Fine-grained access control with attribute based cache coherency for IoT with application to healthcare

by

Piranava Tamilselvan

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Engineering

Iowa State University

Ames, Iowa

2017

DEDICATION

Dedicated to my parents – Mr. S. Tamilselvan and Mrs. V. Kokila for all their love and support and for giving me the best always, my sister Chinmaya Tamilselvan for her unending support and love.

Dedicated to my friends and relatives, who have supported me throughout the process. I will always appreciate all they have done and I wouldn’t have gotten through this process if not for them.

Also dedicated to the memories of my Grandfather Mr. Sivasambu, who always motivated me to achieve great heights.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would first like to thank my Major Professor, Dr. Manimaran Govindarasu for his guidance, encouragement, and patience over the last two years. Thank you so much for pushing me hard to look at research in different ways and for being my constant source of knowledge and inspiration. You consistently steered me in the right direction whenever you thought I needed it.

I would also like to thank Dr. Doug Jacobson and Dr. Swamy Ponpandi for agreeing to be on my Committee and taking out time to respond to my e-mails.

In addition, I would like to thank my research group for providing me valuable feedback during all my research presentations and my friends who provided moral support and helped me in understanding certain technicalities that I was not aware of previously.

ABSTRACT

The Internet of Things (IoT) is getting popular everyday around the world. Given the endless opportunities it promises to provide, IoT is adopted by various organizations belonging to diverse domains. However, IoT's "access by anybody from anywhere" concept makes it prone to numerous security challenges. Although data security is studied at various levels of IoT architecture, breach of data security due to internal parties has not received as much attention as that caused by external parties. When an organization with people spread across multiple levels of hierarchies with multiple roles adopts IoT, it is not fair to provide uniform access of the data to everyone. Past research has extensively investigated various Access Control Techniques like Role Based Access Control (RBAC), Identity Based Access Control (IBAC), Attribute Based Access Control (ABAC) and other variations to address the above issue. While ABAC meets the needs of the growing amount of subjects and objects in an IoT Environment, when implemented as an encryption algorithm (ABE) it does not cater to the IoT RDBMS applications. Also, given the query processing over huge encrypted dataset on the Cloud and the distance between the Cloud and the end-user, Latency issues are highly prevalent in IoT applications. Various Client Side caching and Server Side caching Techniques have been proposed to meet the Latency issues in a Client-Server Environment. Client Side caching is more appropriate for an IoT Environment given the dynamic connections and the large volume of requests to the Cloud per unit time. However, an IoT Cloud has mixed critical data to every user and conventional Client Side caching Techniques do not exploit this property of IoT data.

In this work, we develop (i) an Attribute Based Access Control (ABAC) mechanism for the IoT data on the Cloud in order to provide a fine-grained access control in an organization and

(ii) an Attribute Based Cache Consistency (ABCC) Technique that tailors Cache Invalidation according to the users' attributes to cater to the Latency as well as criticality needs of different users. We implement and study these Models on a Healthcare application comprising of a million Electronic Health Record (EHR) Cloud and a variety of end-users within a hospital trying to access various fields of the EHR from their Smart devices (such as Android phones). ABAC is evaluated with and without ABCC and we shall observe that ABAC with ABCC provides a lower average Latency but a higher staleness percentage than the one without ABCC. However, the staleness percentage is negligible since we can see that much of the data that contributes to the staleness percentage are the non-critical data, thus making ABAC with ABCC an efficient approach for IoT based Cloud applications.

CHAPTER 1. INTRODUCTION

1.1 The Internet of Things

1.1.1 An Overview

The Internet of Things (IoT) [1] refers to the use of intelligently connected devices and systems to leverage data gathered by embedded sensors and actuators in machines and other physical objects. Advances in technologies especially Wireless and Mobile connectivity, Radio-Frequency Identification (RFID), smart sensors, etc., when combined, could help realize a miniaturized, embedded, automated Internet of connected devices communicating regularly and relatively effortlessly. IoT promises to change our way of doing things through better information in real-time and improves learning opportunities. IoT can improve efficiency (achieving similar levels of impact with fewer resources) and/or enhance effectiveness (increasing impact with similar levels of existing resources). In short, IoT is an ecosystem of inevitably related processes and other technologies from the perspective of a goal within a specific use case.

In many respects, it can initially look the same as M2M (Machine to Machine) communication – connecting sensors and other devices to Information and Communication Technology (ICT) systems via wired or wireless networks. In contrast to M2M, however, IoT also refers to the connection of such systems and sensors to the broader Internet, as well as the use of general Internet technologies [2]. M2M is almost synonymous with isolated systems of sensors and islands but IoT is the ecosystem that connects these disparate vertical pillars.

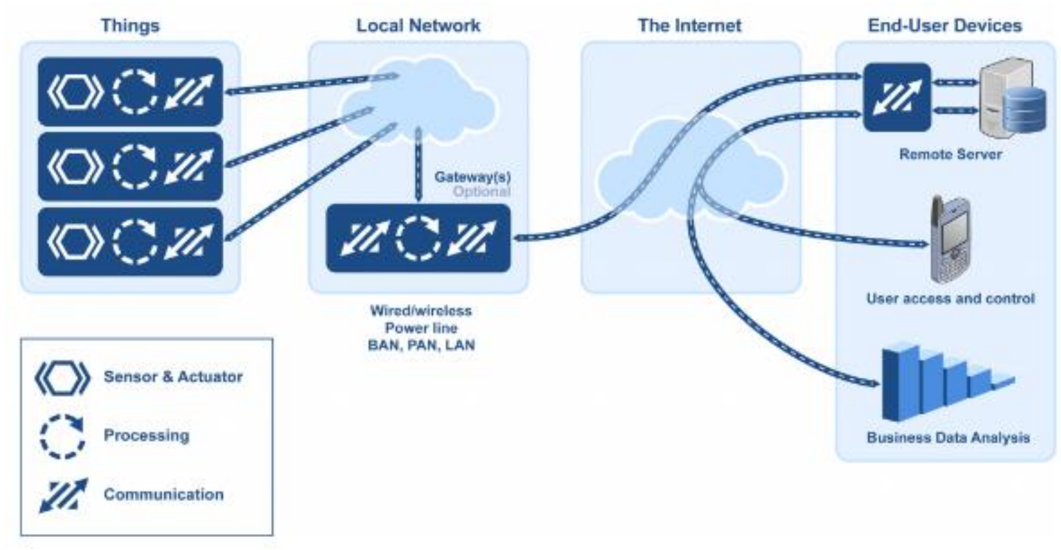


Figure 1. *The Elements of IoT*

The main components of the IoT are:

- The thing itself
- The Local Network, which moves data in and out of the device
- The Internet, from where it goes to the Cloud considering the massive amount of data collected from various devices spread across various regions. The data stored goes to the end-user devices at appropriate times
- End-user devices (desktop, laptop, smartphones) or enterprise data systems that receive and manipulate data

Looking towards the applications and services in the IoT, we see that the application opportunities are open-ended, and only imagination will set the limit of what is achievable. There are many specialized use cases of IoT. Some of the most prominent application areas are: building and home automation, medical and healthcare, transportation, manufacturing, Environmental monitoring, etc.,

1.1.2 Need for Access Control in IoT Applications

The greater the connected devices, the more the possibilities for cyber-attacks [4]. One of the most prevalent concerns of implementing the IoT for any application would be the security risks revolving around the various entities – the edge device, the network through which the transit of data occurs and the cloud. Breaching of confidential data through any of these entities is dangerous. In a recent proof-of-concept exploit, for example, researchers demonstrated that a network could be compromised through a Wi-Fi-enabled light bulb [3].

Several measures are already being taken to close the holes and prevent security breaches at the device level, and efforts are being led to tackle major disasters before they happen. A lot of encryption and other cryptography Techniques play a crucial role in safe data generation and transit. However, to ensure an end to end security, the Cloud has to be trusted [5]. The IoT and the Cloud are dependent on each other. While IoT exploits the unlimited capabilities of the Cloud in terms of storage and processing, Cloud would not be interesting if it's not for the IoT data! Nowadays, with a lot of trusted Cloud Service providers and with no security keys being stored on the Cloud, we can be sure that the encrypted data stored in the Cloud would not be exposed to any hackers trying to get illegal access to the data. However, ensuring confidentiality and integrity is much more than mere encryption preventing only the hackers from accessing the data. To ensure the CIA (Confidentiality-Integrity-Availability), access must be restricted to those authorized to view the data in question. That is, no matter if it's the people inside/outside the organization, a resource could only be made available to a subject who is entitled to accessing it. This is to make sure that users at various hierarchies in an organization get access to only the appropriate data. This appropriateness would be determined by the IoT application given its nature of data and the various types/levels of users accessing the data.

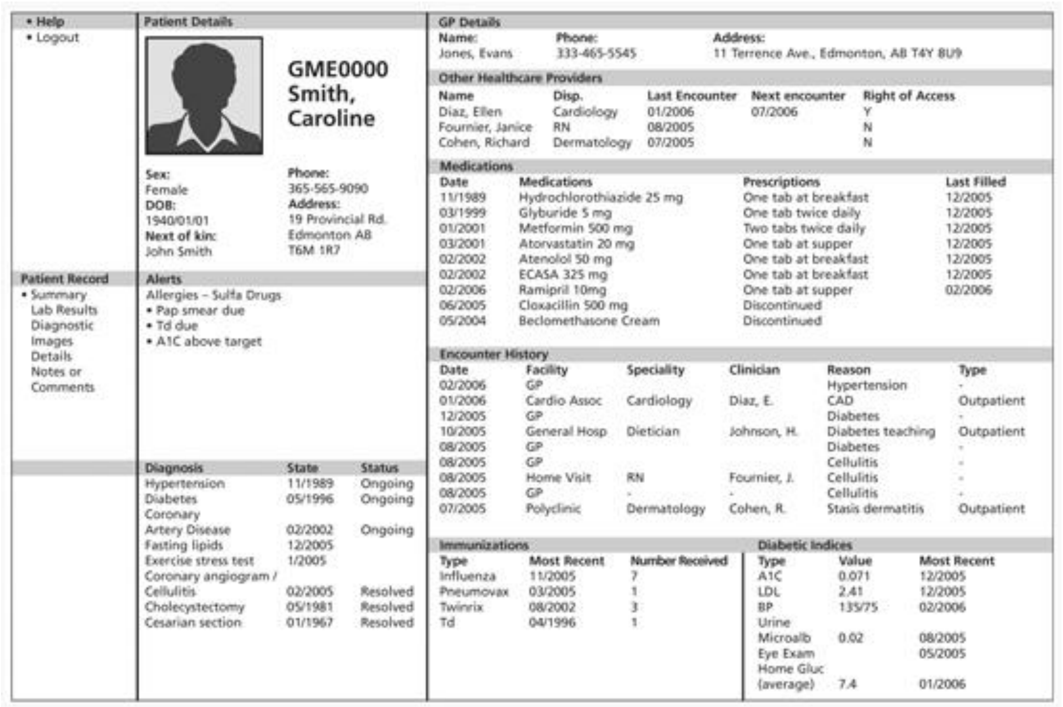
Some of the popular IoT applications include Smart Home, Smart Healthcare, Smart Cities, etc.,[6] In this work, we apply the proposed Models on a Healthcare IoT application due to the high sensitivity and criticality of the data. In the context of an IoT Healthcare application, a patient deserves the right to care about the privacy of his/her data. Various vital signs of patients like Blood Pressure, heart rate, etc., would be generated and transmitted to the cloud repository. These data along with other sensitive data in Electronic Health Records (EHR) like Social Security Number (SSN), billing information, etc., have to be kept secret [7] due to obvious security reasons. Many conventional end-to-end security mechanisms involving encryption would prevent intruders/eavesdroppers from accessing these critical data. However, the data has to remain secure internally within the organization as well. That is, which users with what privileges can have access to which kind of data must be enforced.

1.1.3 Internet of Things in Healthcare

With the advent of IoT, Smart Healthcare is now possible. Smart Healthcare devices are wirelessly enabled and can be used to monitor and collect data from patients suffering from various disorders. They put the critical data, such as CT scans, test results and other records, into the hands of patients as well as medical teams at any-time and on any smartphones or computers. The smart devices can gather data on their own and remove the limitations of human-entered data. The doctors can obtain the data that they need whenever they want, thus enabling them to have better interaction with patients remotely. This process reduces the risk of error, which means increased efficiency, reduces cost of care and increases quality of care in healthcare.

Off-line monitoring of patients has also become possible because of IoT. Healthcare monitoring and wearable devices are capable of transmitting data from a patient's home to the hospital. This type of automated process replaces the doctor/patient visiting by regular intervals to check the health of the patients.

An Electronic Health Record (EHR) is an electronic version of a patient's medical history, that is maintained by the provider over time, and may include all of the key administrative clinical data relevant to that persons care under a particular provider, including demographics, progress notes, problems, medications, vital signs, past medical history, immunizations, laboratory data and radiology reports. Given the digital nature of an EHR, information is available whenever and wherever needed. Healthcare professionals claim that with the data, timeliness and availability of EHRs, better decisions and more coordinated care are made possible.



• Help
• Logout

Patient Details

GME0000
Smith,
Caroline

Sex: Female
DOB: 1940/01/01
Next of kin: John Smith
Phone: 365-565-9090
Address: 19 Provincial Rd. Edmonton AB T6M 1R7

Patient Record
• Summary
Lab Results
Diagnostic
Images
Details
Notes or
Comments

Alerts
Allergies – Sulfa Drugs
• Pap smear due
• Td due
• A1C above target

Diagnosis	State	Status
Hypertension	11/1989	Ongoing
Diabetes	05/1996	Ongoing
Coronary Artery Disease	02/2002	Ongoing
Fasting lipids	12/2005	
Exercise stress test	1/2005	
Coronary angiogram / Cellulitis	02/2005	Resolved
Cholecystectomy	05/1981	Resolved
Cesarian section	01/1967	Resolved

GP Details

Name:	Phone:	Address:
Jones, Evans	333-465-5545	11 Terrence Ave., Edmonton, AB T4Y 8U9

Other Healthcare Providers

Name	Disp.	Last Encounter	Next encounter	Right of Access
Diaz, Ellen	Cardiology	01/2006	07/2006	Y
Fournier, Janice	RN	08/2005		N
Cohen, Richard	Dermatology	07/2005		N

Medications

Date	Medications	Prescriptions	Last Filled
11/1989	Hydrochlorothiazide 25 mg	One tab at breakfast	12/2005
03/1999	Glyburide 5 mg	One tab twice daily	12/2005
01/2001	Metformin 500 mg	Two tabs twice daily	12/2005
03/2001	Atorvastatin 20 mg	One tab at supper	12/2005
02/2002	Atenolol 50 mg	One tab at breakfast	12/2005
02/2002	ECASA 325 mg	One tab at breakfast	12/2005
02/2006	Ramipril 10mg	One tab at supper	02/2006
06/2005	Cloxacillin 500 mg	Discontinued	
05/2004	Beclomethasone Cream	Discontinued	

Encounter History

Date	Facility	Speciality	Clinician	Reason	Type
02/2006	GP			Hypertension	-
01/2006	Cardio Assoc	Cardiology	Diaz, E.	CAD	Outpatient
12/2005	GP			Diabetes	-
10/2005	General Hosp	Dietician	Johnson, H.	Diabetes teaching	Outpatient
08/2005	GP			Diabetes	-
08/2005	GP			Cellulitis	-
08/2005	Home Visit	RN	Fournier, J.	Cellulitis	-
08/2005	GP	-	-	Cellulitis	-
07/2005	Polyclinic	Dermatology	Cohen, R.	Stasis dermatitis	Outpatient

Immunizations

Type	Most Recent	Number Received
Influenza	11/2005	7
Pneumovax	03/2005	1
Twinrix	08/2002	3
Td	04/1996	1

Diabetic Indices

Type	Value	Most Recent
A1C	0.071	12/2005
LDL	2.41	12/2005
BP	135/75	02/2006
Urine Microalb	0.02	08/2005
Eye Exam		05/2005
Home Gluc (average)	7.4	01/2006

Figure 2. *An example of an Electronic Health Record (Note: The individual is fictional)*

Internet of Things is helping hospitals avoid, mitigate, or predict adverse events by focusing on integrating medical devices into a smart network of monitoring tools linked to the

EHR. A lot of providers are working on integrating and streamlining all of the patient-generated health data in their EHRs. The real time data from sensors, tablets, smartphones, and peripherals will soon be captured in EHR.

The HIPAA Security Rule establishes national standards to protect individuals' electronic personal health information that is created, received, used, or maintained by a covered entity (a healthcare provider in our context). The Security Rule requires appropriate administrative, physical and technical safeguards to ensure the confidentiality, integrity, and security of electronic protected health information. [8]

In an IoT connected Healthcare system, the CIA terms could be appropriately defined as the following:

- Confidentiality: the EHRs of patients are given access only to the professionals who have the adequate access privileges
- Integrity: the EHRs of the patients can be altered only by the professionals who have the adequate access privileges
- Availability: the EHRs of the patients and the other services revolving it should be available to the appropriate professionals without any conflict whenever needed

The HIPAA requirements ask the patients (IoT data owner) to check if the following security features are addressed for his/her practice.

- ePHI encryption
- Auditing functions
- Backup and recovery routines
- Unique user IDs and strong passwords

- *Role- or user-based access controls*
- Auto time-out
- Emergency access
- Amendments and accounting of disclosures

1.2 Thesis Motivation

1.2.1 Access Control in an IoT Environment

There are a variety of Techniques for providing security to the data that are transmitted and stored in the cloud [9]. However, most of these Techniques' main vision is to keep the data safe and secure from the third parties. This is inconsistent with the data sharing requirements of most real-world applications. The solutions to manage and provide internal security to the massive volume of data produced and stored by the IoT objects are yet to mature.

There are few Access Control methods that might suit various IoT applications. The literature survey shows that there is some good amount of cryptographic Techniques that would help to achieve the given fine grained access control by allowing the data to be decrypted only by the people with appropriate privileges. However, it is also evident that such cryptographic Techniques are implemented assuming an unlimited amount of computation and energy, which unfortunately cannot be provided by the IoT devices.

There are few light-weight access control encryption protocols [9,12] that can account for the resource constraints of a typical IoT device and can give the device the ability to encrypt based on appropriate access control policies and store in the cloud. The decryption can be done at a trusted proxy or at the client. However, these protocols are more suited to file-systems on the cloud rather than conventional DBMS which sound to be a more sensible storage option when it comes to storing EHRs on the Cloud. This thesis work addresses this problem of providing a fine grained access control to the EHR data stored on the cloud and at the same time taking into account the various resource constraints of an IoT device in healthcare.

1.2.2 Mitigating Query Latency

In an IoT application say Healthcare, which comprises of a variety of entities like doctors, nurses, practitioners, medical students, administrators, accountants, etc., every user would be interested in a particular type of data. For example, doctors belonging to the oncology department would be more interested in the vitals of patients diagnosed with cancer, while the accountants would be more interested in the billing information and other monetary details of the patients. That is, every type of user, based on his role will have his/her very own interest space. This results in the user querying for the same data over and over. We can try to exploit this nature of the user's workload to obtain the results faster.

Caching is an obvious solution to provide faster retrieval of frequently used data. However, when it comes to real-time IoT like a Smart Healthcare Environment, we might have to consider various factors into account when it comes to caching: if there is enough storage capacity at the Caching Node, how frequently we are allowed to hit the stale data and which kind of data can never go stale in the cache. In addition to this, we must also make sure that there is not a lot of contacts made between the server and the client in the process of ensuring cache consistency because in an IoT Environment where there would be a huge number of clients subscribing to a server, letting a lot of connections open is not a wise idea. This problem of providing a caching option for a healthcare IoT scenario is also considered and addressed in this thesis work.

1.3 Thesis Organization

The rest of the thesis work is organized as follows:

- Chapter 2: a literature survey of the existing access control mechanisms and caching approaches in a Client-Server Model
- Chapter 3: proposal of an Attribute Based Access Control mechanism and the method of achieving it in a conventional RDBMS. The chapter also deals with the proposal of novel Attribute Based Cache Coherency that would provide a faster query retrieval while adhering to the criticality requirements of the data
- Chapter 4: experimental evaluation
- Chapter 5: conclusion and future work

CHAPTER 2. LITERATURE REVIEW

As discussed in Chapter 1, there is a lot of work done on the various security mechanisms that could guarantee an end-end security in an IoT Environment.

2.1 End-to-End IoT security

When a data is said to be End-to-End secure, it means that the data is not prone to external attack right from the time of data generation until the time of data consumption. In terms of IoT, the data has to be secure when it is generated at the sensor and it has to maintain the same level of security until it reaches the end-user. We can divide the end to end security into two phases: i) Phase I – security while data generation and transmission to the Cloud and ii) Phase II – data security while residing in the Cloud.

Traditionally, we use AES for most of the security demanding applications. However, the cost demanded by these traditional security algorithms are pretty high for an IoT device. An IoT device which is considered to be constrained in terms of energy and computation needs algorithms that would suit their specifications. A lot of work has been done on the chip level security of the IoT devices. In the literature, a lot of light weight cryptography algorithms like PRESENT [12] have been proposed for the IoT producer devices like the sensors. So using these feasible and light weight encryption algorithms, the produced data can be encrypted and thus sent to the Cloud in a secure way, sparing any sort of attack that might happen during the transit.

Cloud security is also widely studied and nowadays, the Cloud Service Providers encrypt the data before storing them [13]. Cloud security solutions for a lot of IoT applications like Smart Grid [14], eHealth [15], Robotics [16] have been proposed. All these methods proposed in the literature, together can help achieve the end-to-end security in an IoT Environment, by keeping the data encrypted until the end-user decrypts it. However, they don’t take into account the various loopholes that are inside the organization like inappropriate users accessing critical data, leakage of sensitive information from inside parties to the wrong hands, etc. So a proper access control that meets the IoT constraints and at the same time restricts inappropriate users from having access to certain data has to be implemented.

2.2 Access Control Models in Information Security

At the cloud, when the data is encrypted and secure, we can trust that no other third party gets access to the data. However, in order to ensure a secure way of data sharing within the organization, some kind of access control mechanism has to come into picture. In the field of information security, access control means prohibition of irrelevant users from accessing data that are beyond their rights and privileges. Some of the established access control schemes are: Mandatory Access Control (MAC), Discretionary Access Control (DAC), Role Based Access Control (RBAC) and Attribute Based Access Control (ABAC).

Under MAC, Bell-LaPadula [17] Model deals with the confidentiality of the information and Biba Model deals with the integrity of the information. The Models had a similar hierarchical approach that would prevent unrestricted users to read/write a restricted file. This is achieved by assigning certain labels to the subjects and objects: Top Secret (highest priority), Secret, Confidential and Unclassified (least priority), that dictates that a subject assigned with a particular label would only be able to access the objects that are below the level of his/her label. This was sufficient in a computer system where a course-grained access control was sufficient. But in the context of IoT where there would be plenty of users accessing the data, categorizing them into a handful of labels would not serve the purpose of restricting various kinds of users from accessing undesirable data. Thus MAC fails to provide a fine-grained Access Control for the huge amount of IoT data.

On the other hand, DAC [18] resolves to provide a pretty fine-grained access control by using the concept of Access Control Matrix / Access Control Lists. This list has the details containing which user is authorized to provide which resource. Although this provides a great distinction among users, the granularity is so fine-grained that it cannot be adopted for an IoT Environment given the huge volume of data on the Cloud and also the huge volume of users trying to access the data. In other words, maintenance of such a large Access Control List would be so arduous in an IoT Environment.

Role Based Access Control [19] is an alternative approach that gained a great importance in cloud security. It is about providing access rights based on the roles possessed by the user in an organization. It combines the properties of MAC and DAC and implements a good level of granularity among the users. The RBAC has two separate mappings: i) one from the various users to various roles and ii) one from various roles to privileges. To elaborate, role is an abstraction to create the user behavior and their assigned duties. In this way, access to an object is permitted based on the user's role and not the user himself. That is, RBAC provides a means of naming and describing many-to-many relationships between individuals and rights unlike a one-one relations in the case of DAC. Also in RBAC, when the roles are removed or revoked, it is not mandatory to change the access permissions that are assigned to that roles. This scheme overcomes the disadvantages of the previous Models and it is more flexible paving way for many enterprises and organizations to formulate their access policies without violating their organization structure and policies. This kind of approach makes real-time cloud based data access control more practical. Works like [20] and [21] have focused on implementing the RBAC scheme in a Cloud architecture to ensure security and at the same time maintain the properties of an organization's hierarchy. However, researchers soon realized a serious problem with RBAC called 'Role Explosion' [22] in large enterprises. Role Explosion is due to the necessity of thousands of roles being fashioned for different collections of permissions. In addition to the role explosion problems, RBAC also suffered from role management problems since it was difficult to retain and reallocate the roles and permissions for users after change of their positions and job titles in the organization hierarchy. Although the identity management problem is well understood in RBAC, research performed over the last several years suggests that the separate problem concerning the proliferation of roles is not generally appreciated within the academic and practitioner communities.

In order to overcome the drawbacks of RBAC, Attribute Based Access Control (ABAC) [23] came into picture. ABAC realizes that one of the main problems with the design of RBAC is its two-way mapping: user to role and role to object. ABAC converges this two-way mapping to a single-way mapping – users with attributes to objects with Access Control Policies (ACP). In an ABAC Model, every user will be given various attributes and every object will be assigned with a particular access control policy that might suit its nature. For example, in a healthcare IoT, a user's attributes could be (doctor, oncology, level=3), (nurse, oncology), (medical student), etc., An object on the other hand would be the medical data like the EHRs. In this case, the data of a cancer diagnosed patient would be assigned with an ACP that would look like (doctor AND oncology), resulting in the access of this particular data only by users owning both the 'doctor' and 'oncology' attributes. ABAC systems can create a variety of definitions and very fine-grained purposeful expressions. Some examples would be:

- An early stage cancer patient's data could be defined with an ACP (doctor AND oncology) OR (nurse AND oncology)

- A critical cancer patient's data could be defined with an ACP (doctor AND oncology AND level>3)

In this way, ABAC makes the system design more flexible and more scalable. ABAC's working is mainly based on analyzing the user's attributes and thereby restricting the unauthorized users from having access to sensitive data. Thus we observe that an ABAC Model would suit the IoT applications better than any other access control Models.

2.3 Attribute Based Access Control in a Cloud Environment

A good amount of work exists in the literature that implements an ABAC scheme in a Cloud Environment [24-28]. Almost all of the implementations are based on using the Attribute Based Encryption [23] Technique since the data on the cloud has to go encrypted. Two variants of ABE were also proposed based on various use cases:

- Cipher text-Policy ABE (CP-ABE) [29]: A secret key is associated with user's attributes and Cipher-text is labeled with an ACP
- Key-Policy ABE (KP-ABE) [30]: Attributes are used to describe the Cipher-text and policies are built into users' keys.

CP-ABE is observed to be more suitable in a Cloud Environment. Before uploading data files to the cloud server, the data owner can specify an ACP and encrypt data under this policy. Only such users whose attributes satisfy the policy can successfully decrypt the encrypted data using their secret keys.

A variety of works focus on the hierarchy issues [31], attribute revocation and policy updating issues [28, 32, 33] in an ABE scheme. But all these works assume the source of encryption to be computationally powerful to perform an expensive cryptographic algorithm.

However, in most of the IoT applications where the source data are from WSNs and other resource constrained devices, a full-fledged ABE would not be feasible. Works like [34-36] focus on developing a lightweight ABE approach to suit an IoT Environment by incorporating some pre-processing and tweaking of the traditional ABE algorithm. Although this kind of encryption scheme answers the question of energy efficiency in IoT devices, this might not be suitable for all kind of IoT applications. It might be applicable for individual file systems, which can be encrypted using the light-weight ABE and be stored on the cloud and at the other end can be decrypted by the appropriate users. But for applications demanding a relational database storage on the Cloud like the EHRs, this might not be a suitable option. This is because an ABE or any of its variants would not accommodate an encrypted query processing on the dataset. So we shall conclude that ABE is not the right way to achieve ABAC for IoT systems demanding a relational database system.

Along those lines, DBMask [37] is the first practical implementation of an ABAC on relational databases. DBMask proposes a novel Technique that separates fine grained access control from encrypted query processing when evaluating SQL queries on encrypted data and enforces fine grained access control at the granularity level of a column, row and cell based on an expressive attribute-based group key encryption scheme. DBMask is inspired by the CryptDB project [38, 39], which is the first research effort that has systematically investigated access control for SQL queries on encrypted relational data. However, DBMask overcomes some of the limitations of CryptDB related to encrypted query processing and fine grained access control. In DBMask, each cipher-text is only a single layer of encryption as opposed to onions of layers of encryption in CryptDB and the cipher-text supports comparison operation. Also unlike CryptDB, the data are never decrypted to weaker encryptions inside the cloud server and therefore the security of the data does not weaken over time. However, both DBMask and CryptDB are designed with web applications in mind and is not suitable for IoT application scenarios because: they rely on a trusted proxy which intercepts the communication and applies encryption/decryption transparent to the user. This approach apart from causing security conflicts also adds a computation overhead of 25% [40].

Talos [40] is a system that is developed exclusively for IoT applications, that stores the IoT data securely in a Cloud database while still allowing query processing over the encrypted data. Talos architecture, combined with the fine-grained access control mechanism proposed by DBMask would help us in achieving a fine-grained access control for the huge amount of IoT data stored in a Relational Database Cloud Systems.

2.4 Caching Techniques

2.4.1 Client Side caching

Caching is one of the best solutions to account for the Latency problems in most of the Client-Server applications. In a Client-Server Environment, caching can be done at any end – at the Client or at the Server. In conventional relational database systems, caching is done at the Server Side [41, 42] in order to reduce disk traffic by storing the frequently used pages in the cache. However, in order to hit this cache, a query has to be sent through the network to the server. This had been happening under the assumption that the Client Side is not capable enough to hold the frequently hit data. But given today's smart phones (the primary IoT consumer devices), their capability is not just limited to sending queries but also storing certain data and querying over the stored data is easily possible. A lot of study has been done on the various ways to achieve Client Side caching [43-46].

Client Side caching is appropriate to our application, given the diverse needs of various users belonging to the organization. Based on the roles and privileges of different users, they might have different interest spaces. Server Side caching might not be applicable to this scenario, where the data requirements are disjoint and there is no guarantee that the consequent users would need the same data. Client Side caching proves to be a better choice for our given IoT application because each client can have his/her own interest space based on their attributes. It also reduces network traffic by serving the queries locally rather than sending every single query to the server.

2.4.2 Cache consistency

One of the main factors while designing a cache Model is deciding on the cache consistency method. Data can be invalidated at appropriate instances to maintain the data consistency in the cache. The invalidation Techniques can be divided into two categories: server initiated (push-based) invalidation and client initiated (pull-based) invalidation. Server initiated invalidation sends an Invalidation Report(IR) to the appropriate clients at the appropriate time, thus invalidating potential inconsistent data. The methods proposed in literature for server initiated invalidation can be broadly classified into two broad categories: stateless [47], where every update has to be broadcasted through IRs to all clients periodically and stateful [48, 49], where metadata has to be maintained about particular clients and any update has to be multi-casted to appropriate clients who are interested in a particular data. The greediness of the stateful approach and the maintaining of metadata and the network traffic that could be caused by stateful approach make server initiated invalidation unsuitable for our application. Client Side invalidation methods proposed in the literature can again be broadly classified into: client polling [50], where a client polls the server at regular intervals to check if the cache is up-to-date and TTL based Techniques [51-54], where a heuristic Time-To-Live (TTL) are applied for every entity of the resource, which the client caches in addition to the data. The data is deemed to be invalid if the corresponding Time-To-Live in the Cache expires. Client Polling, again creates a lot of contact between the client and the server. TTL is the most inexpensive method in terms of network traffic compared to all the given methods.

However, TTL can create ambiguity between the strong and weak consistency Models. The lower the TTL, higher the data consistency and vice-versa. In a healthcare application, the critical data could be assigned with a lower TTL while the non-critical data could be assigned with a higher TTL, so as to reduce the number of total queries to the server. A variety of Techniques for TTL assignment including a lot of adaptive TTL schemes have been proposed in the literature [55, 56]. But none of the works consider the diverse type of users and the various levels of criticality the users hold towards various data. If this is considered and if a caching with an appropriate cache consistency is designed, we can further reduce the average Latency.

CHAPTER 3. PROPOSED ATTRIBUTE BASED ACCESS CONTROL WITH ATTRIBUTE BASED CACHE COHERENCY

3.1 Background

In an Internet of Things scenario, there are two types of people: i) data producers and ii) data consumers. Data producers could be sensors, aggregators or other others who push data periodically/aperiodically to the cloud. Data consumers could be actuators or the end-users who need the produced data for observation and to gain other meaningful insights. Since the data producer devices mostly comprise of wireless sensor networks and embedded devices, they are already widely studied in terms of their resource constraints (computation, communication and other security concerns). On the other hand, the end-devices on the consumers' end are still prone to issues like network congestion due to enormous number of users sending uplink/downlink requests, power consumption of the client devices, Latency constraints given the time-criticality of an application and unauthorized data access.

From the perspective of a Healthcare IoT, we consider two major issues: i) unauthorized access of data by the internal and external stakeholders of EHR. Internal stakeholders of the EHR involves the hospital staff and the external stakeholders involves government agencies, insurance agencies and other vendors. An important attribute of EHR is that the EHR of a patient can be shared between more than one health organization. Considering this vast number of users of the EHR data, the Healthcare IoT should be made completely secure by allowing only the authorized users to access their corresponding data and ii) the time-criticality of data. A Healthcare IoT is considered to be the most time-critical IoT application today. So the end-users will expect the required data to be available to them within a reasonable deadline. For instance, a group of specialty doctors observing a critical patient might not have the patience to wait for a long time for the queried data.

To overcome these two issues, this thesis proposes the following approaches: i) a modified DBMask architecture by applying Talos to make it suitable to the IoT applications. We utilize the attribute based grouping and query rewriting Technique similar to the approach proposed in DBMask. We extend this work to account for hierarchical groups and appropriate query rewriting. By applying Talos, we eliminate the proxy, thus securing the keys at the client and also by reducing a significant amount of computational overhead to fit DBMask into the IoT scenario and ii) a novel client-Side caching Technique that takes into account the granularity of criticality of data across a wide variety of user groups obtained from (i). The idea here is that for a particular group of users, a particular field might be highly critical and consistency of the data cannot be compromised. However, there could be another group of users who also have access to the same data but they can tolerate staleness to a particular threshold. An example could be, the users belonging to the Admin department might have access to few of the vitals like weight of a patient. They would need the information just to fill some forms or files that’s not really highly significant. On the other hand, there could be a group of medical students who observe the patients as a part of their training. They can tolerate some amount of staleness of the weight data as they wouldn’t encounter critical patients. However, a group of dieticians have to know the most recent weight data in order to observe the sudden drop/gain of weight of a particular patient. That is, the level of criticality depends on the attributes of the end-user.

3.2 System Architecture

The architecture that is suitable for achieving a fine-grained access control of data with minimum Latency across users in IoT applications is given in Figure 3.

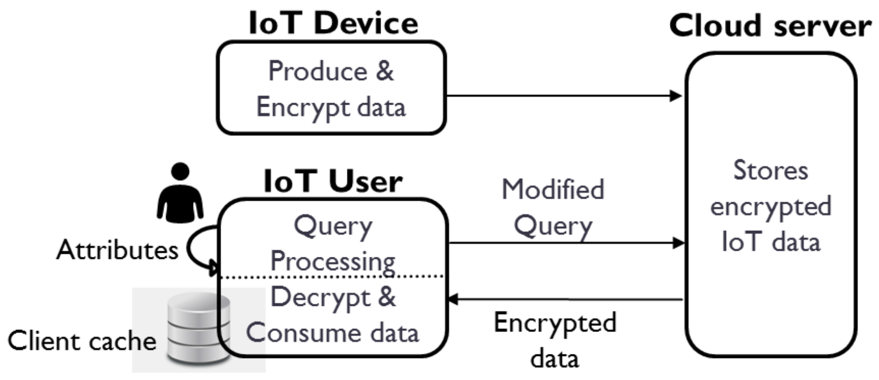


Figure 3. *The Proposed Architecture for achieving Fine Grained Access Control on IoT Cloud with Minimal Average Latency*

Here, encrypted EHR along with periodic vital values collected from the IoT devices like wearable devices and other fitness trackers like Fitbit are stored in a secure RDBMS in the Cloud. This forms the data producer and the storage parts of the architecture. Data consumers for this application include mobile and web apps through which the hospital staff view the patient data. Here due to the vast amount of work done at the Cloud level and device level security, we assume that the data is secure during the commute between the IoT devices and the Cloud and also at the Cloud. We propose a query rewriting Technique based on the attributes of the hospital staff to achieve the fine grained access control of data. Also, we shall see a local cache at the Client Side, which stores the data queried from the Cloud Server.

3.3 Fine Grained Access Control of Encrypted Data

There are some key definitions involved in the approach:

<u>DEFINITION 1 – Attribute</u>

Attributes are entities such as roles, department, level, etc., that a person holds in an organization

Example:

Attributes of User 1 - doctor, oncology, 3

Attributes of User 2 - receptionist, admin department

<u>DEFINITION 2 - Access Control Policy</u>

Let T be a table. On T, an ACP (c, p) can be defined, where c is a particular cell of T and p is the policy pertaining to c. This p is a combination of attributes and Boolean operators.

Example:

ACP over a critical cancer patient's data – (doctor AND oncology AND level > 3)

ACP over the billing information of a patient – (account)

<u>DEFINITION 3 - Groups</u>

A group is a set of users satisfying the same attribute conditions.

Groups are obtained by converting ACP into Disjunctive Normal Form(DNF). For every distinct disjunctive clause, a group is formed.

Example:

Let's assume an ACP (c,p) defined over a cell c : (A1) AND (A2 OR A3), where A1, A2 and A3 are the attributes

After converting to the DNF, ACP (c,p) becomes : (A1 AND A2) OR (A1 AND A3).

Now users satisfying the clause (A1 AND A2) become group G1 and users satisfying the clause (A1 AND A3) become group G2.

DEFINITION 4 - Group Hierarchy

Group Hierarchy is a partial ordered tree obtained by linking the relationship between various groups created. The root node forms the most privileged group.

An example of this hierarchy is given in Figure 4. Based on the organization hierarchy, different structures of groups hierarchy can be formed as seen in Figure 4 (Case (i) – a hierarchical tree, Case (ii) – a hierarchical graph)

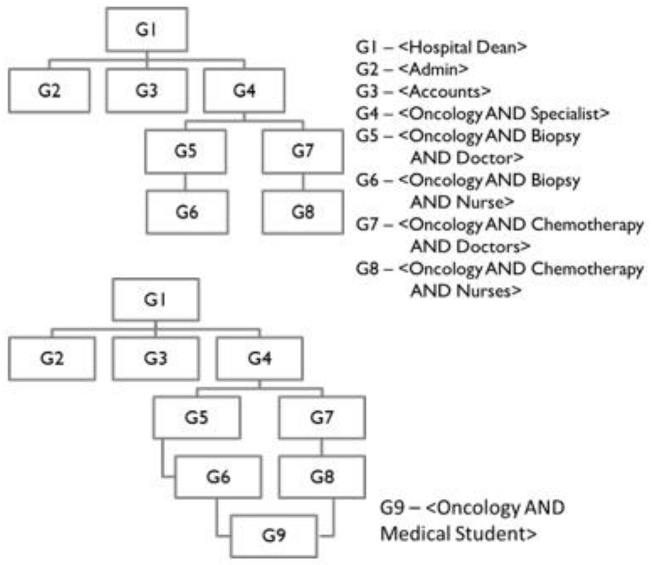

G1
G2
G3
G4
G5
G7
G6
G8
G1 – <Hospital Dean>
G2 – <Admin>
G3 – <Accounts>
G4 – <Oncology AND Specialist>
G5 – <Oncology AND Biopsy AND Doctor>
G6 – <Oncology AND Biopsy AND Nurse>
G7 – <Oncology AND Chemotherapy AND Doctors>
G8 – <Oncology AND Chemotherapy AND Nurses>
G1
G2
G3
G4
G5
G7
G6
G8
G9
G9 – <Oncology AND Medical Student>

Figure 4. *Group Hierarchy – Case (i) a hierarchical tree (top), Case (ii) a hierarchical graph (bottom)*

PHASE I: System Initialization

Decision of ACPs:

The EHR vendor in consultation with the healthcare authority determines the list of ACPs and puts them in a table in the RDBMS on the cloud. ACP to Groups conversion also occur at this step. The resultant table is the Assignment table, an example of which is show in Table 1.

Table 1. *Assignment Table*

Table	Column	Condition	ACP	Groups
Patient	ID, Name, Diagnosed, Severity, <Other vitals>	Severity = ‘High’	(Oncology AND Specialist)	G4
Patient	ID, Name, Diagnosed, Severity, <Other vitals>	Severity = ‘Medium’	(Oncology AND Biopsy AND (Doctor OR Nurse))	G5, G6
Patient	ID, Name, Insurance, SSN, Billing_info		(Accounts)	G3

<u>Appending additional columns to the existing main tables:</u>

In addition to the required columns, the table structure is modified to add one comparison friendly column to every existing column (see Table 2 vs Table 3). These columns are known as the corresponding label columns of the respective columns. The cells of these columns contain the name of the group, who has access to the corresponding data column. This information is obtained from the Assignment table. Please note the insurance and billing_info columns are not illustrated in the example tables given the space restrictions.

Table 2. *Patient Table before modification*

ID	Name	Diagnosed	Severity	<Vitals>	SSN
1	Alice	Cancer	High	…	xxx-xx-xxxx
2	Bob	Cancer	Medium	…	xxx-xx-xxxx
3	Carol	Cancer	High	…	xxx-xx-xxxx

Table 3. *Patient Table after modification*

ID	ID_label	Name	Name_label	Diagnosed	Diagnosed_label	Severity	Severity_label	<Vitals>	<Vitals>_label	SSN	SSN_label
1	G3, G4	Alice	G3, G4	Cancer	G4	High	G4	…	G4	xxxx-xx-xxxx	G3
2	G3, G6	Bob	G3, G6	Cancer	G6	Medium	G6	…	G6	xxxx-xx-xxxx	G3
3	G3, G4	Carol	G3, G4	Cancer	G4	High	G4	…	G4	xxxx-xx-xxxx	G3

However, when more than one group is obtained from the Assignment table and if the groups have a link in the group hierarchy, the node whichever acts as the child (the less privileged group) is given to the label columns. We use the Nested Set Model to represent the hierarchies in RDBMS and determine the relationship between various nodes. This relationship is given in Table 5, 6 and a similar Technique that is discussed in Phase III Group Retrieval Query is used to obtain the group names.

Populating attribute information:

Various users, their attributes, their authentication details, and the associated group names are loaded on the Cloud server into the User-Attribute Table and the Group Tables respectively.

Group Hierarchy in RDBMS:

The hierarchy of the created groups is stored in the table form using the Nested Set Model in the Cloud database. In case of a hierarchical graph structure, the graph is converted into a hierarchical tree and then the Nested Set Model is applied over the obtained tree as indicated in Figure 5.

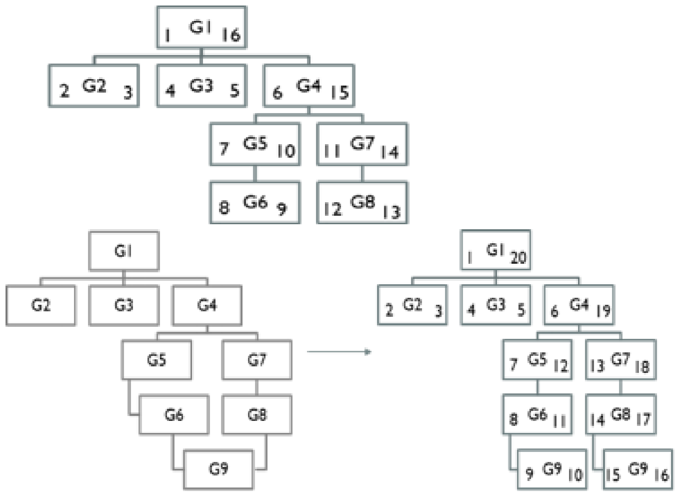


Figure 5. *Nested Set Model for Case (i) (top), Hierarchical Graph to Hierarchical tree conversion for Case (ii) (bottom left), Nested Set Model for Case (ii) (bottom right)*

PHASE II: Data Producer End - Data Encryption and Insertion

There are constant data inserts/updates from the producer end, which could be from the sensors or directly by the hospital staff observing the corresponding patients. All these data are encrypted using appropriate lightweight algorithms and uploaded into the Cloud server.

PHASE III: Data Consumer End - User Query and Data Retrieval

User's actual data query is split into two different queries: i) the group retrieval query based on the attributes of the user. It is executed over the User-Attribute table to obtain the attributes, the Group Table to determine the attributes' group and the group hierarchy table to obtain the child group(s) ii) the query which is rewritten using the group names obtained from (i).

For an example in case (i), let’s assume that a particular login of a user gives the Attributes <Oncology, Biopsy, Doctor> from the User-Attribute Table and group G5 from the Group Table. The user queries for the Patient ID, Name and other vitals for the Patient ID:2. The query on the Nested Set Table [see Table 4, 5] in this case is:

Query 1: *Group Retrieval Query*

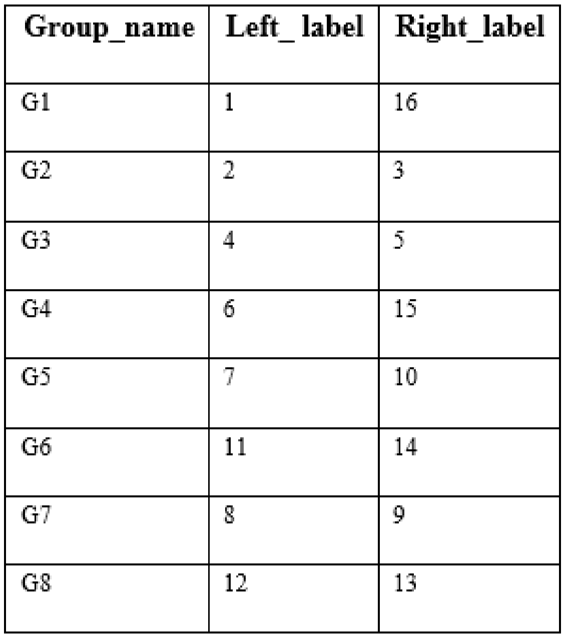
SELECT DISTINCT *b.group_name*

FROM *nested_set a, nested_set b*

WHERE *a.group_name* = 'G5'

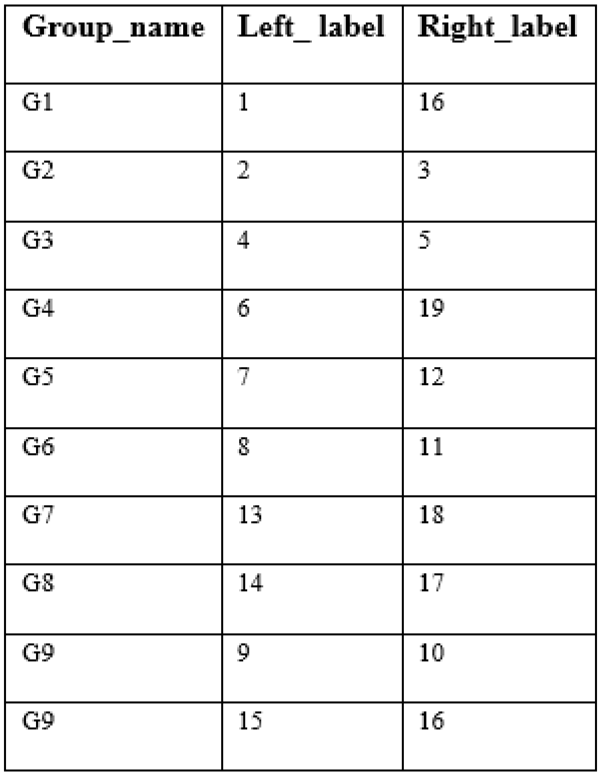
AND *b.left_label* >= *a.left_label*

Table 4. *Nested Set Table for Case (i)*



Group_name	Left_ label	Right_label
G1	1	16
G2	2	3
G3	4	5
G4	6	15
G5	7	10
G6	11	14
G7	8	9
G8	12	13

Table 5. *Nested Set Table for Case (ii)*



Group_name	Left_ label	Right_label
G1	1	16
G2	2	3
G3	4	5
G4	6	19
G5	7	12
G6	8	11
G7	13	18
G8	14	17
G9	9	10
G9	15	16

The results obtained from Query 1 are G5 and G6. The Query 2 generated by the user gets rewritten to Query 3, which gets executed on the modified patient table. Now the Cloud server returns the encrypted results of Query 3 to the user’s device, which then decrypts and displays the results to the user.

Query 2: *Actual query*

SELECT *Pid, Name, <Vitals>*

FROM *Patient*

WHERE *Pid* = 2

AND *b.right_label* <= *a.right_label*

Query 3: *Modified Query*

SELECT *Pid, Name, <Vitals>*

FROM *Patient*

WHERE *Pid* = enc(2)

AND (*Pid_label* **LIKE** ‘%G5%’ **OR** *Pid_label* **LIKE** ‘%G6%’**)**

AND (*Name_label* **LIKE** ‘%G5%’ **OR** *Name_label* **LIKE** ‘%G6%’**)**

AND (*<Vitals>_label* **LIKE** ‘%G5%’ **OR** *<Vitals>_label* **LIKE** ‘%G6%’)

Thus the user successfully retrieves the Name and other vitals of Patient ID:1. If the same user queries for the Patient ID, Name and other vitals for the Patient ID:1 or the SSN detail of any patient, no data will be retrieved. This is because the group G5 is not entitled to viewing any of these data and the rewritten query ensures this access control.

3.4 Client Side Caching and Attribute based Cache Coherency

As discussed above, minimal Latency is very important in an IoT Environment. We propose a novel Attribute Based Cache Coherency Technique for a Client Side cache. We have a remote cloud server from where the data is retrieved from and the local Client cache that stores the queried data based on an appropriate invalidation scheme. The invalidation scheme takes into account the fact that different data can be of different criticality to different groups of users. It is not really necessary to maintain a uniform Latency across all groups of users. From an end-user's perspective, there are some data that are highly critical while the rest wouldn't be as critical. When we say that a data is critical to the user, we emphasize on a couple of factors: i) the user is expecting the most recent version of the data and ii) the user tends to query this data more frequently than the other data.

The conventional Client Side caching stores the recently queried values in the Client's cache and retrieves values from the cache the next time the same query or a subset of the query is posed, thus bypassing the communication to the server. When caching comes into picture, a couple other factors must also be taken into account – cache replacement policy and cache consistency. Given the sufficient memory space in most of the smart devices used to view the data and also given the limited number of queries posed by the user in a day, a highly stringent cache replacement policy is not desirable. We stick to the traditional LRU based data replacement policy, which would be applicable to our scenario as well. On the other hand, there are a variety of invalidation Techniques to maintain the consistency of the cached values at the client. Given the enormous amount of overhead with the server initiated invalidation Techniques in an IoT Environment, we go with the client initiated TTL(Time-To-Live) based invalidation Technique. Here the key is that the data in the cache is no longer trustable once the corresponding TTL expires.

$$T_1 - T_2 > TTL$$

After the TTL expiration, the client has to fetch the data from the Cloud and the same process goes on. However, by considering a uniform TTL for fields across various clients, the problem of a huge number of clients in an IoT Environment still exists. The cost of hitting the Cloud is going to be the same for all the clients, but the benefit obtained is not the same across all the clients. In order to achieve a better cost-benefit factor, we have to make sure that the clients who might not be benefitted by fetching a certain data shouldn't be spending more than how much that data would benefit the client. Based on this, we shall modify the existing TTL invalidation Technique to suit our needs. The TTL dictates when the client should reach out for the server and when it has to look for data within itself. So if we could tailor the TTL values according to the criticality needs of various clients, we can actually reduce the total number of requests hitting the server. Also, the cost and the benefit for a particular data for a particular client would be balanced.

Now the task at hand is to assign TTL values to the various fields. It is completely reasonable to assign different TTL values to different fields. Because certain fields like age, location, etc., are long lived, while certain fields like the BP, glucose and other vitals would rapidly go stale in the cache. Here, there are only two criticality levels: low (data that are long lived) and high (data that rapidly change). So a longer TTL value can be assigned to the high critical fields and a shorter TTL can be assigned to low critical fields. Similarly, there could be more number of criticality levels. We shall further tune the TTL assignment to exploit the granularity in criticality of a particular field to a particular user. That is every group of users has a certain tolerance to certain kind of data. Considering the variety of roles and the corresponding nature of the job, a data cannot be of the same criticality to all groups of users. It should be categorized into a particular level of criticality only based on how relevant or important it is to perform the job of a particular user group. So based on how critical a particular data is to a particular user group, every user group has its own tolerance towards a particular data based on which the criticality is determined. That is when a user has more tolerance towards a data, the data is assumed to be less critical for the user and the vice versa. Based on this tolerance of a user group (determined by the user's attributes) to a data, we shall tune the TTL of a field in the Client's cache accordingly. That is when the user group is more tolerant to a particular data field, it means that he can tolerate a certain degree of staleness. So it's wise to assign the user group with a higher TTL value for the given data field. Similarly, when a user group is less tolerant to a particular data field, it means that he cannot tolerate staleness beyond a very less degree. So we must assign the user group with a lower TTL value for the given data field. By doing this, we actually cut down the total number of requests going to the Cloud at a given time. This clears traffic for the critical queries that demand data from the Cloud server, thus resulting in a comparatively lesser Latency.

In Table 6, every cell is a grain and the notations used in each cell is used to depict the number of subjects belonging to the particular grain. As we discussed above, initially when ABCC in not used there are only two grains – low and high. Using ABCC, we try to make these coarse grains into fine grains by creating a new grain at every level and moving some people across to the newly created grain. C1, C2, C3 etc., are called the Grain constants, which determine how many people are moved from the high criticality grain to the newly created medium criticality grain. The Grain constants would range from 0.01 to 0.99. In the table the TTL is increasing from left to right, with the left most being TTL 0 where the queries always have to go to the server and the right most being TTL infinity where the queries always go to the cache. For example, consider the field 'weight' in an EHR. The field could change somewhat rapidly for someone who is undergoing rigorous diet changes or for someone who is under a variety of medications. While, this field could be of high criticality to subjects with Attributes <Department: Nutrition, Dietician> or other attribute sets who keep track of the weight of the patient constantly, it could be of medium criticality to <Department: Nutrition, Medical Students> and could be of low criticality to <Department: Admin>. So if we cut down the number of subjects in the high criticality group, the traffic in the Cloud at a time gets reduced and thus the average Latency for a query would also decrease as the number of grains increase.

Table 6. *Coarse-grains of Attribute groups to Fine-grains based on data criticality levels*

Number of Grains	***Always to Server (High critical)***	***Cache (Criticality decreases towards the right)***			
N = 2	X	Y			
N = 3	(1 - C1).X	C1.X	Y		
N = 4	(1 - C1).X	(1 – C2).C1.X	C2.C1.X	Y	
N = 5	(1 - C1).X	(1 – C2).C1.X	(1 – C3).C2.C1.X	C3.C2.C1.X	Y
...	…	…	…	…	…

By implementing this attribute based Client Side caching, we tailor the caching scheme according to the user needs. This way, we can ensure better Latency without compromising the consistency of the critical data for a user group.

CHAPTER 4. PERFORMANCE EVALUATION

4.1 Experimental Setup

A basic IoT testbed has to be established to simulate the proposed approach. The actual architecture consists of the IoT producer devices, Cloud server and IoT consumer devices. Since the implementation is going to remain the same at the IoT producer devices, we form a testbed only with the Cloud server and an IoT consumer device. The following platforms were used for implementation: i) An Android Asus Google Nexus that is used as an IoT consumer device (the Client). It has an in-built local database called the SQLite, which can be used as the local cache and ii) Amazon Web Services – Relational Database System (AWS RDS) is used for implementing the Cloud server. We assume that the EHR information of the patients collected by the hospital staff and the incoming sensor values are stored encrypted in the Cloud. The encryption algorithm that is used for encrypting the String data is Advanced Encryption Standard in Cipher Block Chain mode (AES-CBC), while the one used for encrypting numerical data is Order Preserving Encryption (OPE), which preserves the numerical ordering of plain-texts. We use it in our implementation because when we consider querying over encrypted data, we have to take into account the various types of operators that the user may use: =, <>, >, <, >=, <=. BETWEEN, LIKE and IN. Now we have adequately encrypted values in the Cloud server, that can cater any type of query. The specifications of various components are given in Table 6. A testbed is setup using the given components for performing extensive experimentation.

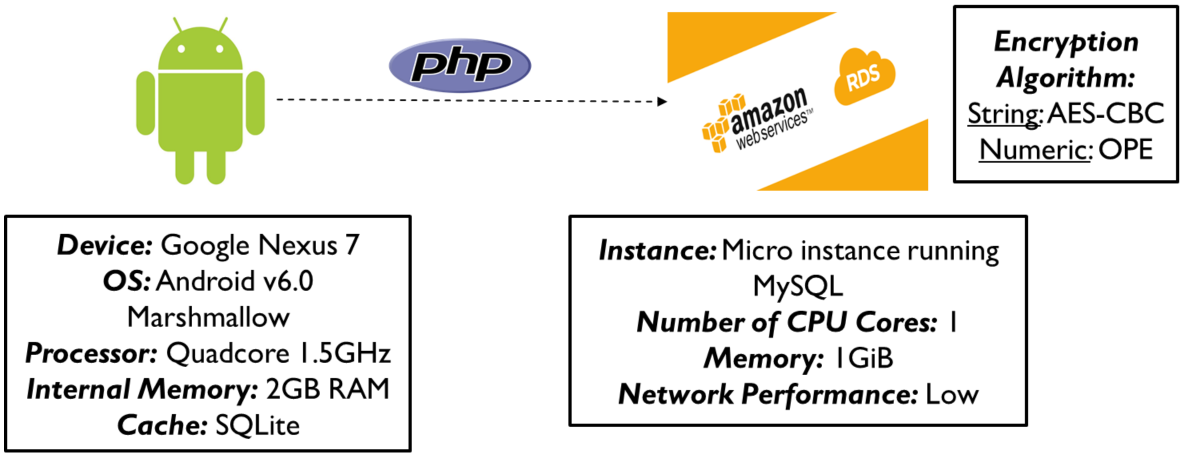


Figure 6. *Experimental Setup*

To understand our proposed architecture in terms of the evaluation platform, consider the following sequence of steps:

- A user query is sent from the Android device
- Before sending the user query, the Android device does the following:
    - checks if the data is in the local SQLite database (after making the appropriate cache invalidation). If yes, it decrypts and displays the data to the user
    - If no, the device rewrites the existing query and sends it to the Cloud
- A connection to the Cloud is made from the Android device by establishing a HTTP connection between the device and the Cloud using the server's hostname and port number provided by Amazon Web Services
- Encrypted Query Processing happens over the data on the Cloud and the result is returned
- The Android device stores the obtained encrypted results in the local cache
- It then decrypts the data and displays it to the user

4.2 Performance Evaluation

For the evaluation purpose, we run the queries on the main data table on the Cloud comprising of a million encrypted rows with 25 actual columns. We append 25 additional comparison friendly columns for every actual column. The actual database size is 75 MB with the additional columns compromising of around 20MB. The other tables like the assignment table and the group table account for a negligible 0.05MB together. Since the Cloud server provides a very good amount of storage at a reasonable price, utilizing additional storage to achieve the required results wouldn't be a problem. At the user end, the Android device faces an overhead of … memory for the Cache implementation. Given that only the IoT producer devices have resource constraints, while the IoT consumer devices have adequate processing and energy capacity these days, we call a memory overhead of 50KB (for over 100 records) as negligible.

The evaluation comprises of a series of experiments to evaluate the efficiency of ABAC with the proposed ABCC caching Technique. Obviously, ABAC would give a lower Latency with a Client Side cache. The question now is if ABCC could give a relatively lower Latency without making a lot of compromises. To perform the regular ABAC with cache (without ABCC) analysis, we assume two different kinds of data: critical and non-critical. The critical data always goes to the Cloud and the non-critical data hits the cache until the given TTL expires, after which it goes to the Cloud and gets the data to the cache. To perform ABAC with cache (with ABCC) analysis, we assume more than two different kinds of data. That is, we have varying levels of criticalities based on which we vary the ABCC from coarse-grains to fine-grains of TTLs. The TTL to perform the cache invalidation is varied from 0 to infinity across various grains. By doing so, for a given resource, not all the users come to the Cloud contention at the same time. We assume a total of 3000 employees in the hospital having access to a particular data and we perform experiments to observe the Latency variations for various number of grains for that particular data [see Figure 7]. The minimum number of grains is two and ABAC with cache utilizes this. The results show that when the number of grains is increased from two, the average Latency for a given user considerably decreases. Also, this experiment is repeated for various factor sizes to observe the Latency changes. The following table gives the step by step fragmentation of every grains. Initially we assume that out of the 3000 hospital staff who have access to a particular data, 2000 subjects query from the cache since the data is critical to them and 1000 subjects query from the cache. Then we increase the number of grains at each level, thus creating one additional group at every level that compromises of the subjects for whom the data is mid-critical and towards which they have a certain amount of tolerance. Table 7 shows the fragmentation of grains and the number of subjects belonging to each grain when the grain constant is 0.5. For example, subjects belonging to the mid-critical region in (N=3) = 0.5 * subjects belonging to high critical region in (N=2).

The experiment follows the above way of fine-graining the criticality levels and is repeated for various grain constants. We shall observe that for a smaller grain constant (0.25), there is not a significant change in Latency. This is due to the fact that only a very less number of subjects are moved from lower TTL grain to the next higher TTL grain, resulting in most of the queries going to the Cloud server. As the grain constant increases, we shall observe a drastic change in average Latency, given that the higher grain constants move more number of subjects from the lower TTL grain to the next higher TTL grain. However, the average Latency tends to saturate after a particular number of grains. This is due to the fact that after a certain number of grains, there are not many subjects to move from the current lower TTL grain to the next higher TTL grain, making the lower Latency contributed by the higher TTL grain negligible to the average Latency.

Table 7. *An Example used for evaluating fine-grained Attribute Groups based on data criticality*

No. of Grains	***To Server***	***To Cache (Criticality level decreases towards the right)***					
N=2	2000	1000					
N=3	1000	1000	1000				
N=4	1000	500	500	1000			
N=5	1000	500	250	250	1000		
N=6	1000	500	250	125	125	1000	
N=7	1000	500	250	125	62	62	1000

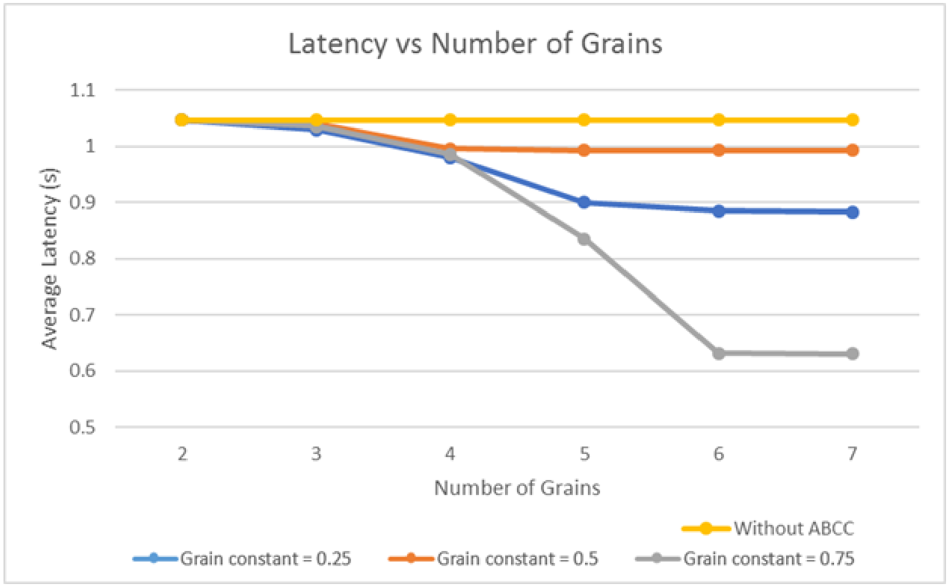


Figure 7. *Evaluation of Average Latency vs Number of Grains in ABCC*

Yet another important metric that must be analyzed while experimenting on a cache consistency Model is staleness. In an IoT Environment, there will not only be INSERTs but there will also be UPDATEs. Thus, we further analyze ABAC with ABCC to determine the various staleness percentages by performing the following experiments:

i) data staleness for increasing number of grains (constant update rate) [see Figure 8]

ii) data staleness for varying update rates (constant number of grains n) [see Figure 9]

While performing (i), we vary the grain constants to observe the relationship between staleness and the number of grains for different grain constants. We assume that a data in the Cloud is updated for every one second and our Android testbed makes 1 query per second. As we can see, as the number of grains increase, the staleness also increases. Because, more number of grains means that more the number of subjects using the cache and higher the probability for accessing the stale data. However, the staleness is comparatively low and almost remains constant for the grain constant = 0.25. This is because the number of subjects belonging to the high critical region for grain constant 0.25 are more than the ones belonging to the high critical region for grain constants 0.5 and 0.75. More the number of subjects in the high critical regions, lower the overall staleness since the staleness faced by the entire high criticality group is almost zero (given that these queries always go to the Cloud). Also, for every number of grains we can see that the higher grain constant gives the highest staleness. This is because, as the grain constant increases the number of subjects in the newly created grain (having higher TTL) also increases. When there are more number of subjects using higher TTL for a given data, the probability of these subjects hitting stale data is also high. The graph for various grain constants begin to saturate at n=7 or higher. The reason is that the number of subjects that we move to the next newly created grain becomes less at a certain point after which the staleness contributed by these subjects becomes negligible.

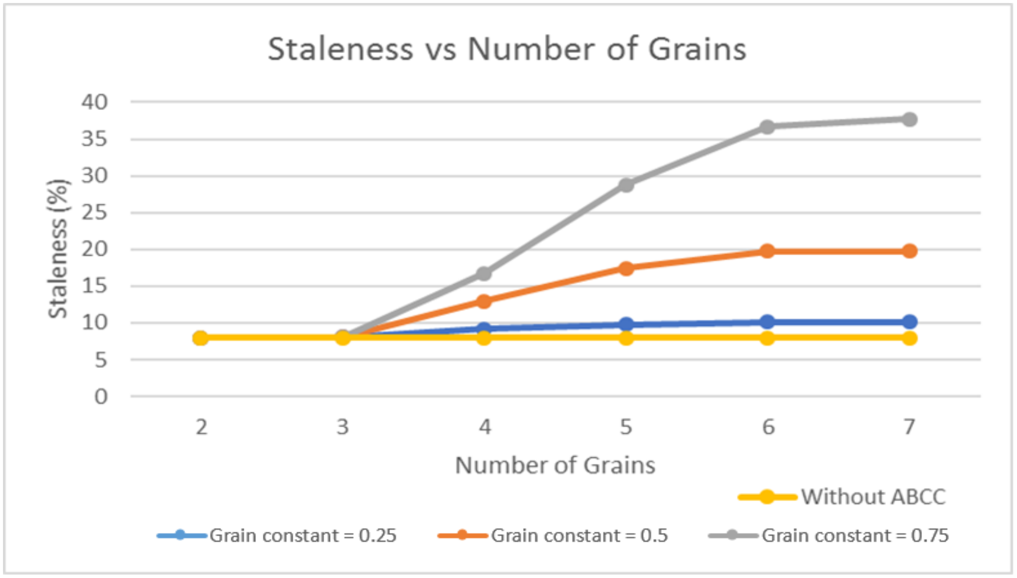


Figure 8. *Evaluation of Staleness vs Number of Grains in ABCC*

While performing (ii), we keep the number of grains as constant (in our experiment, the number of grains = 5). We study the relationship between the staleness and the update rate with different grain constants. Undoubtedly, the staleness keeps dropping as the update rate decreases. However, there is a significant difference in the staleness percentage for various grain constants for a given update rate. This is again due to the fact that the number of subjects in the high critical region keeps increasing for increasing grain constants, thus decreasing the overall staleness for the same. The graph shows a high staleness percentage for the grain constant 0.75. However, it is to be noted that the high critical data always gives the least staleness percentage for a given update rate [see Figure 10]. As we can see in the figure, as the criticality decreases the staleness increases since the subjects utilizing the lower critical data use the cache for a longer time (due to the higher TTLs) than the subjects utilizing the higher critical data.

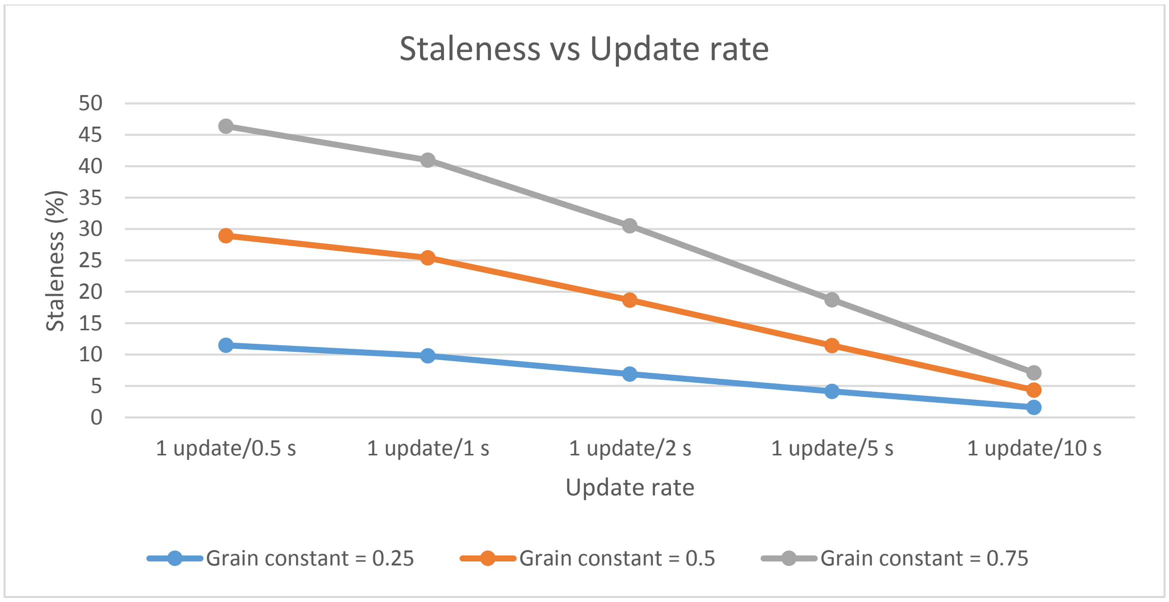


Figure 9. *Evaluation of Staleness vs Update Rate in ABCC*

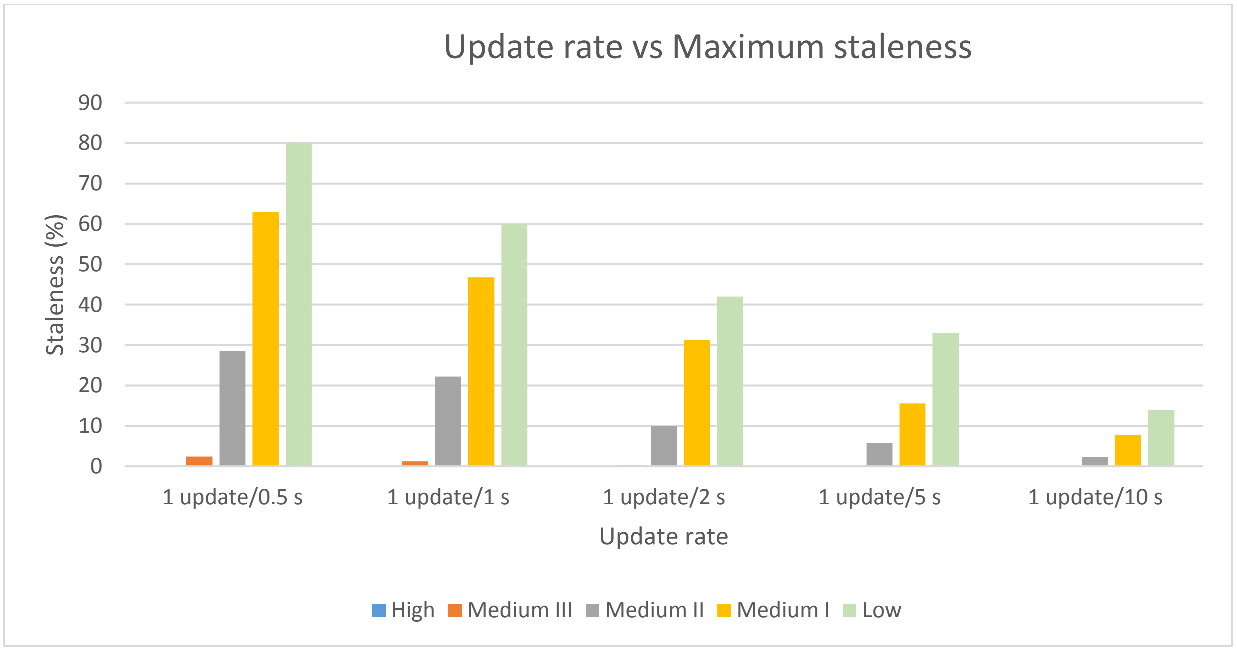


Figure 10. *Evaluation of Average Latency vs Number of Grains in ABCC for various Grain constants*

CHAPTER 5. CONCLUSIONS AND FUTURE WORK

5.1 Summary

The recent developments in IoT open doors for the realizing various sectors' dreams like Smart Health, Smart Cities, Smart Homes, etc., However, the security issues in IoT also keeps growing with the developments made in IoT. Irrespective of whichever sector adopts IoT, data security is very important given the huge amount of distinct and sensitive data being stored in the Cloud. The question that is aimed to be answered in the thesis work is "how to provide a fine-grained access control to the huge volume of data stored in the IoT Cloud?". To answer this question, we first conducted an extensive literature survey on the state-of the-art Access Control protocols and Techniques. Once the limitations in adopting certain protocols/Techniques are identified, we proposed: an Attribute Based Access Control for IoT RDBMS Cloud for providing a fine-grained access control and an Attribute Based Cache Coherency for a minimal average Latency.

This thesis work discusses IoT and its overall architecture by presenting various IoT components mainly from the perspective of data security. In Chapter 3, two contributions are made:

- An Attribute Based Access Control (ABAC) Model for IoT data on Cloud Relational Databases. It includes an appropriate architecture and an approach for query rewriting with encrypted query processing to enforce ABAC. It also deals with the hierarchical relationship between various attribute groups and incorporates Nested Set Model to deal with various possible hierarchical structures

- A novel Attribute Based Cache Coherency (ABCC) approach that utilizes a Client Side caching Technique and exploits the ‘different data – different users – different criticality’ nature of the IoT data. Under this approach, we also try to make fine-grained criticality levels of data based on users’ attributes and tailor the TTL based cache invalidation scheme to provide better Latency performance. We introduce the term grain constant and vary it under different Environments to analyze its impact on Latency performance

The proposed Techniques are implemented using the experimental setup and evaluated using extensive experiments. A comparison is made between the ABAC with ABCC and without ABCC in terms of average Latency and staleness. We can infer from the experiments that,

- ABAC with ABCC gives better average Latency as the number of grains increases. However, practically after a certain number of grains, the average Latency starts to saturate given that the number of subjects in the newly created grains is very less. Thus, if an organization has a lot of diverse IoT data, where a given data is highly critical only to a certain number of users we can make the grain constant higher, thus resulting in less average Latency.
- In case of staleness in case of ABCC, lesser grain constants give less staleness percentage while the higher grain constants give a higher staleness percentage. However, we can also see that the staleness percentage also saturates after a certain number of grains.
- The grain contributing to the higher staleness percentage is the low criticality grain. Based on the following practical assumptions, we can call the higher staleness due to the low critical grain negligible: i) the people belonging to the low critical grain has good amount of tolerance to the data and ii) the people tend to query the low critical data only occasionally. Even if the given data is highly critical only to a large number of users we

can make the grain constant smaller but still achieve a comparatively lesser Latency against ABAC without ABCC.

5.2 Future Work

This works considers the IoT RDBMS data on the Cloud and addresses the access control and Latency problems prevalent in IoT. The potential future works could be:

- Scaling the proposed idea to a large number of organizations. This work considers the fact that the IoT data on the Cloud is shared by multiple parties inside an Organization. IoT data, nowadays is shared among various organizations to achieve interoperability and also get a lot of meaningful insights.
- Implementing the ABAC with ABCC Technique for non-relational databases. IoT data are now increasingly stored in NoSQL given its scalability and flexibility. So an appropriate mechanism to achieve Access Control and better Latency in a NoSQL Environment could be a valuable extension of the current work.
- Enforcing ‘Write’ or ‘*’ property, since the current work focuses only on the ‘Read’ property. In this way, we can restrict only the authorized users to modify/update the existing dataset, thus preserving data integrity.

REFERENCES

1. Schoenberger CR, Upbin B. The internet of things. Forbes Magazine. 2002 Mar 18;169(6):155-60.
2. Shah SH, Yaqoob I. A survey: Internet of Things (IOT) technologies, applications and challenges. In Smart Energy Grid Engineering (SEGE), 2016 IEEE 2016 Aug 21 (pp. 381-385). IEEE.
3. Ronen E, O'Flynn C, Shamir A, Weingarten AO. IoT goes nuclear: Creating a ZigBee chain reaction. Weizmann Institute of Science, Tech. Rep. 2016 Nov.
4. Nawir M, Amir A, Yaakob N, Lynn OB. Internet of Things (IoT): Taxonomy of security attacks. In Electronic Design (ICED), 2016 3rd International Conference on 2016 Aug 11 (pp. 321-326). IEEE.
5. Kaye K. FTC: fitness Apps can help you shred calories–and privacy. Online: http://adage.com/article/privacy-and-regulation/ftc-signals-focus-health-fitness-data-privacy/293080. 2014.
6. Miorandi D, Sicari S, De Pellegrini F, Chlamtac I. Internet of things: Vision, applications and research challenges. Ad Hoc Networks. 2012 Sep 30;10(7):1497-516.
7. Alasmari S, Anwar M. Security & Privacy Challenges in IoT-Based Health Cloud. In Computational Science and Computational Intelligence (CSCI), 2016 International Conference on 2016 Dec 15 (pp. 198-201). IEEE.
8. HIPAA Guide to Privacy and Security of Electronic Health Information https://www.healthit.gov/sites/default/files/pdf/privacy/privacy-and-security-guide.pdf
9. Goyal TK, Sahula V. Lightweight security algorithm for low power IoT devices. In Advances in Computing, Communications and Informatics (ICACCI), 2016 International Conference on 2016 Sep 21 (pp. 1725-1729). IEEE.
10. Kuusijärvi J, Savola R, Savolainen P, Evesti A. Mitigating IoT security threats with a trusted Network element. In Internet Technology and Secured Transactions (ICITST), 2016 11th International Conference for 2016 Dec 5 (pp. 260-265). IEEE.
11. Sivaraman V, Gharakheili HH, Vishwanath A, Boreli R, Mehani O. Network-level security and privacy control for smart-home IoT devices. In Wireless and Mobile Computing,

Networking and Communications (WiMob), 2015 IEEE 11th International Conference on 2015 Oct 19 (pp. 163-167). IEEE.

12. Bogdanov A, Knudsen LR, Leander G, Paar C, Poschmann A, Robshaw MJ, Seurin Y, Vikkelsoe C. PRESENT: An ultra-lightweight block cipher. In International Workshop on Cryptographic Hardware and Embedded Systems 2007 Sep 10 (pp. 450-466). Springer Berlin Heidelberg.
13. Google Cloud Database. Security and Integration with Google Cloud. [Online] November, 2015: https://cloud.google.com/sql/ (accessed November 20, 2015).
14. Guan Z, Li J, Wu L, Zhang Y, Wu J, Du X. Achieving Efficient and Secure Data Acquisition for Cloud-supported Internet of Things in Smart Grid. IEEE Internet of Things Journal. 2017 Apr 3.
15. Alasmari S, Anwar M. Security & Privacy Challenges in IoT-Based Health Cloud. In Computational Science and Computational Intelligence (CSCI), 2016 International Conference on 2016 Dec 15 (pp. 198-201). IEEE.
16. Horton M, Chen L, Samanta B. Enhancing the security of IoT enabled robotics: Protecting TurtleBot file system and communication. In Computing, Networking and Communications (ICNC), 2017 International Conference on 2017 Jan 26 (pp. 662-666). IEEE.
17. Rushby J. The bell and la padula security Model. Computer Science Laboratory, SRI International, Menlo Park, CA. 1986.
18. Miller MS, Yee KP, Shapiro J. Capability myths demolished. Technical Report SRL2003-02, Johns Hopkins University Systems Research Laboratory, 2003. http://www. erights. org/elib/capability/duals; 2003 Mar.
19. Sandhu RS. Role-based access control. Advances in computers. 1998 Dec 31; 46:237-86.
20. Zhou L, Varadharajan V, Hitchens M. Integrating trust with cryptographic role-based access control for secure cloud data storage. In Trust, Security and Privacy in Computing and Communications (TrustCom), 2013 12th IEEE International Conference on 2013 Jul 16 (pp. 560-569). IEEE.
21. Zhou L, Varadharajan V, Hitchens M. Achieving secure role-based access control on encrypted data in cloud storage. IEEE transactions on information forensics and security. 2013 Dec;8(12):1947-60.

22. Elliott A, Knight S. Role Explosion: Acknowledging the Problem. In Software Engineering Research and Practice 2010 Jul (pp. 349-355).
23. Goyal V, Pandey O, Sahai A, Waters B. Attribute-based encryption for fine-grained access control of encrypted data. In Proceedings of the 13th ACM conference on Computer and communications security 2006 Oct 30 (pp. 89-98). Acm.
24. Zhu Y, Huang D, Hu CJ, Wang X. From RBAC to ABAC: constructing flexible data access control for cloud storage services. IEEE Transactions on Services Computing. 2015 Jul 1;8(4):601-16.
25. Riad K, Yan Z, Hu H, Ahn GJ. AR-ABAC: A New Attribute Based Access Control Model Supporting Attribute-Rules for Cloud Computing. In Collaboration and Internet Computing (CIC), 2015 IEEE Conference on 2015 Oct 27 (pp. 28-35). IEEE.
26. Balamurugan B, Shivitha NG, Monisha V, Saranya V. A Honey Bee behaviour inspired novel Attribute-based access control using enhanced Bell-Lapadula Model in cloud computing. In Innovation Information in Computing Technologies (ICIICT), 2015 International Conference on 2015 Feb 19 (pp. 1-6). IEEE.
27. Liu Z, Jiang ZL, Wang X, Yiu SM, Zhang C, Zhao X. Dynamic Attribute-Based Access Control in Cloud Storage Systems. In Trustcom/BigDataSE/I SPA, 2016 IEEE 2016 Aug 23 (pp. 129-137). IEEE.
28. Lv Z, Chi J, Zhang M, Feng D. Efficiently attribute-based access control for mobile cloud storage system. In Trust, Security and Privacy in Computing and Communications (TrustCom), 2014 IEEE 13th International Conference on 2014 Sep 24 (pp. 292-299). IEEE.
29. Bethencourt J, Sahai A, Waters B. Ciphertext-policy attribute-based encryption. In Security and Privacy, 2007. SP'07. IEEE Symposium on 2007 May 20 (pp. 321-334). IEEE.
30. Attrapadung N, Libert B, De Panafieu E. Expressive key-policy attribute-based encryption with constant-size ciphertexts. In International Workshop on Public Key Cryptography 2011 Mar 6 (pp. 90-108). Springer Berlin Heidelberg.
31. Wang G, Liu Q, Wu J. Hierarchical attribute-based encryption for fine-grained access control in cloud storage services. In Proceedings of the 17th ACM conference on Computer and communications security 2010 Oct 4 (pp. 735-737). ACM.

32. Hur J, Noh DK. Attribute-based access control with efficient revocation in data outsourcing systems. IEEE Transactions on Parallel and Distributed Systems. 2011 Jul;22(7):1214-21.
33. Yang K, Jia X, Ren K. Attribute-based fine-grained access control with efficient revocation in cloud storage systems. In Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security 2013 May 8 (pp. 523-528). ACM.
34. Yao X, Chen Z, Tian Y. A lightweight attribute-based encryption scheme for the Internet of Things. Future Generation Computer Systems. 2015 Aug 31; 49:104-12.
35. Jo M, Odelu V, Das AK, Khan MK, Choo KK. Expressive CP-ABE Scheme for Mobile Devices in IoT satisfying Constant-size Keys and Ciphertexts. IEEE Access. 2017 Feb 16.
36. Guo F, Mu Y, Susilo W, Wong DS, Varadharajan V. CP-ABE with constant-size keys for lightweight devices. IEEE transactions on information forensics and security. 2014 May;9(5):763-71.
37. Sarfraz MI, Nabeel M, Cao J, Bertino E. DBMask: fine-grained access control on encrypted relational databases. In Proceedings of the 5th ACM Conference on Data and Application Security and Privacy 2015 Mar 2 (pp. 1-11). ACM.
38. Popa RA, Redfield C, Zeldovich N, Balakrishnan H. CryptDB: protecting confidentiality with encrypted query processing. InProceedings of the Twenty-Third ACM Symposium on Operating Systems Principles 2011 Oct 23 (pp. 85-100). ACM.
39. Popa RA, Redfield C, Zeldovich N, Balakrishnan H. CryptDB: processing queries on an encrypted database. Communications of the ACM. 2012 Sep 1;55(9):103-11.
40. Shafagh H, Hithnawi A, Dröscher A, Duquennoy S, Hu W. Talos: Encrypted query processing for the internet of things. In Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems 2015 Nov 1 (pp. 197-210). ACM.
41. Zhang Y, Li D, Zhu Z. A server side caching system for efficient web map services. In Embedded Software and Systems Symposia, 2008. ICESS Symposia'08. International Conference on 2008 Jul 29 (pp. 32-37). IEEE.

42. Zeng Z, Veeravalli B. Hk/T: A novel server-side web caching strategy for multimedia applications. In Communications, 2008. ICC'08. IEEE International Conference on 2008 May 19 (pp. 1782-1786). IEEE.
43. Keller AM, Basu J. A predicate-based caching scheme for client-server database architectures. The VLDB Journal—The International Journal on Very Large Data Bases. 1996 Jan 1;5(1):035-47.
44. Al Ridhawi I, Mostafa N, Masri W. Client-Side Partial File Caching for Cloud-Based Systems. In Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld), 2016 Intl IEEE Conferences 2016 Jul 18 (pp. 909-914). IEEE.
45. Liu X, Ma Y, Liu Y, Xie T, Huang G. Demystifying the imperfect client-side cache performance of mobile web browsing. IEEE Transactions on Mobile Computing. 2016 Sep 1;15(9):2206-20.
46. Froese KW, Bunt RB. The effect of client caching on file server workloads. In System Sciences, 1996., Proceedings of the Twenty-Ninth Hawaii International Conference on, 1996 Jan 3 (Vol. 1, pp. 150-159). IEEE.
47. Wu CC, Fang JF, Hung PC. A counter-based cache invalidation scheme for mobile Environments with stateless servers. In Communications, Computers and signal Processing, 2003. PACRIM. 2003 IEEE Pacific Rim Conference on 2003 Aug 28 (Vol. 2, pp. 623-626). IEEE.
48. Chand N, Joshi R, Misra M. Efficient cache invalidation in mobile Environment. In India Annual Conference, 2004. Proceedings of the IEEE INDICON 2004. First 2004 Dec 20 (pp. 107-112). IEEE.
49. Chand N, Joshi RC, Misra M. Energy efficient cache invalidation in wireless mobile Environment. In Personal Wireless Communications, 2005. ICPWC 2005. 2005 IEEE International Conference on 2005 Jan 23 (pp. 244-248). IEEE.
50. Ahmad NM, Geok TK. Enhanced client polling with multilevel pre-fetching algorithm for wireless networks. Journal of Communications and Networks. 2007 Mar;9(1):43-9.
51. Alici S, Altingovde IS, Ozcan R, Cambazoglu BB, Ulusoy Ö. Timestamp-based result cache invalidation for web search engines. In Proceedings of the 34th international ACM

SIGIR conference on Research and development in Information Retrieval 2011 Jul 24 (pp. 973-982). ACM.

52. Blanco R, Bortnikov E, Junqueira F, Lempel R, Telloli L, Zaragoza H. Caching search engine results over incremental indices. In Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval 2010 Jul 19 (pp. 82-89). ACM.
53. Fawaz K, Artail H. DCIM: Distributed cache invalidation method for maintaining cache consistency in wireless mobile networks. IEEE Transactions on Mobile Computing. 2013 Apr;12(4):680-93.
54. Shukla SS, Ingle YS. Cache maintenance using distributed cache invalidation method and time to live mechanism in wireless mobile network. In Engineering and Technology (ICETECH), 2015 IEEE International Conference on 2015 Mar 20 (pp. 1-4). IEEE.
55. Alici S, Altingovde IS, Ozcan R, Cambazoglu BB, Ulusoy Ö. Adaptive time-to-live strategies for query result caching in web search engines. In European Conference on Information Retrieval 2012 Apr 1 (pp. 401-412). Springer Berlin Heidelberg.
56. Chatterjee D, Tari Z, Zomaya A. A task-based adaptive TTL approach for web server load balancing. In Computers and Communications, 2005. ISCC 2005. Proceedings. 10th IEEE Symposium on 2005 Jun 27 (pp. 877-884). IEEE.