider's system.
- When the user signs up, a new compound identity (user, company) is created and sent, with predefined permissions, to the blockchain.
- Data sent from the user to the system is encrypted using this compound identity.
- Permissions can be set at any time by the user, limited by certain corporate rules (no one is allowed to request that PHI be made public, for example.)
- Services that request permission to query data, for research purposes for example, also create a compound identity.
- Requests for data return some anonymized subset of data. The principal of least exposure will be adhered to based on the corporate governance strategy and the user's preference. Any exposed data must be agreed to be exposed by both the company and the user.

- Note that a compound key is created, just like the Enigma model where a public blockchain is used. While technically this would not be necessary in a private blockchain, this is meant to be consumed by another company with their own private blockchain, or even a consortium with a public blockchain. Regardless, we are implementing locally but planning to distribute globally.

Interface Layer

- Data that is sent from a device to the system is stored on Kafka.
- A Spark job will pull the data from the Kafka topic, create a key for Hbase, and then store the data in both Hbase and Blockchain.
- HBase is going to rely on cell-based tags to enforce the terms of the blockchain contract by providing role-based visibility at the datum level. This is the injection part of of trust injection.
- When a record is first created in Hbase through this system, this record will be persisted but not yet visible. Hbase is extremely fast while blockchain is extremely slow. However, the assumption will be that eventual trust will be sufficient for most use cases. This is an asynchronous system with a fairly hefty latency.
- Periodically, through some scheduler like Oozie, the blockchain will be queried to retrieve records that have been authenticated but not yet minimized. Those records will compare a have of their content against a hash of the Hbase content using the rowkey as a match. When there is a match, the cell-level visibulity will be updated in Hbase and the blockchain will just store a pointer to the Hbase record, rather than the full dataset.

Implementation

- Hbase is a sparse, distributed, persistent multidimensional sorted map, which is indexed by a row key, column key, and a timestamp. Row keys are unique. Cells store data as key value pairs and are stored in a column family.
- Every cell can have zero or more tags. Every tag has a type and the actual tag byte array. Just as row keys, column families, qualifiers and values can be encoded , tags can also be encoded as well. You can enable or disable tag encoding at the level of the column family, and it is enabled by default.
- Hbase provides two levels of cell level access control
  - Cell level ACL
    - Sets RW using Role-based Access Control (RBAC)
  - Cell Level Visibility
    - Allows labels to be associated with the data cells. Cells across rows and columns can have visibility labels. Users or groups can be granted authorization to the labels.
    - For example, certain cells can have a visibility label called ‘private’. Only users granted authorization to ‘private’ can access the labeled data.
    - Another example, patient or customer data can be labeled ‘branch1′, branch2’ etc based on where the patient or customer is located or registered. A doctor or administrator can only access the data that he or she is authorized.

Implementation

- Interpreting the labels authenticated for a given get/scan request is a pluggable algorithm. By creating an algorith, it will be possible to add metadata management to this process, using an existing corporate metadata repository, a tool such as Cloudera navigator, or even storing the metadata in a column family on the same Hbase row.
- The ID of the blockchain serves as a pointer to the Hbase record. Access to Hbase is not done through the blockchain, so its necessary to reflect blockchain rules at this level.
- Apache Ranger is used to manage role-based and tag based solutions and can also store and operate on metadata. At this time, only Hive and HDFS can have their access and auditing needs handled by Ranger. This functionality will need to be provided in order to enable this level of security out of the box.

Summary

- The tools to implement blockchain are, to some extent, already available in most Health Care enterprises.
- While a network of loosely connected private blockchains communicating though a RESTful APIs conforming to a public specification is not ideal from a technical perspective, it is more likely to be implemented into an existing IT infrastructure.
- The goal of having the patient being involved in their own EHR is preserved though the implementation of IBM’s Hyperledger Fabric using MIT’s specifications.
- Hbase can be used as the off-blockchain storage. While it is not as seamless as storing to another distributed hash tag, it does provide the ability to store and manage metadata and providence at the cell level.