Spring 2015

A Study of Access Control for Electronic Health Records

Sergio Vincent Senese
Governors State University

A STUDY OF ACCESS CONTROL FOR ELECTRONIC HEALTH RECORDS

By

Sergio Vincent Senese
A.S., Moraine Valley Community College, 1998
B.S., St. Xavier University, 2001

THESIS

Submitted in partial fulfillment of the requirements

For the Degree of Master of Science,
With a Major in Computer Science

Governors State University
University Park, IL 60466

2015

Acknowledgments

I would like to thank all my graduate level teachers at Governors State University: Dr. Soon-Ok Park, Dr. Steven Shih, Dr. Xueqing Tang, Professor Steve Hyzny, and Department secretary Nancy Rios for all their support, help, and guidance throughout my graduate degree program. I would like to thank my parents for all that they have done for me during my lifetime. I would like to thank Edward Martig, a former undergraduate professor who always helped me and inspired me to further my Computer Science education. Dr. Dinbang Xu, a former Governors State University professor and advisor. Lastly, I would like to thank Linda who has always be there for me and helped in every way possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract

The expansion between Information Technology and Healthcare has created many new options for both disciplines, as well as challenges. One of these topics is the Electronic Health Record (EHR) and the push for a universal record. A challenge for this topic is access control: how to keep patient's personal health information secure, but at the same time accessible to all fields of healthcare and accomplish this within the federal privacy laws made by our government. This study focuses on the idea of a single EHR containing all the different medical information for all the areas of healthcare for a patient. This single EHR would be stored in a database and its use secured though the use of access control using a hierarchy of user groups, which would be divided into different roles to assign access privileges. This access control method would be implemented by possibly using mechanisms such as Bell-LaPadulla Model, The Strawman Design, Public/Private Key algorithms, or other methods. The first goal would be to create this structure for a single entity (e.g., One Hospital, Clinic, or Doctor's office) and then progress to a distributed model where multiple entities can store and share information.

CHAPTER 1

Introduction 1.1

Healthcare over the years has become increasingly advanced in scientific breakthroughs, and in part due to the use of Information Technology and Computer Science. In the attempt to streamline the recording process of medical information the Electronic Health Record (EHR) was created and implemented to use across the healthcare field to digitize the massive amounts of paper-based records that are taking up space and contributing to medical errors. An EHR can contain important personal information such as name, address, phone number, birth date and Social Security Number. It also holds vital medical information about a person like diagnosis, treatment history, lab results, and prescriptions. There are great advantages to using electronic records more extensively, within both the offices of individual providers, where they are known as electronic medical records (EMRs), and when such records are linked across multiple providers, in which case they are known as electronic health records (EHRs). [32] Another benefit of the EHR is that vital medical information and patient history could be accessed at any location with ease of use and with immediate results. It also reduces the amount of time, money, and resources spent on copying multiple versions of a patient's history. One aspect of concern in the use of an EHR is security, who will be able to see such information and what kind of information will each person be given permission to view. Privacy is an underlying governing principle of the patient – physician relationship for effective delivery of healthcare. [7] A remedy to this quandary is the use of Access Control. Access Control has been a staple in Computer Science for years, being refined and tested in variant ways. The one aspect of Access Control that this research is attracted to is that of Row and Cell Level Security and the use of Labels to create a secure database for EHR patient information. The achievement of this research is to apply methods from Access Control, particularly Row and Cell Level Security using Labels to an EHR database to demonstrate the protection that such technology can offer a most important document.

Summary 1.2

Security is a need when dealing with privileged information, but more so when that information is people’s confidential medical records. In particular, the adoption of electronically formatted medical records, so called Electronic Health Records (EHRs), has become the primary concern for a broad range of health information technology applications and practitioners. [17] Electronic medical records are exposed to possible misuses and risks. The process in which these parcels of information are stored and accessed currently has to be questioned and scrutinized as to what and whom this information is being divulged to, as well as the need for a more secure process then one being currently implemented. Medical data is also susceptible to misuse by those seeking to profit from it. [9] For example, some companies make a business of buying and selling doctors' prescribing habits to pharmaceutical companies. [9] In an attempt to streamline EHR documents this thesis narrows in on applying access control to Electronic Health Records. The likelihood that information will be improperly disclosed depends on its value, and the number of people who have access to it. [6] In deploying access control to Electronic Health Records, overall security can be increased and patient care and quality will improve as well as overall healthcare costs.

Organization 1.3

This thesis contains a breakdown of access control into different methods and implementation of use and applying these theories to securing an Electronic Health Record. There is one main type of access control that is focused on in this research, using row and cell-level security (RLS/CLS) in a database scenario for securing patient information and implementing an EHR Application. In order to gain full knowledge on the subject of EHR’s and how they relate to the access control, a more detailed explanation is needed. This concentrated review of Electronic Health Records is included in Chapter 2, followed by a study of Access Control in Chapter 3. Chapter 4 explains how access control can be configured and its terminology. Chapter 5 describes the tool used to create Security Labels, the SQL Server Label Security Toolkit 1.5. Chapter 6 displays the EHR demo created to convey the theory of access control applied to use for an Electronic Health Record. Chapter 7 provides future work and research in the field of access control and security labels. The conclusion is presented lastly on the final chapter of this thesis.

CHAPTER 2

Electronic Health Record (EHR) 2.1

Electronic Health Record (EHR) is a compilation of medical records of an individual patient or population group transformed into a digital format. The record is capable of being utilized amongst various healthcare disciplines though networks to share vital information about a patients overall medical history. These records can include a wide assortment of data in various forms including, but not limited to personal information such as name, address, social security number, and birth date. An EHR will also include information like prescriptions, allergies, laboratory results, surgical history, radiology images, billing information and diseases. Most EHR's are created at a doctor's office, hospital, clinic, pharmacy or medical facility. This usually serves as an overall record of the patients interactions with these locations. EHR's are starting to be viewed as a resource to advance fulfilment for patients and help them achieve the quality of health care they are expecting. It also is used, as a process to reduce overall operational costs for health care providers and companies that provide insurance. EHR's sharing could be widespread with the eventual goal being that a patient's overall medical record is available anytime, anywhere, to any healthcare provider or location, which further provides additional benefits. One of the significant features of an EHR is that healthcare data can be created and overseen by authorized personnel in a digital format capable of being distributed with other personnel across more than one health care organization. EHRs are built to share material with other health care professionals and organizations – such as labs, specialists, surgeons, and pharmacists. EHRs and the capability to exchange health information electronically can help healthcare organizations provide higher quality and safer care for patients while producing noticeable improvements for the overall medical field.

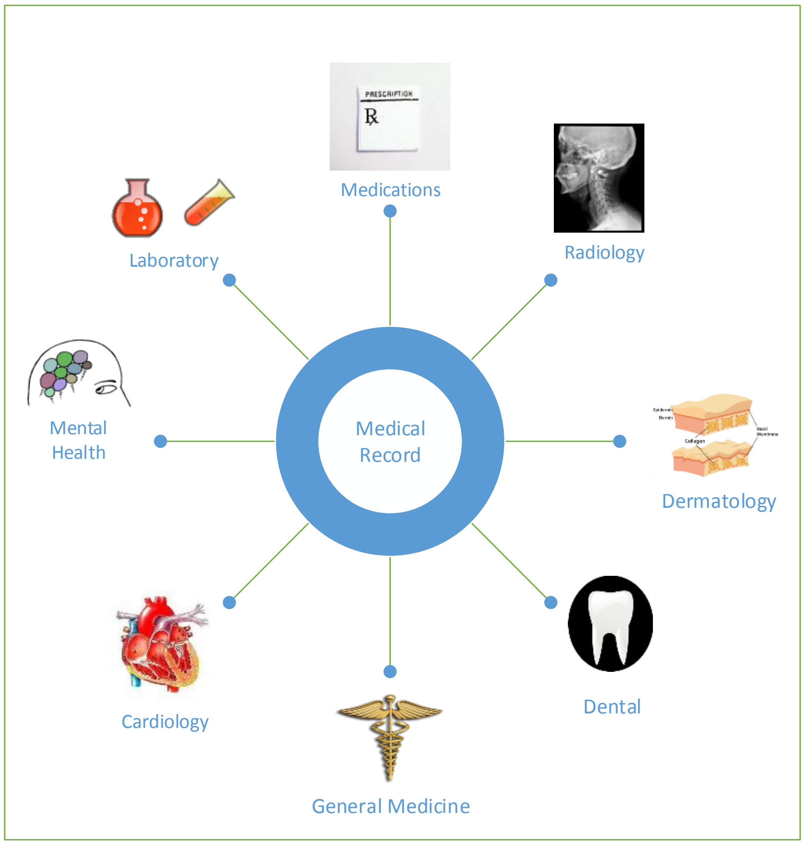


Figure 2.1: Medical Record Schema

EHR Standards 2.2

The following standards have been adopted for use in Electronic Health Records using Information Technology:

- ISO - ISO TC 215 provides international technical specifications for EHRs. ISO 18308 describes EHR architectures
- HL7 - a standardized messaging and text communications protocol between hospital and physician record systems, and between practice management systems
- DICOM - an international communications protocol standard for representing and transmitting radiology (and other) image-based data, sponsored by NEMA (National Electrical Manufacturers Association)
- Continuity of Care Record - ASTM International Continuity of Care Record standard
- HISA (EN 12967), a services standard for inter-system communication in a clinical information environment.
- CONTSYS (EN 13940), supports continuity of care record standardization.
- EN 13606, communication standards for EHR information
- ANSI X12 (EDI) - transaction protocols used for transmitting patient data

These standards are the basic building blocks to form the subjects in various healthcare EHR examples. HL7 has standards defined that form the generic information used across all HL7 protocol, while other standards define detailed structure and outline medical documents in terms of objects that model basic medical procedures. By implementing or conforming to a common form for EHR standards, an established relationship can be achieved between different medical information systems to communicate and share uniform medical data with each other. Therefore, the decisions on this unified platform of approval and selected sharing of medical information should be carried out with the understanding of the standards for the EHR.

Architecture 2.3

The architecture for the Electronic Health Record must have homogeneous framework fundamentals in order to facilitate auto processing and interchangeable operations. The data contained in these records must be secure and accessible by many different users such as medical staff, billing/financial personnel, and patients just to name a few. Basic architecture begins with a database to store information for the EHR. There are six requirements of an EHR database suitable for clinical use: available data range, access control, sharing control, search query performance, usability, and database management policy agreement. [19] This patient information can be shared through a network with firewalls provided to ensure security. At the database level is where access control will be the primary provider of information flow and availability. Access control will be discussed in depth in the next chapter. A secure wireless or wired network empowers healthcare professionals to access and update EHR records straight from a hospital or lab, providing a current, inclusive view of the patient's health where healthcare providers can utilize it to help the overall treatment of a patient. Considering the complexity of the medical work flow, the large number of health records and the variety of institutions, users and systems involved, it seems likely that checking these circumstances and conditions is slow and prone to errors. [12] Being able to access patient health information during a patient visit can lead to a better quality of treatment.

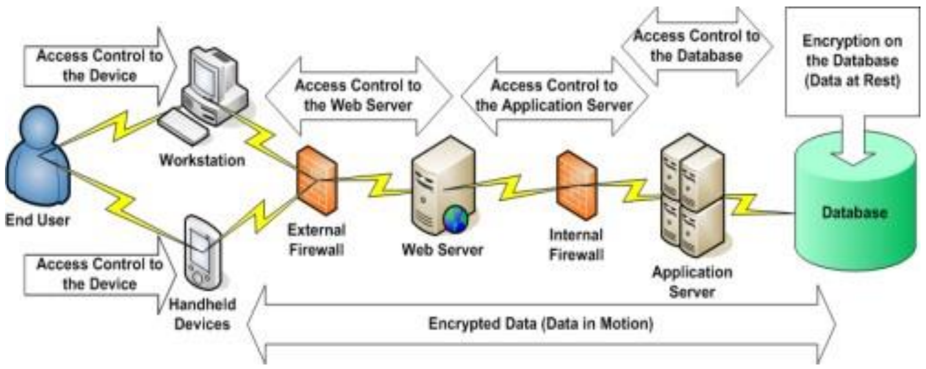

Access Control to the Device
Workstation
End User
Access Control to the Device
Handheld Devices
Access Control to the Web Server
External Firewall
Web Server
Access Control to the Application Server
Internal Firewall
Access Control to the Database
Application Server
Encryption on the Database (Data at Rest)
Database
Encrypted Data (Data in Motion)

Figure 2.2: EHR Security Layout [11]

This basic architecture can be implemented for a single entity such as a hospital, doctor’s office, or clinic to produce an Electronic Health Record that could be then shared in a national/regional Public Health System. Use of extensible markup language (XML) and resource description framework (RDF) standards are the current choice of consolidating World Wide Web technology and data structuring in attempting to enforce global Internet standards. This will allow clinical ontology’s better querying and information transmission, considering clinical information technology is still not yet refined or a top priority among the medical profession.

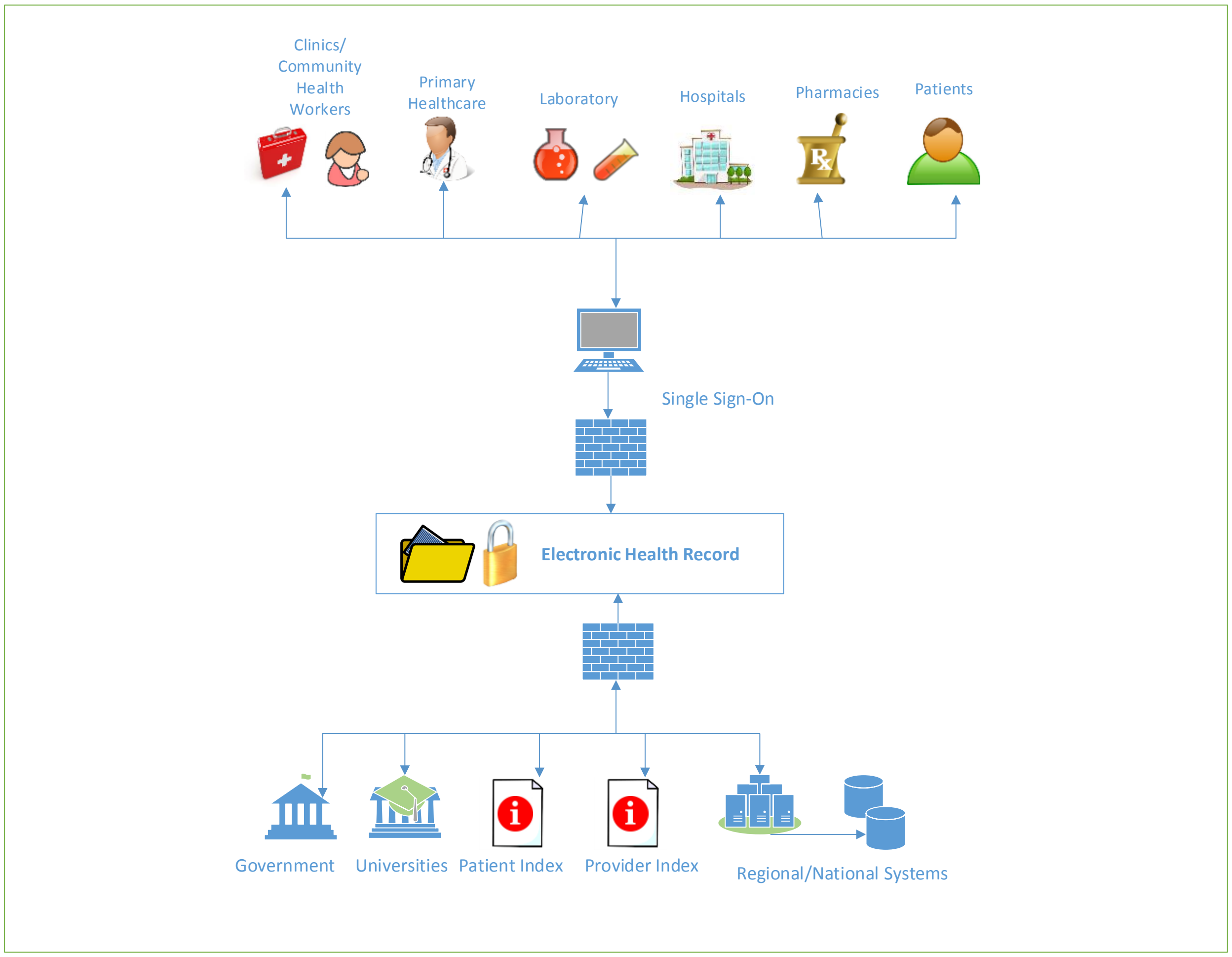


Figure 2.3: EHR Network

Database Type 2.4

The ideal database type for securing electronic health records must have some sort of principle behind it. The database must respect the disclosure of data it manages. It must allow present business operations to continue with nominal or no fluctuations to present systems, while safeguarding that disclosure fears are not a concern. The solution is what is known as a Hippocratic Database. A Hippocratic Database is a set of technologies that manages disclosure of electronic health records in compliance with data protection laws without impeding the legitimate flow of information. [2] These types of databases can implore any of the access control methods that will be discussed in the upcoming chapter. It is the job of this database that takes its name and ethics from The Hippocratic Oath which guides healthcare professionals for hundreds of years – to ensure privacy and security by which the founding principles dictate. The principles of Purpose Specification, Consent, Limited Collection, Limited Use, Limited Disclosure, Limited Retention, Accuracy, Safety, Openness, and Compliance.

EHR Software 2.5

EHR systems currently are in a median phase of development, with small companies trying to grab a niche, while larger corporations some with healthcare background and some not are vying for an untapped business model. In many cases larger corporations will simply acquire start-up or small ventures in an attempt to enter the EHR arena. This seems to be the private sector model and in creating an EHR there has not been an official certified system or implementation of an EHR. It is the hope or near future of Electronic Health Records that the U.S. Government will initiate an overall plan and certification process for the EHR. Currently the most successful execution of an EHR system is the U.S. Department of Veterans Affairs (VA) system known as VistA. Australia, the United Kingdom, and Canada have also begun EHR system development in the goal of overall consolidation of patient information.

Evaluation 2.6

The foremost problem with creating an EHR it seems is everyone has his or her own interpretation of what the best EHR system should be. Over time the need for information technology and healthcare to merge was realized, but it still lacks the research, development, and allure that some other disciplines such as business, science, and engineering that have drawn interest from Computer Science for the development of systems. A step forward will be the adoption by governments of a centralized and certified model to help in the overall consolidation of patient data. In order for this to move forward healthcare leaders, politicians, and information technology leaders must work together to obtain a common ground and sturdy foundation to build upward. More importantly, there must be a buy in from practicing physicians. They are usually the first level of healthcare most patients seek out. There are many reasons why most physicians are reluctant to implement an EHR. The top reason being financial, this is asking the doctor to purchase new hardware and software that will cost additional money time, and resources that some doctors may not have. The second most issue is education. Older physicians are unaware of the EHR and are not technology savvy. They fear a change of routine and the feeling of not being in the know enough about something in their own field. Overall, most are not adept to change. This is a substantial change in the way they do business and it does not sit well. Many distinctions will be made, but requirements, standards, and research will all have to be generated as well as a bridging of existing EHR technologies for a national solution. This may take ample amounts of time and resources, but the overall benefits such as improved patient care and quality, proficient tracking of patients and expenditures would far exceed the obstacles in creating a centralized Electronic Health Record.

CHAPTER 3

Access Control 3.1

Access Control has been described as “The prevention of unauthorized use of a resource, including the prevention of use of a resource in an unauthorized manner” [31]. Access Control can allow not only a person admittance to secure data, but also the type of access granted as well. Access control comes in three major favors – Administrative, Logical, and Physical. Administrative deals with policies and procedures intended to administer rules for security. An illustration of this principle would be training for security, monitoring transactions and usage limits, and internal polices such has hiring practices or company rules. Logical or technical control is the use of software or hardware to impede access to objects. An example would be user ID and authentication, encryption or coding, and isolated network architecture. Physical control is as it sounds, reducing access to hardware by using physical objects such as secured doors, walls, or electronic codes and combination locks. We will be focusing on the logical type in this paper. Access control models assume that the users are authorized to access the information system. [14] After permission is granted the access control method will outline what data each permitted user can access. Access control usually begins with authentication, which provides the identity of the user or client systems attempting to gain entry. The second phase is granting the permitted user access to specific data based on rules and the clearance level attached to the person or group created for usage. Access Control can be implemented in a variety of fields in Computer Science from healthcare, financial industry, universities, and government. The need for secure information and data has driven the appeal to use Access Control. Access Control has three major areas – Subject, which is the entity that is given permission to the data an example would be a user, groups of users, or a process request from a user. The second area of Access

Control is Objects, which is the actual items being protected an example would be records, files, folders, tables or cells. The third area of Access Control is Access Rights, which is the way that the level of control is established an example would be read, write, delete, and query. All three areas play a vital role in the successful implementation of the security. The architectures and schemas for Access Control define the rule-based logic behind the definition of entry for the user. Access Control can be a basic concept or it can become very detailed down to the smallest granularity. Most of the models developed so far have been designed to satisfy healthcare security requirements in a controlled environment, such as the Electronic Medical Record database maintained within a hospital. [4] There are many different types of Access Control and it can be applied in many different methods as will be explained in the next section.

Types of Access Control 3.2

In the world of security models for Access Control there are typically types of traditional policies and some new models and types. Access control mechanisms can be grouped into four main classes: discretionary, mandatory, role-based and attribute based. [1] An access control scheme outlines rubrics for users retrieving data, profiles or information. A Subject will ask for permission to a component called an object. The object could be any entity that encompasses information or statistics and resources a subject needs to complete a scheduled job or requirement. The access control scheme that calls for a specific task will ask to be the burden of responsibility for assessing the subject’s demand for access to a certain object and produce a solution that will make a difference. We will discuss a few of these types in this paper.

Discretionary Access Control (DAC) 3.2.1 The first of these polices discussed in this section is Discretionary Access Control (DAC). DAC is based on the characteristics of users and/or links to defined groups. Right of entry is normally based on the authentication approved to the user based on the classifications they presented at the time of validation and the identity of the control to elect whether to authorize or decline an request for access. The object's owner outlines the subject that can pass the object, so all use of the object is at the choice of the administrator of the object. This access control strategy is normally less safe than mandatory access control, but is a very common design in business operating systems. Even though it manages to be less protected, it is simpler to employ and more adaptable for surroundings that do not require rigorous object security. Most objects have authorizations, or privileges that stipulate which users and groups can retrieve the object. These could include user name and password, cookies, or labels. For most common DAC versions, the administrator of data or resources is able to manipulate the authorizations at their choice. DAC could be utilized in two ways, as a centralized or distributed security model. In the centralized version, a single or group of authorization entities grants access to applications, data, or hardware devices. All inquiries for change requests need to be enacted by this group or position.

In a distributed version the most qualified persons, such as team leads, systems analysts, or whoever is the authority figure can give access to the resources to for other users. The benefit of distributed model is that delays can be avoided since the administration of accounts is dispersed [35] Using multiple managers with administrator rites for this access allows better flexibility and fewer logjams. DAC has the disadvantage of the administrators not being able to internally administer permissions on data stored on a web server. There is also the concern that access for end-users with job related tasks that are similar could be compromised or unorganized. A DAC access control method often demonstrates one or more of the following characteristics: Administrators can transfer rites of information to other users. Administrators can choose the kind of permissions given to other users. Multiple authorization errors to access the same resource or object would restrict the user's login. Users who are not authorized would not be able to view items such as file size, name, or type. Access to data is established on permission to access control lists based on user credentials. The next type of policy is Mandatory Access Control (MAC).

Mandatory Access Control (MAC) 3.2.2 Mandatory Access Control guarantees that the implementation of a security regulation is not based on voluntary application user fulfillment. When using MAC control a frequent installation is having it be rule-based. MAC protects data by assigning security labels on data and evaluating the results to the level of access a user is granted. For the most part MAC access control is more secure than DAC but there are concessions in performance and feasibility to users. MAC assigns a security label to all data, and a security level to each user to ensure that all users will only have permission to information for which they have access. MAC security requirements are more often than not hard coded into the operating system or application. This can eliminate the possibility of user error or modification. This can also cause a problem because it will need to be monitored for future changes and is not very dynamic. A MAC access control method often demonstrates one or more of the following characteristics: Only administrators, not information providers, make changes to a user’s security label. All information is granted a security level that imitates its code value. Users can access a lower classification than the one they are permitted, so a "secret" user can see an unclassified document. All users can modify a higher classification, so a "secret" user can change data to a top secret object. All users are granted read/write privileges to items only of the same category, so a "secret" user can only read/write to a secret record. Access is granted or not to objects based on the time of day, week, or year based on the labeling on the data and the user's authorizations. This can be instituted by policies or procedures within an organization.

Role-Based Access Control (RBAC) 3.2.3 The third type of access control policy is Role-Based Access Control (RBAC). Role-Based Access Control access to data is based on a user's roles and position within an organization or company. Role-based access control (RBAC) is a framework for controlling user access to resources based on roles and it can significantly reduce the cost of access control policy administration and is increasingly widely used in large organizations. [22] The procedure of designating roles is usually based on evaluating the basic needs and configuration of an organization and is usually coupled with a security policy. An example for the Electronic Health Record would be the roles of users that may include nurse, doctor, patient, and administrative clerk. These users necessitate separate levels of access in order to carry out their duties. RBAC structure should provide application security administrators with the knowledge to detect who can perform what tasks, when, from what area, and in what order of sequence. RBAC has five major elements. The user or the person trying to gain access, role that configures what permissions the user has, the permissions themselves that grant access to objects, operations such as read, write, update and delete, and finally the objects which are the data or information that are needed to be accessed. The following attributes can be found for a RBAC model:

Roles are given based on organizational arrangement with importance on the organizational security policy. Each role is given profiles that consist of all authorized commands, transactions, and object access. Roles are created with a separation of tasks in mind so that no role should overlap another role. Roles are overseen centrally by a security officer or project manager. RBAC can use role engineering as a method for scalability. It is logical in design because it is based on organizations positions within its hierarchy. This will allow for easy administration and adaptation when the need arises. RBAC is definitely a preferred method of access control and will be the type implemented in this study for the Electronic Health Record.

Bell-LaPadula Model 3.2.4 Another type of access control methodology is The Bell-LaPadula model. The Bell-LaPadula model is a state machine model that employs access control lists and security labels to implement object security. [34] The model uses two rudimentary properties to assess permission needs. A simple security directive which does not allow read up privileges so a user that has an assigned security level cannot see information from a higher level. The second is the asterisk (*) or sometimes known as the ‘star’ property. This prevents a user with a given security level from accessing a lower level or also known as the no write down policy.

The Strawman Design 3.2.5 The final type of access control discussed in this chapter is the Strawman Design. The Strawman Design incorporates the use of a data store with different data options such as collection, insertion, preprocessing, queries, and retention. In this design, we use purpose as the central concept around which we build privacy protection. [3] The following figure represents the Strawman Architecture.

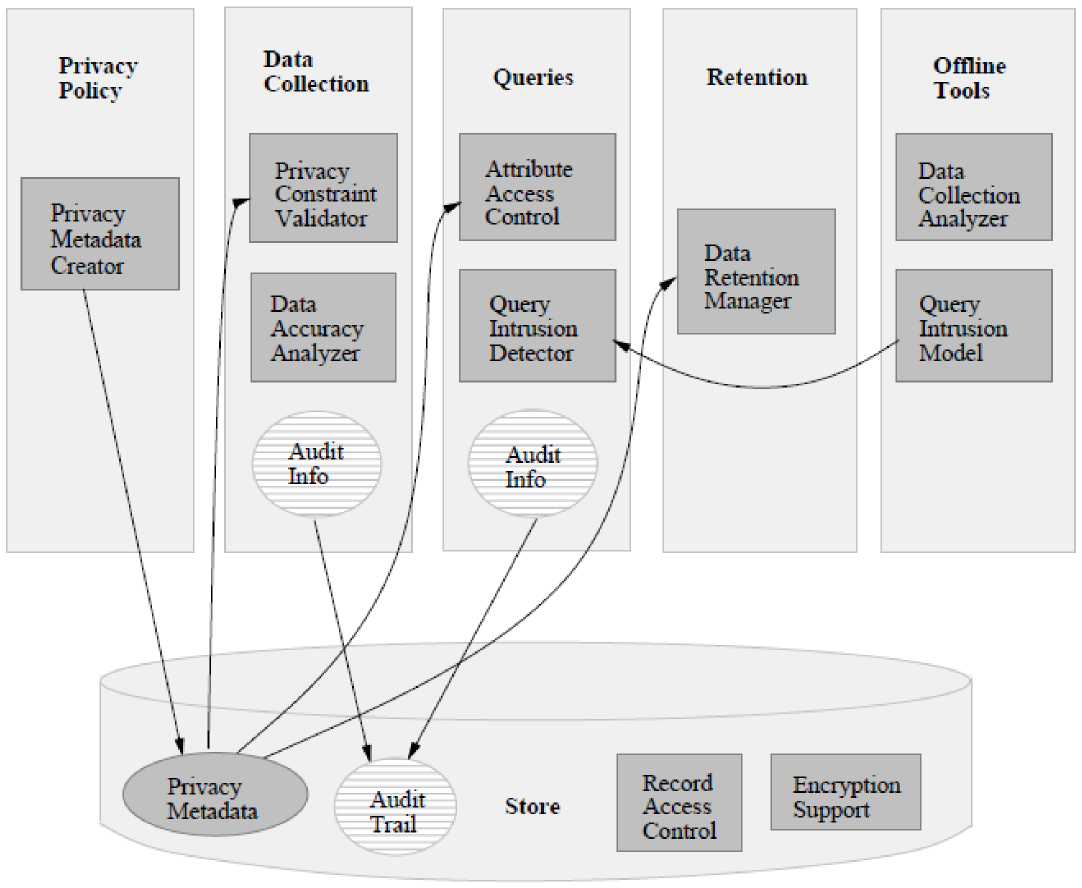


Figure 3.1: Strawman Design Architecture [3]

CHAPTER 4

Access Control Configuration 4.1

This chapter will describe the methods and tools used to implement the access control for the Electronic Health Record example. There are similar technologies that exist on both the Linux®, UNIX®, and Windows® platforms. Topics covered include security labeling, row-level security, cell-level security, and encryption. The database of choice will be Microsoft SQL Server 2012 and the programming language will be Visual Basic.NET using Visual Studios 2010, and MS TSQL

Security Labeling 4.2

A security label can be a piece of data that represents the privacy of an object. It is a sequence containing markings from one or more categories. Users (subjects) have permissions described with the same markings. Security labelling is one of the mechanisms that can be used in the provision of information security and it supports the correct handling of data within and between systems, according to the sensitivity of the data. [21] Security labels allow for the management and enforcement of inter-enterprise privacy policies. [28] Security Labels and Security Clearance provides a mechanism for controlling access to information that works well for large numbers of users. [15] Each subject has a label of their own. The subject's label is compared against the label on the object to determine access to that object. For example, the following table fragment has rows annotated with security labels in the Classification column.

Table 4.1: Security Label example

Name	**Classification**
Sergio Senese	RESTRICTED
Linda Napoli	CONFIDENTIAL
Steve Duignan	PUBLIC

Applications containing this data could have operator accounts as follows:

Table 4.2: Example operator accounts

Operator	**Access**
Anthony	CONFIDENTIAL
Spike	PUBLIC
John	RESTRICTED

Each operator's security label defines which rows in a table they can retrieve. If Anthony delivered a TSQL `SELECT * FROM <name of table>` against this table, he should get the subsequent results.

Table 4.3: Query Result

Operator	**Access**
Anthony	CONFIDENTIAL
Spike	PUBLIC

Access control structures can get more complex than the previous example. There can be additional access benchmarks conveyed in a security label. An example would be, in addition to the required classification, access to data may necessitate additional markings. In some use cases, security labels can include several markings from different categories, and the number of possible label combinations can be infinite.

Terminology 4.2.1 We will now explain the terminology used to accurately identify labels, and the comparison of labels. A *label* is a string that describes either the sensitivity of an object, or the permissions of a subject. [29] The label is an assembly of *markings*. A marking defines a specific authorization. The markings in a label come from one or more *categories*. [29] A subject can access an object if the subject label *dominates* the object label. [29] Given two labels, A and B, label A is said to *dominate* label B if every category present in label B is satisfied by markings on label A. [29]. The *categories* can have the following properties. Domain is the possible markings in the category. Hierarchical which is a yes or no choice determines if the category is hierarchical. Hierarchical categories have a prioritization among values. This ranking determines access. A marking can satisfy any marking on the same or below its level in the hierarchy. Nonhierarchical categories have no listing among values. A marking is either present or not represented. The next step is to create a labeling policy inside the database. This can be done by TSQL code and commands or by using the Label Security Toolkit Version 1.5. [24] The following figure show how the labeling policy will look like:

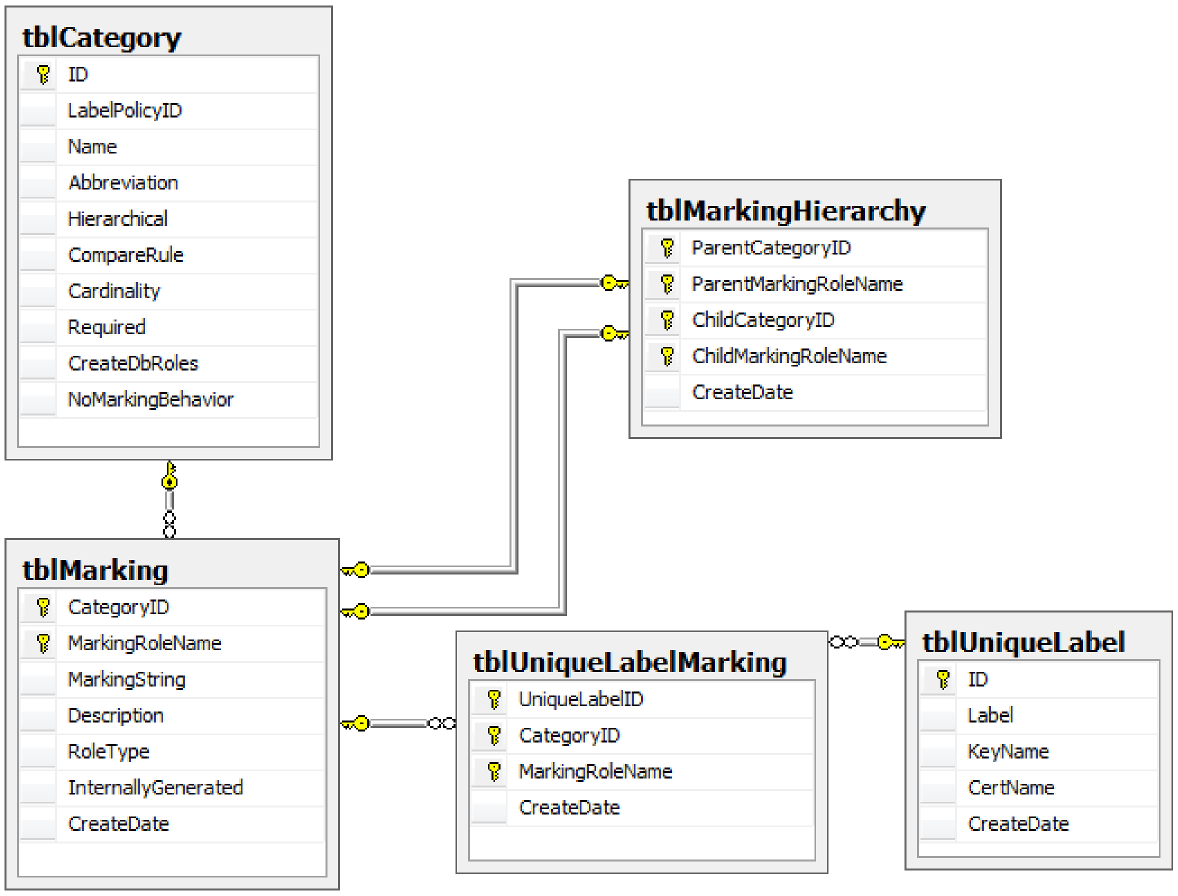


Figure 4.1: Example of a Label Policy mapped in a database [29]

Example Classification Markings by Code

INSERT tblMarking (CategoryID, MarkingRoleName, MarkingString)

VALUES (2, 'rxMedications', 'Medications');

Row Level Security 4.2.2 Having outlined the configuration of labels in the policy, something needs to tie in the SQL Server security model. For each marking in the categories defined, a corresponding database role is created. Database roles have to adhere to the following rules. For nonhierarchical categories that have an ‘Any’ or ‘All’ comparison rule, create a role for each possible value. The name of the role must match the value in the table tblMarking with the MarkingRoleName column for that marking. For hierarchical categories, the process is the same but, the roles must be nested to model the hierarchy. Each role created, must add the parent role that is defined in the tblMarkingHierarchy table as a member. This nesting ensures, that users with a specific clearance can access data at the level and below. Roles allow users to be granted permissions that exactly match the security labels to which they should have access. It is the task of the applications or database administrator (DBA) to correctly maintain role memberships for the operators of the system. The next step is to define a view that will be used to display the data and prevent users from actually accessing the base table. The created view over the base table will give the illusion in the eyes of the end user in the application that they are truly in the base table. Here is what the view definition should look like.

```
CREATE VIEW [dbo].[PatientInformation]

AS
       SELECT
              tblPatientInformation.ID,
              tblPatientInformation.PatientID,
              tblPatientInformation.FirstName,
              tblPatientInformation.LastName,
              tblPatientInformation.PatientAddress,
              tblPatientInformation.Phone,
              tblPatientInformation.Email,
              tblPatientInformation.SS#Number,
              tblPatientInformation.EncryptedSS#Number,
              vwVisibleLabels.Label

       FROM tblPatientInformation WITH (READCOMMITTED) JOIN vwVisibleLabels
              ON (tblPatientInformation.RLS_ID = vwVisibleLabels.ID)
```

By joining the security label identifier in the base table against the Visible Labels view, the label authority logic controls access to rows in the base table. The READCOMMITTED locking applied to the base table prevents bad reads against the base table. This inhibits a reader from viewing rows within another user’s pending transaction which may have been updated with sensitive data, but the security label identifier(s) has not yet been altered. The locking in the view supersedes the reader’s transaction isolation level or any unambiguous locking used when querying the view. For write operations, INSTEAD OF triggers for insert, update, and delete may be added to handle more complex requirements.

Cell-Level Security 4.2.3 Data may be required to be kept at a more concentrated level of detail than the entire row. Most of a row could be visible to one set of users, while certain additional confidential cells might require additional permissions to view. Controlling the visibility of the data based on user permissions would be the ideal solution. Preferably, the database system could simply show data from labeled cells, or not, based on the characteristics of the connected user. Starting with SQL Server 2005, Microsoft introduced encryption capabilities in the database system itself. These can be used to encrypt and decrypt any random data, using a certificate and key association that is internally managed by the database system. Linux® and UNIX® systems employ similar technology for row and cell level security. Row permissions and column masks are two new database concepts that are introduced to address the shortcomings of traditional row and column access control methods. [33] Database-level encryption allows enterprises to secure data as it is written to and read from a database. This type of deployment is typically done at the column level within a database table and, if coupled with database security and access controls, can prevent theft of critical data. [23] There is no need create or add in certificates or keys from an outside source. The basic goals of the design are support for the random labeling of cells of data. Dynamic assessment of the user's label, to show only the pertinent cells. Acceptable performance and impact on the database when there is high volume. SQL Server offers internal functions to effortlessly encrypt and decrypt data using a certificate, asymmetric key, or symmetric key. It administers all of these in an internal certificate store. The store uses an encryption hierarchy that secures certificates and keys at one level with the level above it in the hierarchy. Optionally, the Extensible Key Management feature (EKM), added in SQL Server 2008, and allows parts of the encryption hierarchy to reside in external key management devices. [29]

The swiftest mode of encryption supported by the internal API is symmetric key encryption. This mode is appropriate for handling large capacities of data. SQL Server supports several symmetric key encryption algorithms. Within the range of a database connection, SQL Server can maintain multiple open symmetric keys. When a portion of data is decrypted, there is no need to stipulate which symmetric key to use. Instead, the mechanism matches the encrypted byte stream to an open symmetric key, if the correct key has been decrypted and is open. This key is then used to execute decryption and return the data. If the correct key is *not* open, the system returns NULL.

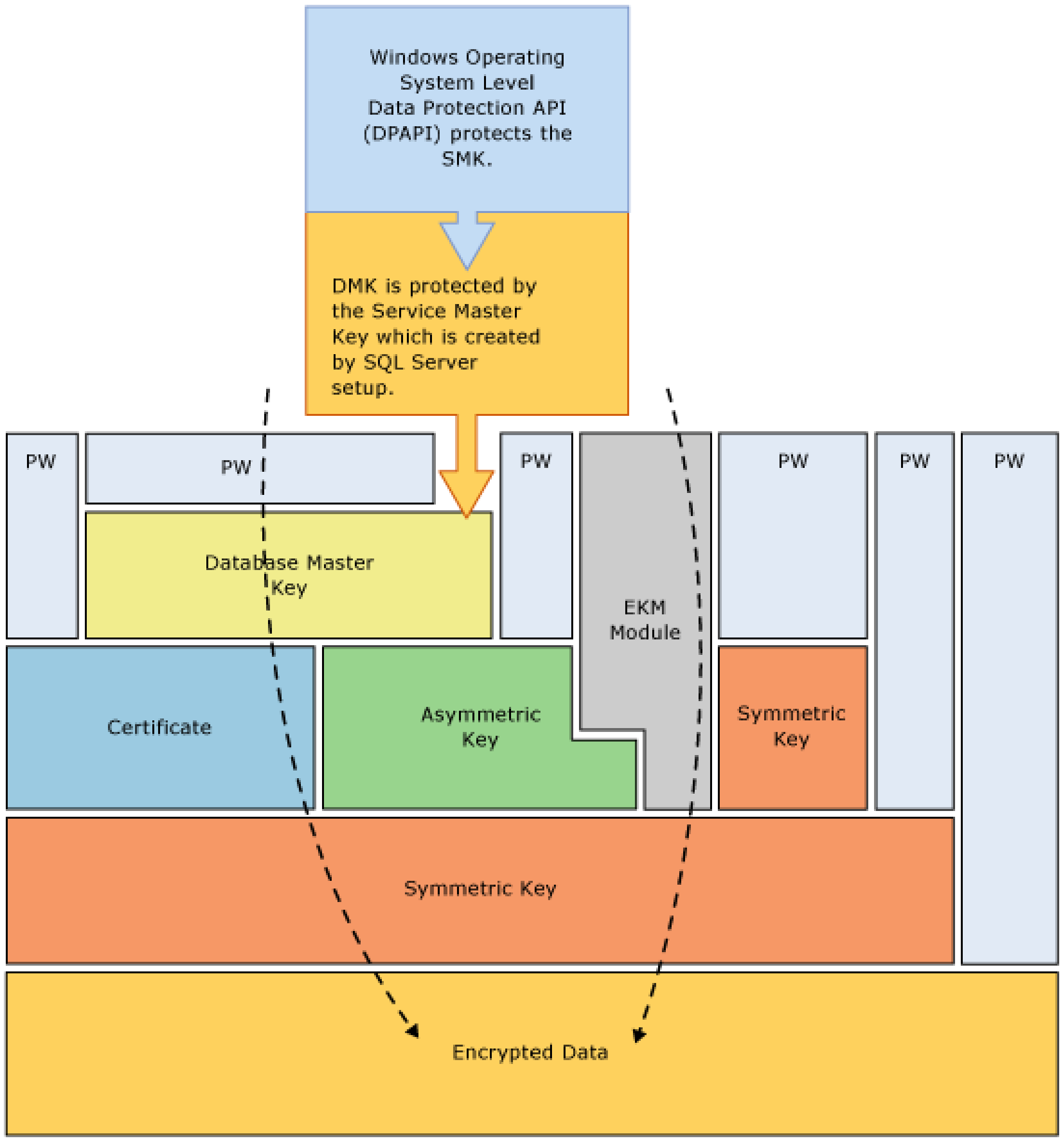

Windows Operating System Level Data Protection API (DPAPI) protects the SMK.
DMK is protected by the Service Master Key which is created by SQL Server setup.
PW
PW
PW
PW
PW
PW
Database Master Key
EKM Module
Certificate
Asymmetric Key
Symmetric Key
Symmetric Key
Encrypted Data

Figure 4.2: Encryption hierarchy [29]

The capability of a key to be "open" is contingent upon being the permission GRANTs on the key. We can use the following principles to enact cell level encryption. Create a symmetric key for each unique label that is used to mark data in the database. Encrypt data in labeled cells with the corresponding key. Control access to keys in such a way that exactly the keys that map to labels to which the user has access can be opened. Provide a modest way to have all these keys opened when the connection is established.

The encryption is created by using the EncryptByKey function and decrypted by the DecryptByKey function. These are helper functions that are included in the label security schema. Combining cell-level security with the row-level security model, we have a design shown in Figure 4.3 as a design for outfitting a table and system with both types of fine-grained access control.

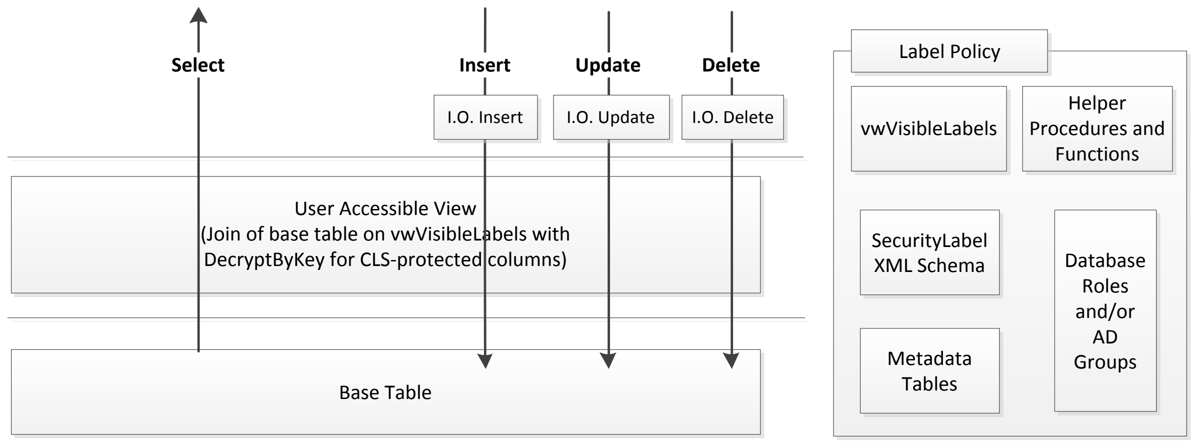


Figure 4.3: Fine grained access control implementation [24]

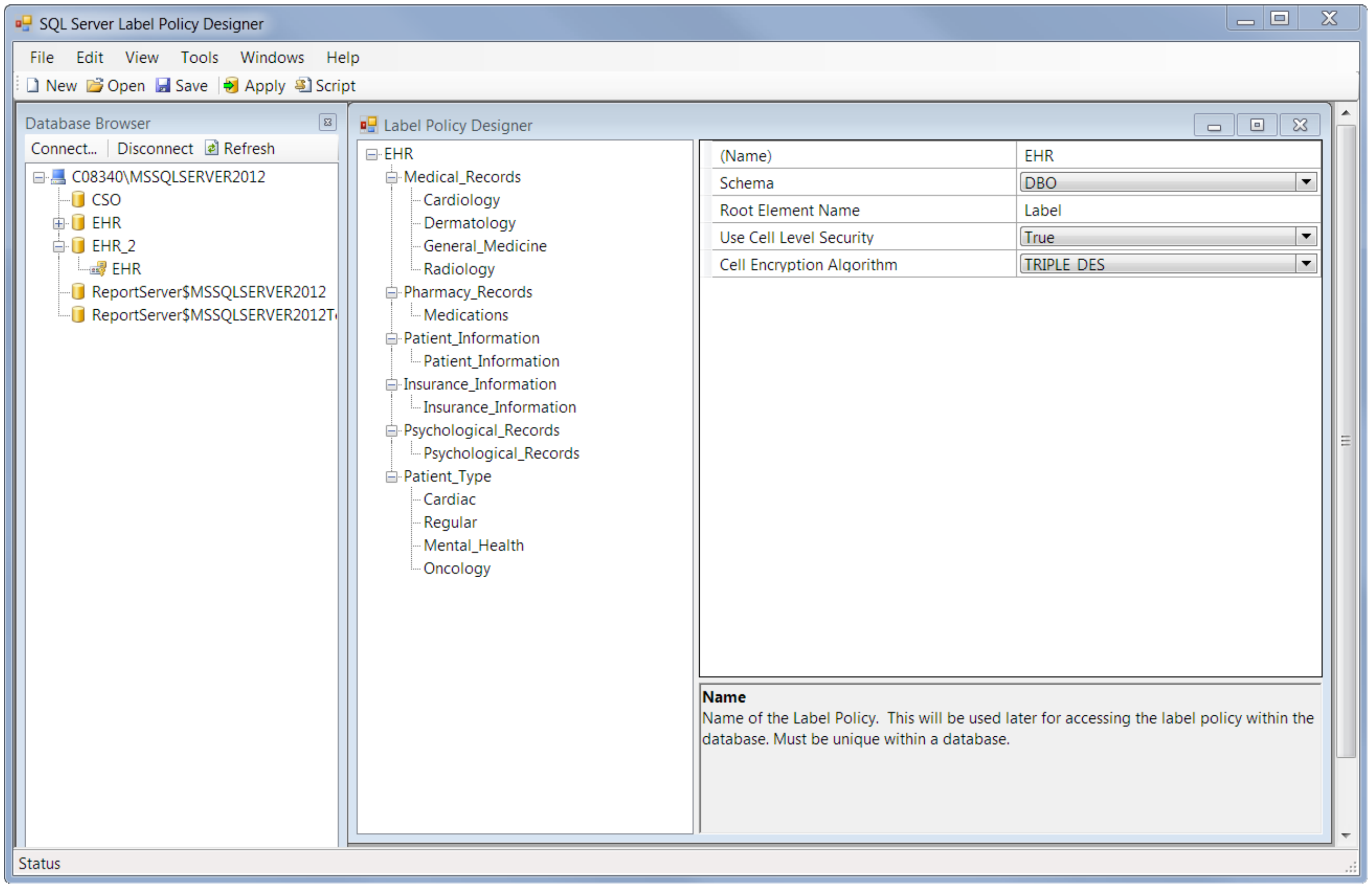
CHAPTER 5

SQL Server Label Security Toolkit 5.1

This chapter will introduce the Label Security Toolkit and its features and how it is used to implement row-level and cell-level security in SQL Server. The Label Security Toolkit provides tools and techniques for using Microsoft® SQL Server (versions 2005 through 2012) to implement row-level security (RLS) and cell-level security (CLS) based on security labels. The Label Security Toolkit provides an automated option to manually creating security labels, encryption, and policies. The Toolkit provides a graphical interface for representing the categories and markings used in a label policy and database.

Label Policy Designer Application 5.2

The SQL Server Label Policy Designer can be implemented to outline a label policy and then apply the label policy by creating objects and code in a database. The label policy influences both the assembly of labels in the system and the markings that are allowed for each of these portions of a label. To explain the overall view of a label policy, it is a group of metadata tables, functions and stored procedures, database roles, and other objects inside of the database. These aid the swift development of an application-specific data archetype that uses label-based security. Through the use of the label policy, tables in the data model can be effortlessly equipped for row- and cell-level access control. After completion, the label policy can be applied immediately to a target database, or transformed into a SQL delivery script for later use. Another possibility for the label policy definition is it could be transferred to XML for later revision and reuse in the tool.

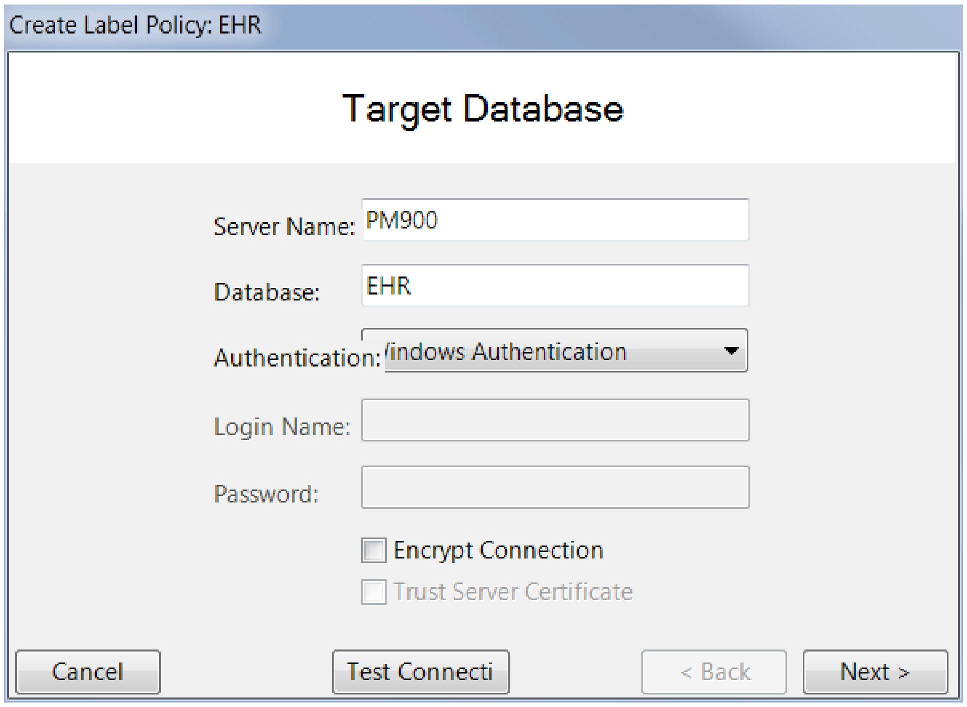


(Name)	EHR
Schema	DBO
Root Element Name	Label
Use Cell Level Security	True
Cell Encryption Algorithm	TRIPLE_DES

Name
Name of the Label Policy. This will be used later for accessing the label policy within the database. Must be unique within a database.

Figure 5.1: Label Policy Designer interface

The Label Security Toolkit will not offer tools for managing the different permissions of system users. User authorizations are defined by association in platform security groups (SQL Server database roles and/or Microsoft® Windows® groups), typical methods for administering group roles should be used. There is always the option of utilizing custom functionality for appointing user permissions. The Label Policy Designer needs full database owner rights on the home database. Ideally this tool should be used only by administrators with authorization to data stored inside these databases. Another option is the tool can be used to create the label policy(s) before classified data is inserted to the database, and then the database administrator's privileges can be diminished. When performing routine administrative work such as taking backups or managing user authorizations, reduced privileges that do not require full database entry can be given. Applying the label policy to the database is relatively simple, most actions in the Toolkit are performed by wizards. This process applies the label policy directly on the database.


Create Label Policy: EHR

Target Database

Server Name: PM900

Database: EHR

Authentication: /indows Authentication

Login Name:

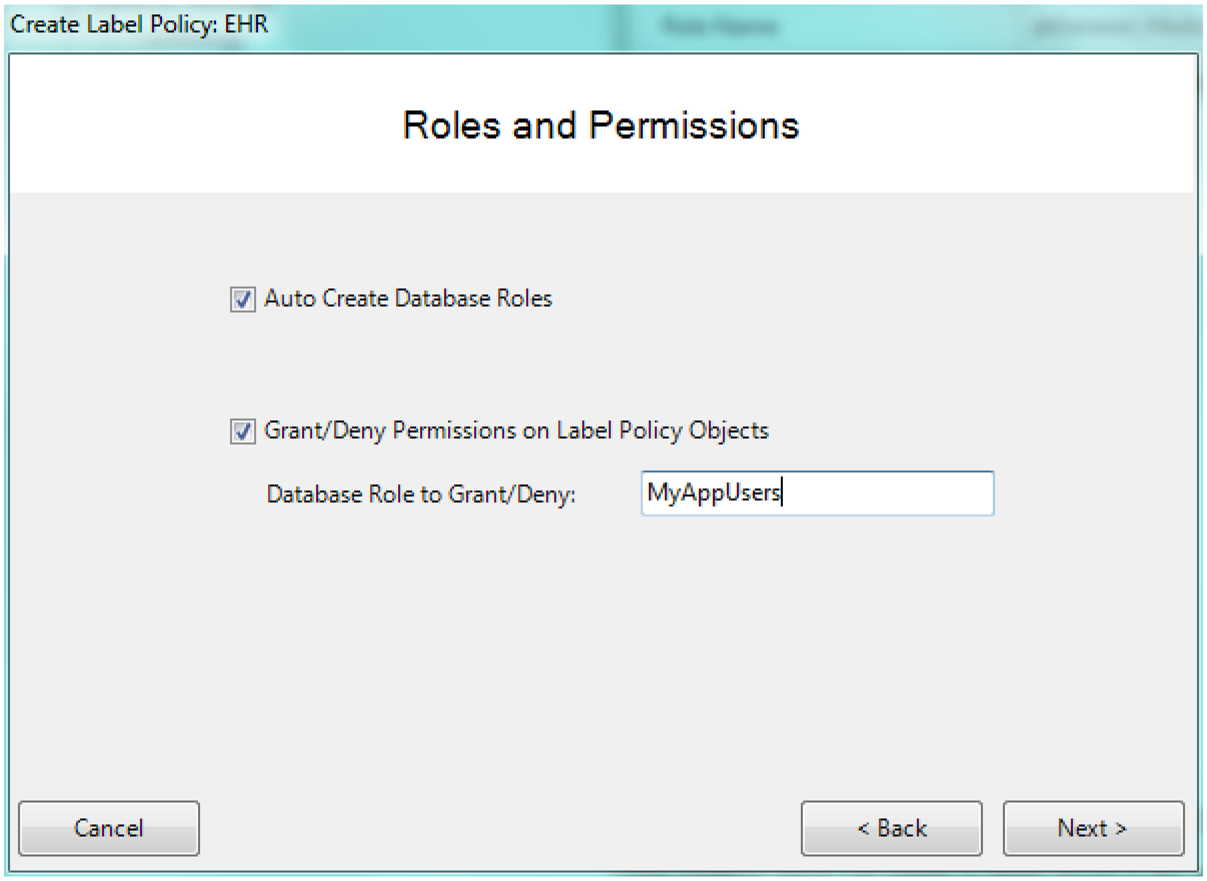
Password:

☐ Encrypt Connection

☐ Trust Server Certificate

Figure 5.2: Applying Label Policy to database

Roles and Permissions are the next item to be created. There is a feature to Auto Create Database Roles, instead of manually configuring them one by one in the database. In order to for this feature to be enacted the corresponding checkbox must be enabled. The Grant/Deny Permissions on Label Policy Objects check box enables or disables authorization grants on the label policy. Some objects should be accessible to over-all user accounts in the database; others should not be accessible. If this check box is checked, the wizard will include grant and deny statements as a default configuration.


Create Label Policy: EHR

Roles and Permissions

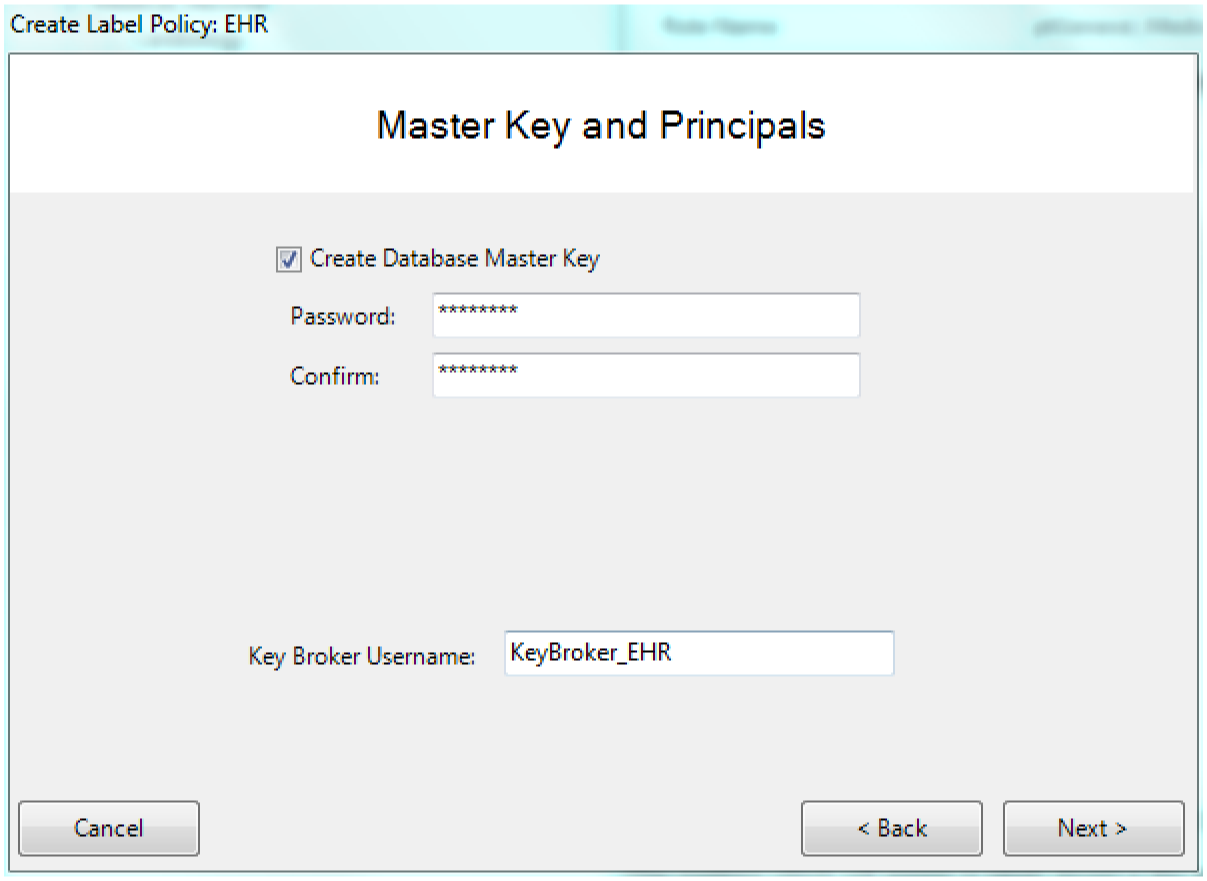
☑ Auto Create Database Roles

☑ Grant/Deny Permissions on Label Policy Objects

Database Role to Grant/Deny: MyAppUsers

Figure 5.3: Creating Roles and Permissions

If used, you must provide a name for the database role. This is the SQL Server fundamental to which authorizations on label policy objects will be granted or denied. Usually this role will include all permitted users of the application. You can use or create a ‘Public’ role. If you choose not to check this box, be mindful that custom database code that use the label policy to deploy labeled data may display error messages and experience permission issues. If you created a label policy with Use Cell Level Security marked as True, another option screen will appear to create a Master Key. To provide for cell-level security, a database master key must be present in the home database. If this is true, the Create Master Key check box can be unchecked. The second entry calls for the Key Broker Username.



Create Label Policy: EHR

Master Key and Principals

[✓] Create Database Master Key

Password: ********

Confirm: ********

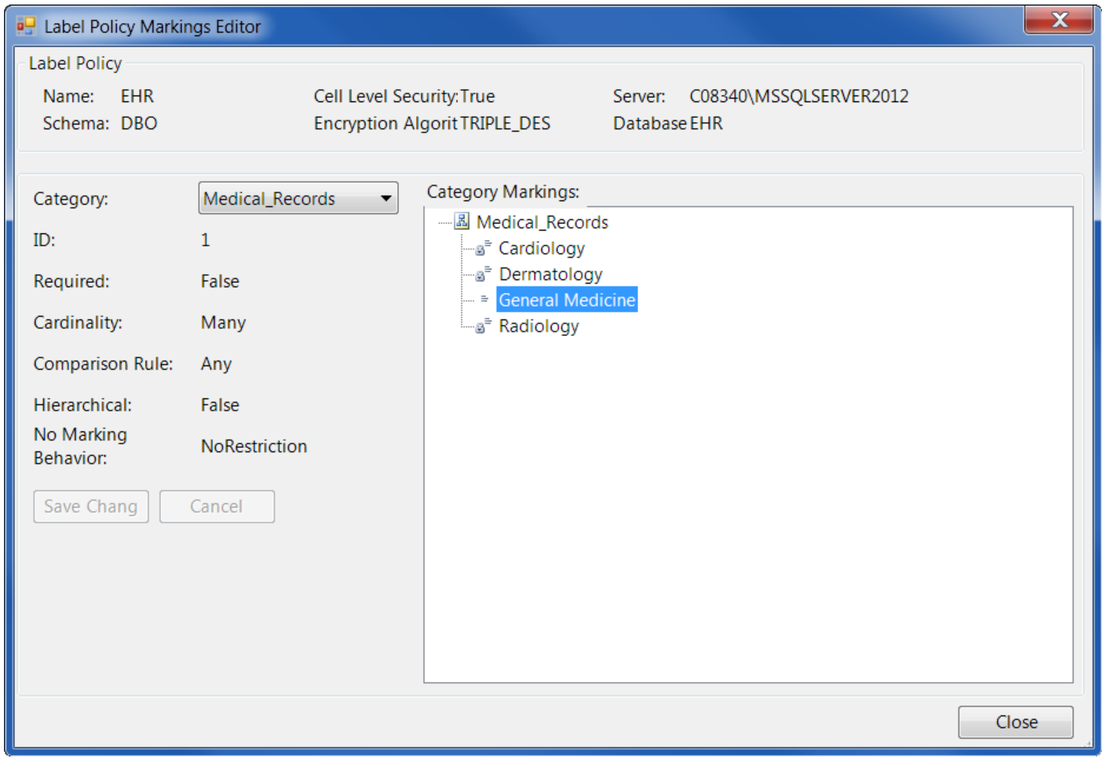
Key Broker Username: KeyBroker_EHR

Figure 5.4: Creating the database Master Key

The Key Broker Username is the name of a database element that will be created to regulate access to symmetric keys in the label policy. The default name can be accepted or renamed if there is a conflict within the database. Cell-level security is based on the encryption features introduced originally in SQL Server 2005. SQL Server provides built-in functions to encrypt or decrypt data using symmetric key algorithms. The keys are handled in an internal certificate and key store, which uses an encryption hierarchy to guard each certificate/key. The Label Security Toolkit develops on this feature set by mapping a symmetric key to each unique security label occurrence used in the database. At the cell level only a particular field value in a particular row of data can be labeled so that only operators with appropriate access can see it. The term "labeled" means that the cell value is encrypted with a key mapped to that label, and that an operator has access to the decryption keys for only those labels that they should be able to see.

Altering Markings in a Present Label Policy 5.2.1

After a label policy has been applied to a database, it may be required to make modifications to the markings in the label policy. A new department might be created within the company. The department requires a new marking to label data that is pertinent to them. Combined work with an outside business may present the need for added markings that are used to control mutual access to data. The Label Policy Designer supports adding and removing markings for an existing label policy. Other changes to a label policy, such as adding or deleting categories, or changing the category attributes are not supported. The reason for this is changing categories can essentially change the meaning of current data. Changing categories in the label policy should be handled more like a data conversion or migration when attempting such action.


Label Policy Markings Editor

Label Policy

Name: EHR
Schema: DBO
Cell Level Security:True
Encryption Algorit TRIPLE_DES
Server: C08340\MSSQLSERVER2012
Database EHR

Category:	Medical_Records
ID:	1
Required:	False
Cardinality:	Many
Comparison Rule:	Any
Hierarchical:	False
No Marking Behavior:	NoRestriction

Category Markings:
- Medical_Records
  - Cardiology
  - Dermatology
  - General Medicine
  - Radiology

Figure 5.5: Editing a Label Policy

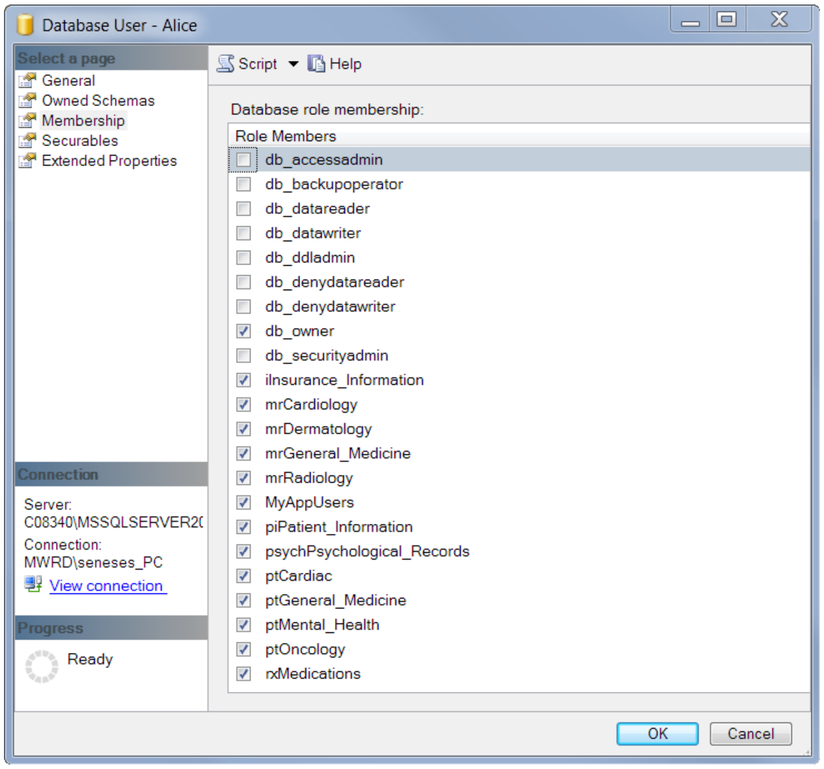
CHAPTER 6

Example Application - EHRPro™ 6.1

This chapter will provide a live prototype to demonstrate the one of the various possibilities on how access control and encryption can be used to create and secure an Electronic Health Record. The database of choice will be Microsoft SQL Server 2012 and the programming language will be Visual Basic.NET using Visual Studios 2010, and MS TSQL.

EHRPro™ Database 6.2

The EHRPro™ Application back-end is a SQL Server database with roles, tables, views, stored procedures and triggers that encompass many different uses. First a user must be created in the database and roles assigned. This would be done by the database administrator or a front-end application could be developed to assign roles. There are many roles for this example and numerous Label Security and encryption scenarios.



Server:
C08340\MSSQLSERVER2(

Connection:
MWRD\seneses_PC

Database role membership:

	Role Members
☐	db_accessadmin
☐	db_backupoperator
☐	db_datareader
☐	db_datawriter
☐	db_ddladmin
☐	db_denydatareader
☐	db_denydatawriter
✓	db_owner
☐	db_securityadmin
✓	iInsurance_Information
✓	mrCardiology
✓	mrDermatology
✓	mrGeneral_Medicine
✓	mrRadiology
✓	MyAppUsers
✓	piPatient_Information
✓	psychPsychological_Records
✓	ptCardiac
✓	ptGeneral_Medicine
✓	ptMental_Health
✓	ptOncology
✓	rxMedications

Figure 6.1: Database roles

EHRPro™ Application 6.3

Tables and views become created using the label security schema allowing for the columns Label and RLS_ID to be added for use in row level security. The front-end Application requires login for authentication and immediately takes the user to a screen that displays patient information. It is at this point that row and cell-level label security has already taken place.

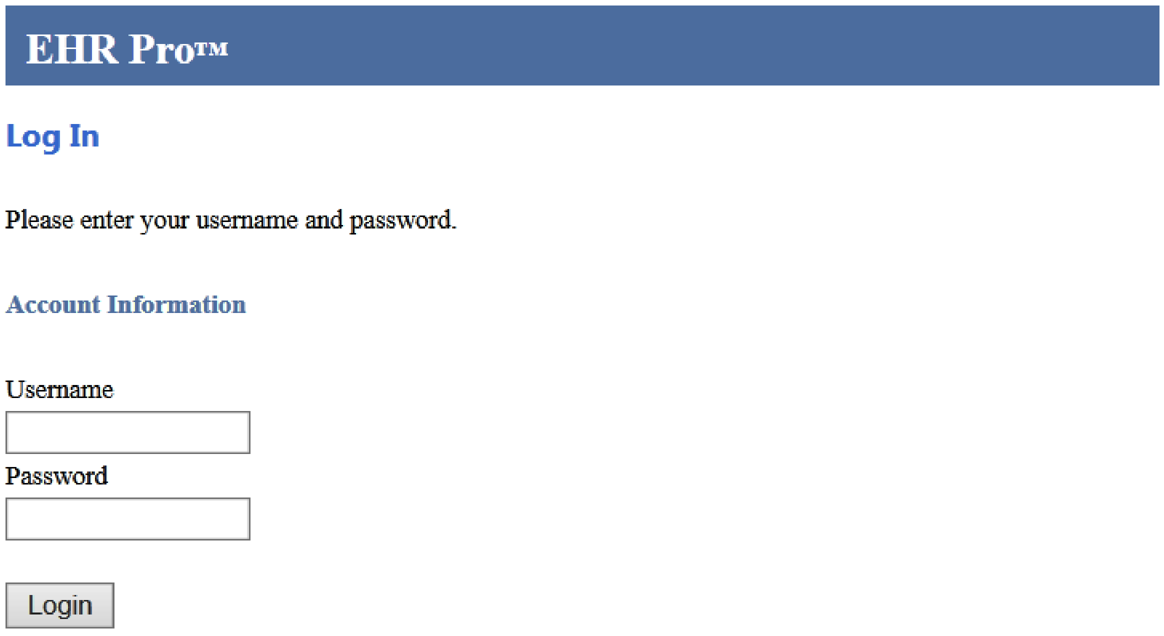
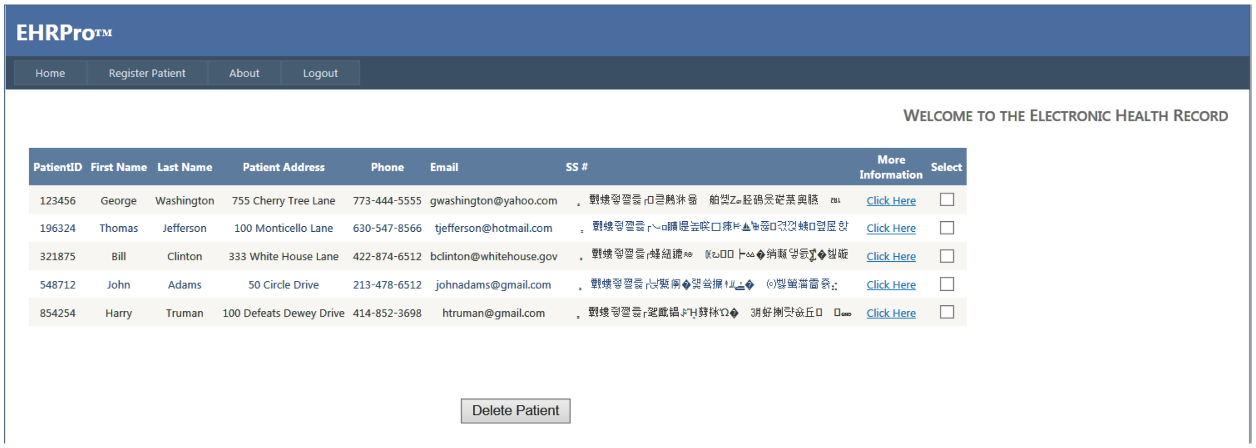


Figure 6.2: Login Screen

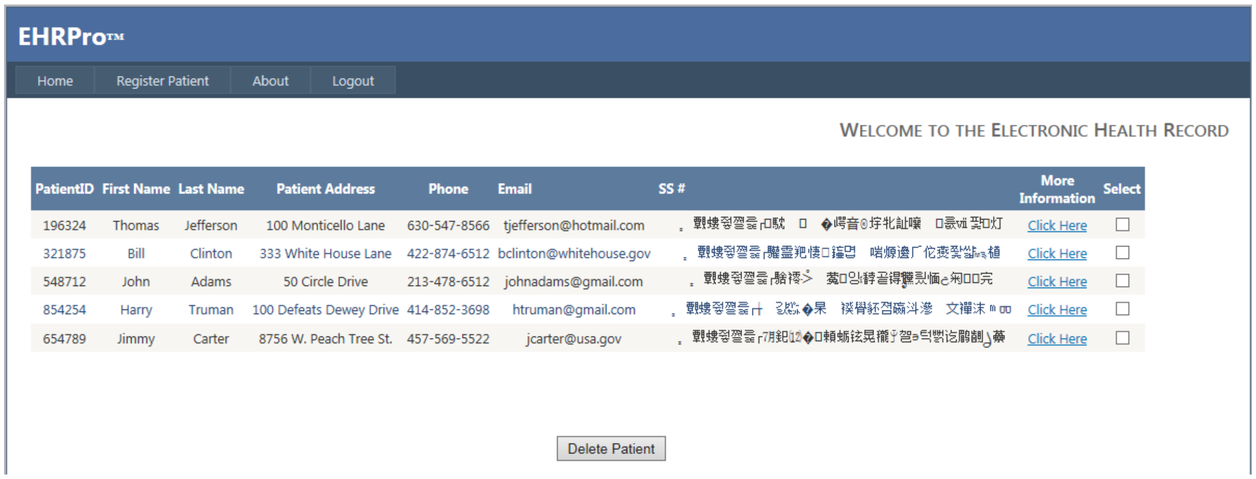


WELCOME TO THE ELECTRONIC HEALTH RECORD

PatientID	First Name	Last Name	Patient Address	Phone	Email	SS #	More Information	Select
123456	George	Washington	755 Cherry Tree Lane	773-444-5555	gwashington@yahoo.com	䕬㵋졍쪯㶦㲨┼ᄃ㛯 畗 숢 舶㵪Zₐ䜑硧㳂䎜萗凄䐡 ᄚᄉ	Click Here	☐
196324	Thomas	Jefferson	100 Monticello Lane	630-547-8566	tjefferson@hotmail.com	䕬㵋졍쪯㶦㲨┌ᆢ□曠㮺㖏咲□痍ᅡ▲ᄓ㦃□걌걶䗝□뎔㞐 핫	Click Here	☐
321875	Bill	Clinton	333 White House Lane	422-874-6512	bclinton@whitehouse.gov	䕬㵋졍쪯㶦㲨┌ᅦ蟮紉鐪ᅠ ᖾᆸ□□ ㅏᅳ�䋥䈐 댕㲀ↁ�쳁碇	Click Here	☐
548712	John	Adams	50 Circle Drive	213-478-6512	johnadams@gmail.com	䕬㵋졍쪯㶦㲨┌㥊楒�볌 㚃摡ᆊ㔖 ㈀뽰黴 㗾ᅲ솖	Click Here	☐
854254	Harry	Truman	100 Defeats Dewey Drive	414-852-3698	htruman@gmail.com	䕬㵋졍쪯㶦㲨┌㟲㽧铝ᄼᄇ譁栤Ω� 㺿蚁拺ᅻ숅丘□ □ᄆ	Click Here	☐

Figure 6.3: Patient List

Users will only be able to view patients if they are given access to the ‘piPatient_Information’ database role. Access is also given based on what type of patient a patient is. An example would be a Cardiac patient with the role ‘ptCardiac’. Access to these roles are granted at the database level and the information displayed is from a view that is created to prevent direct access to the main table and is wrapped in the security label. An Example is user Alice logs in and she can view every patient available. She has access to all roles for patient information and patient types and the total number of patients is five.

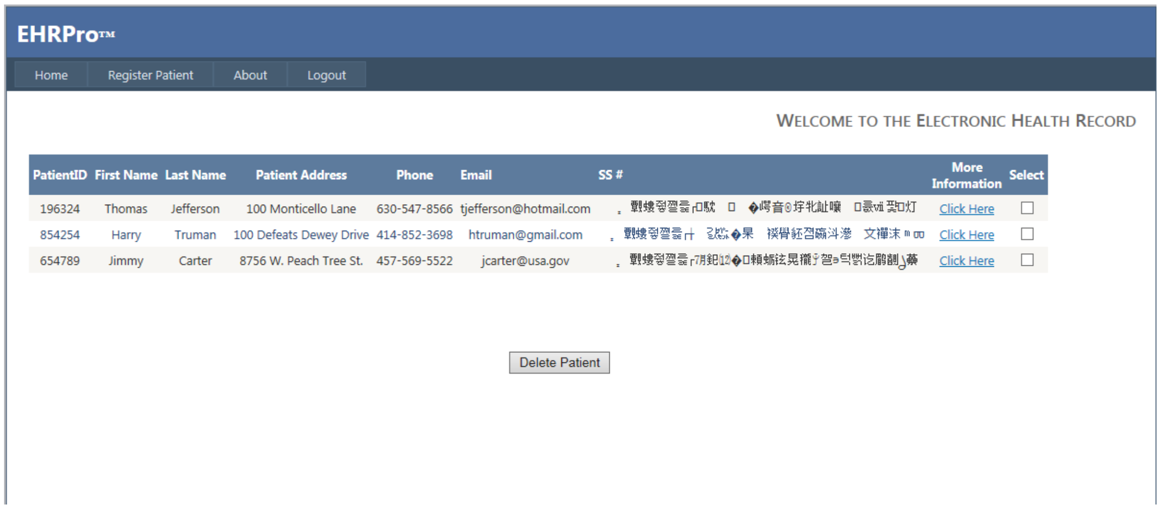


WELCOME TO THE ELECTRONIC HEALTH RECORD

PatientID	First Name	Last Name	Patient Address	Phone	Email	SS #	More Information	Select
196324	Thomas	Jefferson	100 Monticello Lane	630-547-8566	tjefferson@hotmail.com	䕬㵋졍쪯㶦㲨┌□馱 □ �噚音ⓢ垺北訨㬹 □㲀vii 㓞□灯	Click Here	☐
321875	Bill	Clinton	333 White House Lane	422-874-6512	bclinton@whitehouse.gov	䕬㵋졍쪯㶦㲨┌䯃靈羓悽□鎑 唶㵋邉厂佗爽㽸ᅶ稙	Click Here	☐
548712	John	Adams	50 Circle Drive	213-478-6512	johnadams@gmail.com	䕬㵋졍쪯㶦㲨┌骼䅇ᄼ 蔥□잉铐긑㝵鷰㴗悀ₑ㹌□□完	Click Here	☐
854254	Harry	Truman	100 Defeats Dewey Drive	414-852-3698	htruman@gmail.com	䕬㵋졍쪯㶦㲨┼ 㲳㦃�果 㼍㝩䥄㔽礪斗滲 文禪沫ᄆᄆ	Click Here	☐
654789	Jimmy	Carter	8756 W. Peach Tree St.	457-569-5522	jcarter@usa.gov	䕬㵋졍쪯㶦㲨┌ᄀ5鈀⑫�□頼蝂铉晃㰾ᅮ㲿ᄆ둑㗺迄䴄㻌ᆺ蕂	Click Here	☐

Figure 6.4: Patient List for user Alice

Now if another user like Joe logs into the application, he has access to only Regular patient roles. Joe will have access to only three patients. This is an example of the row level security provided by the application.

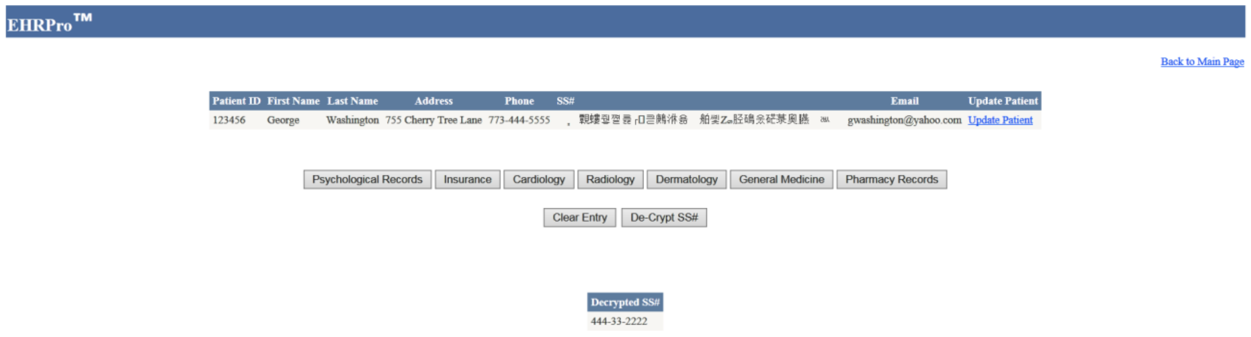


WELCOME TO THE ELECTRONIC HEALTH RECORD

PatientID	First Name	Last Name	Patient Address	Phone	Email	SS #	More Information	Select
196324	Thomas	Jefferson	100 Monticello Lane	630-547-8566	tjefferson@hotmail.com	[illegible]	Click Here	☐
854254	Harry	Truman	100 Defeats Dewey Drive	414-852-3698	htruman@gmail.com	[illegible]	Click Here	☐
654789	Jimmy	Carter	8756 W. Peach Tree St.	457-569-5522	jcarter@usa.gov	[illegible]	Click Here	☐

Figure 6.5: Patient List for user Joe

Cell Level Security and Encryption 6.3.1 Cell level security is also in place, as displayed by the patient’s social security number being encrypted upon the page loading. The text that is displayed is ciphertext, or result of the algorithm of encryption, which in this case is TRIPLE_DES. The text is usually a mixed up version of plain text that is unreadable to humans or computers unless decrypted with another symmetric key. Users can decrypt the SS# field to display its contents by pressing the ‘De-Crypt SS#’ button. The following figure demonstrates the process.



Patient ID	First Name	Last Name	Address	Phone	SS#	Email	Update Patient
123456	George	Washington	755 Cherry Tree Lane	773-444-5555	[illegible]	gwashington@yahoo.com	Update Patient

Decrypted SS#
444-33-2222

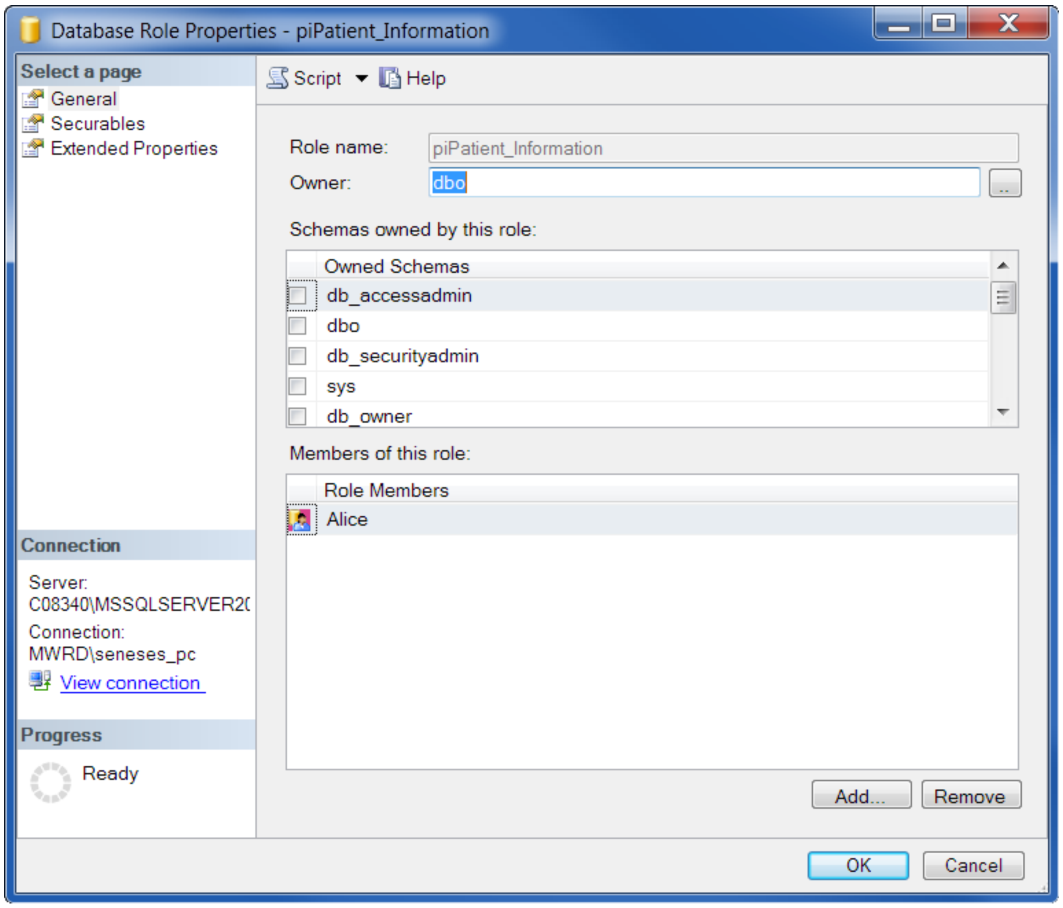
Figure 6.6: Decryption example

The explanation behind the decryption process is the application calling a stored procedure called '`usp_Decrypt_SSNumber`'. Inside the procedure the following code executes the decryption.

```
OPEN MASTER KEY DECRYPTION BY PASSWORD =' ' ---Password was removed for demo

OPEN SYMMETRIC KEY PasswordFieldSymmetricKey
DECRYPTION BY CERTIFICATE PasswordFieldCertificate;

SELECT

CONVERT(nvarchar, DecryptByKey(EncryptedSS#Number)) AS 'Decrypted SS#'
FROM [PatientInformation]
```

The database Master Key must be opened to perform symmetric key and certificate actions. Next the symmetric key is opened – which uses the appropriate key owned by the 'KeyBroker_EHR' owner to access the corresponding certificate. The last part converts the ciphertext using the **DecryptByKey** T-SQL function and decrypts data using the symmetric key defined in the prior lines of code above it. With the accurate symmetric keys open, selecting from the view causes labeled cells to be visible if the user's label dominates the label scheme. All other labeled cells appear as the ciphertext. More taxonomies can be used in row and cell level security.

Marking Strings Example 6.3.2 The marking strings created as database roles determine what a user can and cannot see.


Database Role Properties - piPatient_Information

Role name: piPatient_Information

Owner: dbo

Schemas owned by this role:

	Owned Schemas
☐	db_accessadmin
☐	dbo
☐	db_securityadmin
☐	sys
☐	db_owner

Members of this role:

Role Members
Alice

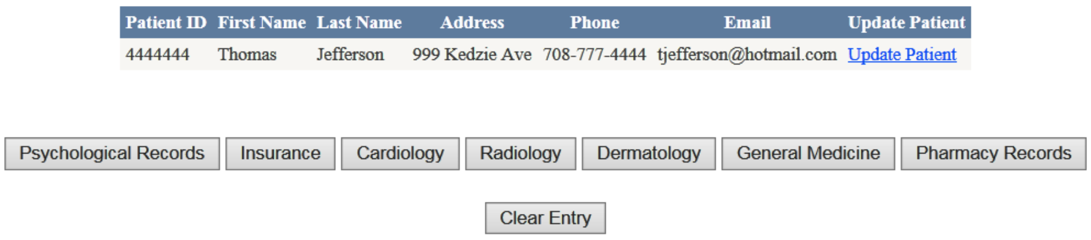
Connection

Server: C08340\MSSQLSERVER2(

Connection: MWRD\seneses_pc

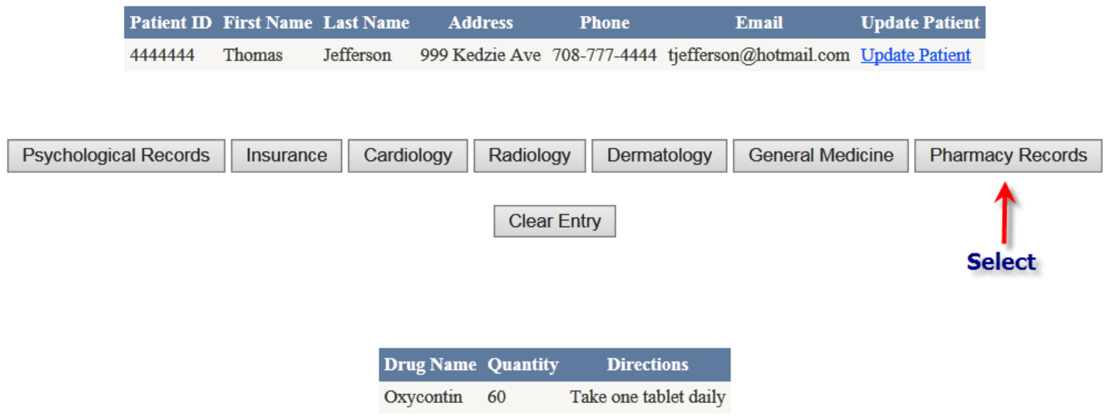
Figure 6.7: Database role properties

In the database those roles are given to the user at the same time a login is granted. The roles have the appropriate schemas attached to them as they are the owners of those schemas. When a user logs in and navigates through the application they are able (or not) to select different patient information based on these label security roles.



Patient ID	First Name	Last Name	Address	Phone	Email	Update Patient
4444444	Thomas	Jefferson	999 Kedzie Ave	708-777-4444	tjefferson@hotmail.com	Update Patient

Figure 6.8: Patient information

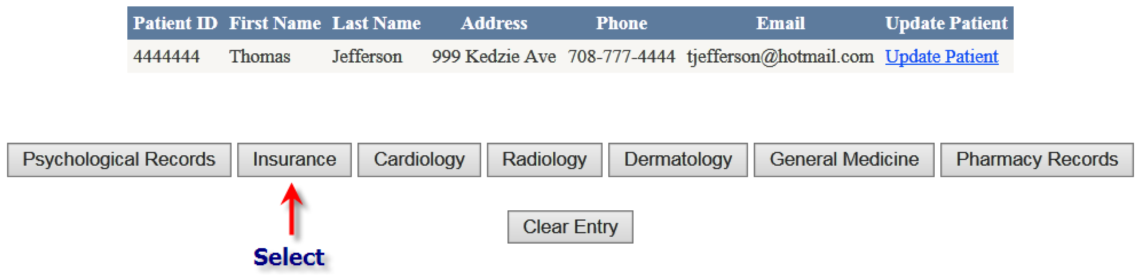


Patient ID	First Name	Last Name	Address	Phone	Email	Update Patient
4444444	Thomas	Jefferson	999 Kedzie Ave	708-777-4444	tjefferson@hotmail.com	Update Patient

Drug Name	Quantity	Directions
Oxycontin	60	Take one tablet daily

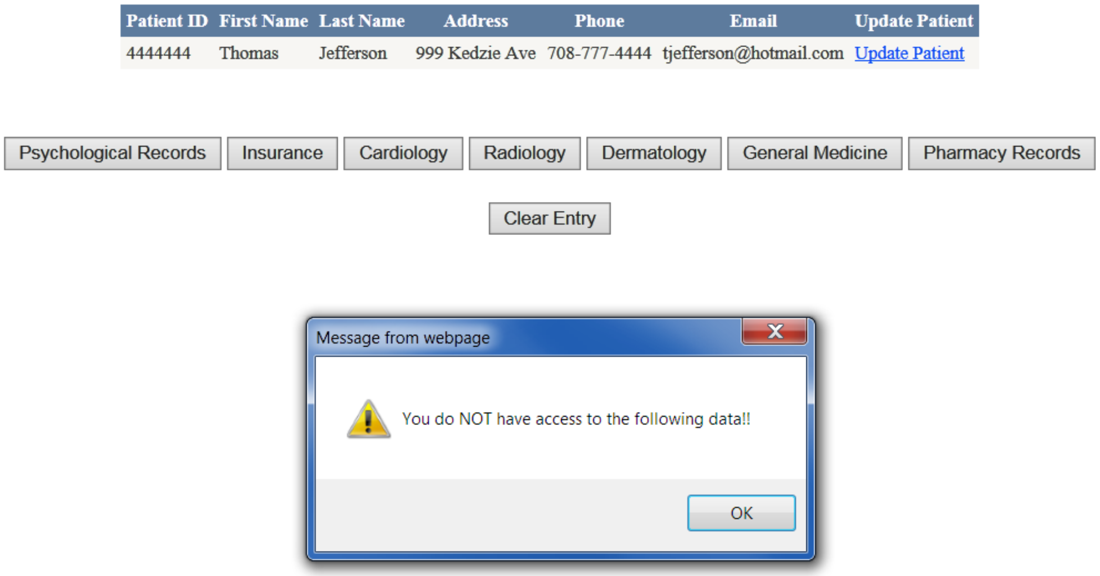
Figure 6.9: Displaying patient information when authorized

The application code calls a query that accesses the appropriate view. This view containes a call to a helper view that concentrates on the actual row-level security logic. The name of the view is vwVisibleLabels. If the information is available the row is displayed or the data in the row is blank. If the user does not have the access to the security label role then they are denied access to the information. This can be controlled at the most grainular level with mutiple compartments and child labels.



Patient ID	First Name	Last Name	Address	Phone	Email	Update Patient
4444444	Thomas	Jefferson	999 Kedzie Ave	708-777-4444	tjefferson@hotmail.com	Update Patient

Figure 6.10: Selecting patient information when not authorized

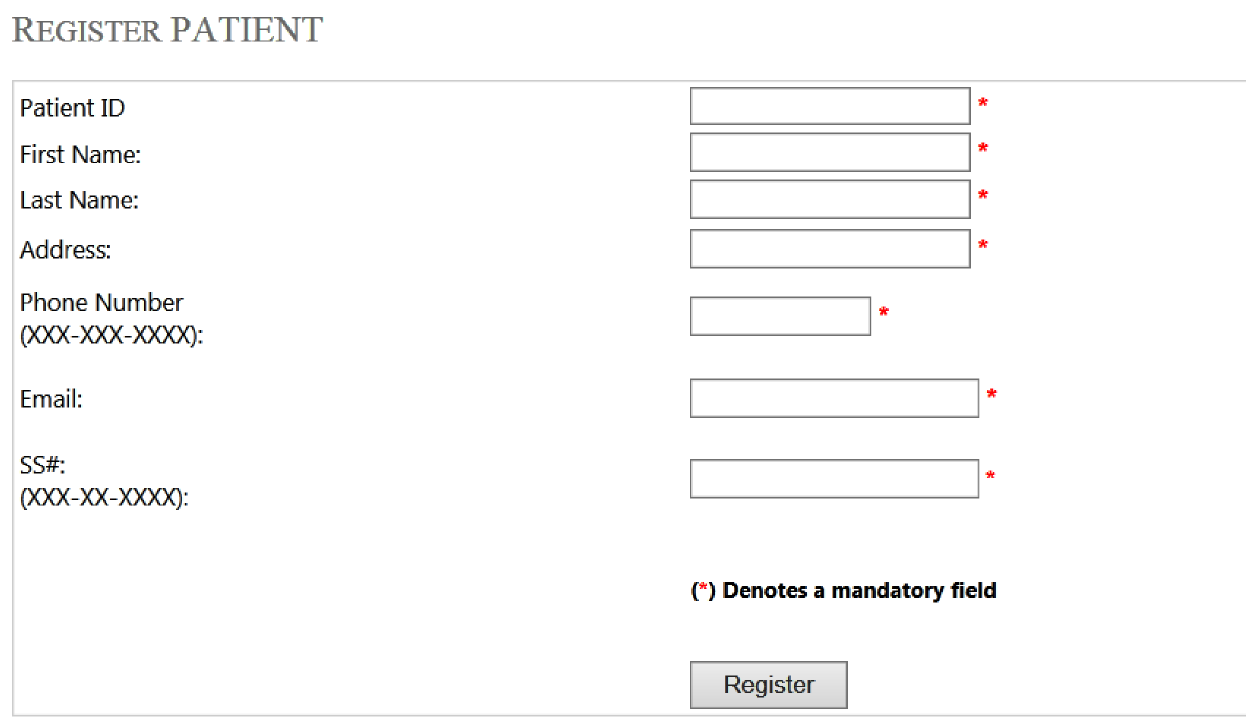


Patient ID	First Name	Last Name	Address	Phone	Email	Update Patient
4444444	Thomas	Jefferson	999 Kedzie Ave	708-777-4444	tjefferson@hotmail.com	Update Patient

You do NOT have access to the following data!!

Figure 6.11: Example of unauthorized access

INSERT, UPDATE, and DELETE Examples 6.3.3 The application also demonstrates explicit labeling for INSERT, UPDATE, and DELETE. The following example displays the INSERT properties by inserting patient information and medical history. The user selects the 'Register Patient' link from the menu and the page navigates to the entry screen.


REGISTER PATIENT

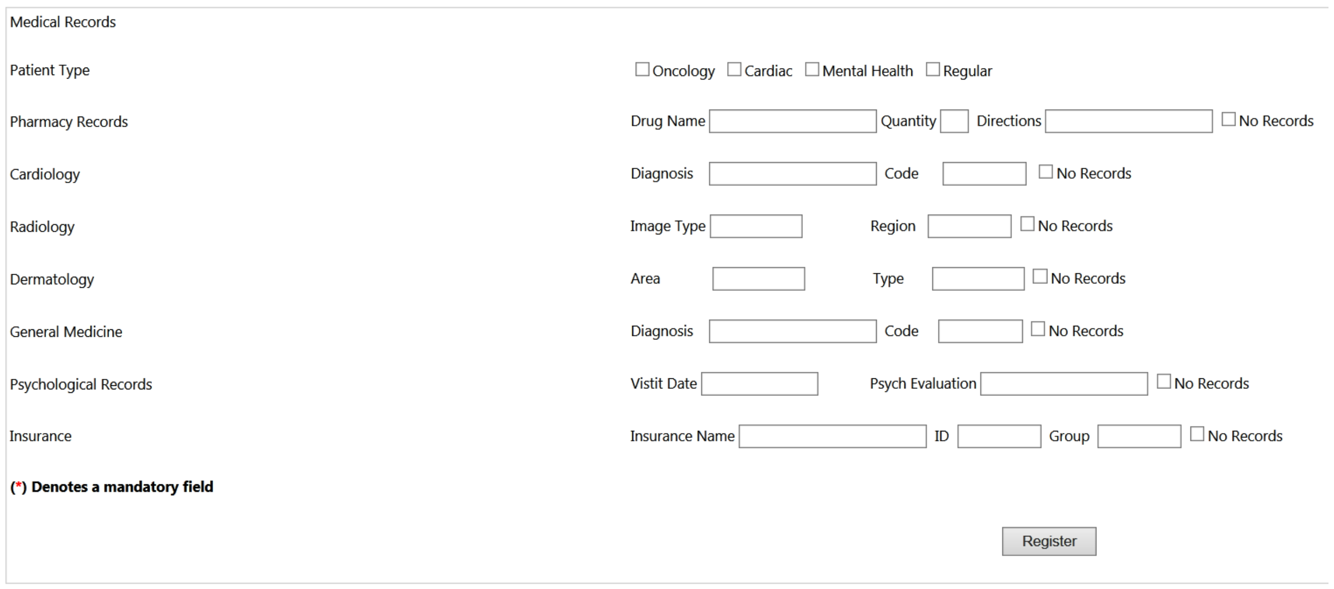
Patient ID	*
First Name:	*
Last Name:	*
Address:	*
Phone Number (XXX-XXX-XXXX):	*
Email:	*
SS#: (XXX-XX-XXXX):	*

(*) Denotes a mandatory field

Figure 6.12: Patient registration for INSERT of new data

The user enters pertinent information about the patient and clicks the 'Register' button. The user then is prompted to enter the medical records listed for the patient, not all records are required. You can select 'No Records' if the information is not available for the patient at that moment. Not all records are required considering that the patient may not have some of the medical issues or be in the care of certain specialists at the time of entry. Unlike the first part of the registration where vital personal information is required and are mandatory fields. This is the final step of the registration process.

RECORDS ENTRY



Medical Records	
Patient Type	☐ Oncology ☐ Cardiac ☐ Mental Health ☐ Regular
Pharmacy Records	Drug Name ▭ Quantity ▭ Directions ▭ ☐ No Records
Cardiology	Diagnosis ▭ Code ▭ ☐ No Records
Radiology	Image Type ▭ Region ▭ ☐ No Records
Dermatology	Area ▭ Type ▭ ☐ No Records
General Medicine	Diagnosis ▭ Code ▭ ☐ No Records
Psychological Records	Vistit Date ▭ Psych Evaluation ▭ ☐ No Records
Insurance	Insurance Name ▭ ID ▭ Group ▭ ☐ No Records

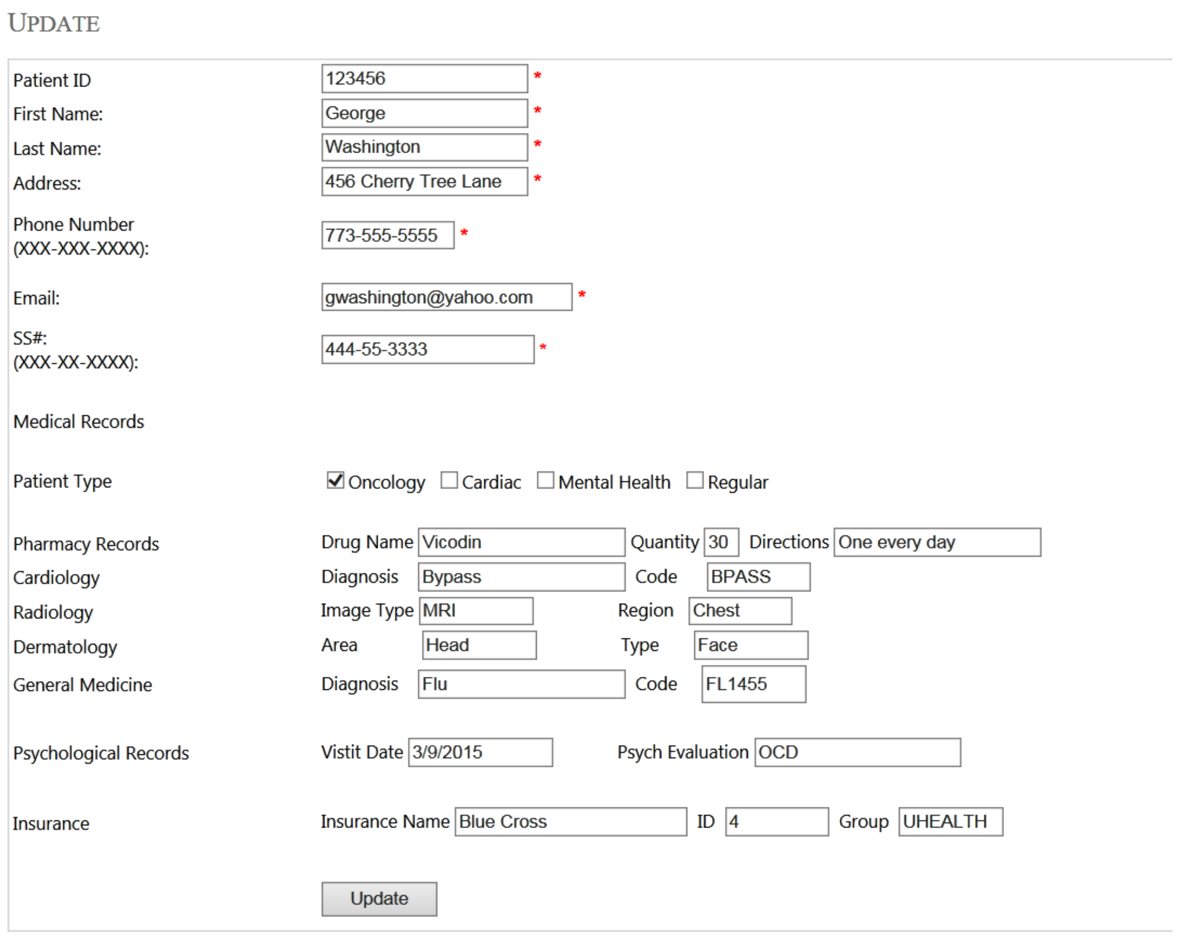
(*) Denotes a mandatory field

Figure 6.13: Final step of registration

At this point is where the row level security begins as the entered information is wrapped in security labels duing the INSERT statement to the database. The encryption of cell data can be conducted in considerably the same way we handled complex rules for row-level security. INSTEAD OF triggers defined on the view manage any necessary security rule checks and also encrypt the cell(s) based on their label. INSTEAD OF INSERT triggers can be described on a view or table to replace the standard action of the INSERT statement. Typically, the INSTEAD OF INSERT Trigger is defined on a view to insert data into one or more base tables. This trigger will activate after the vb.net code INSERT statement is called. The following code defines the INSTEAD OF INSERT Trigger.

```
INSTEAD OF INSERT
AS

       SET NOCOUNT ON
       DECLARE @labels TABLE(Label nvarchar(max), LabelID int)

       INSERT INTO @labels (Label, LabelID)
       SELECT DISTINCT CAST(Label AS nvarchar(max)), NULL FROM inserted

       DECLARE @ThisLabel  xml(dbo.SecurityLabel)
       DECLARE c_Labels CURSOR FOR
       SELECT Label FROM @labels

       OPEN c_Labels
       FETCH NEXT FROM c_Labels INTO @ThisLabel
       WHILE @@FETCH_STATUS = 0
       BEGIN
              DECLARE @LabelID int
              exec usp_GetSecLabelID @ThisLabel, @LabelID OUTPUT

              UPDATE @labels SET LabelID = @LabelID
              WHERE Label = CAST(@ThisLabel AS nvarchar(max))

              FETCH NEXT FROM c_Labels INTO @ThisLabel
       END
       CLOSE c_Labels
       DEALLOCATE c_Labels

       INSERT INTO tblPharmacyRecords (PatientID, Drugname, Quantity, Directions, RLS_ID)
       SELECT inserted.PatientID, inserted.Drugname, inserted.Quantity,
                     inserted.Directions, L.LabelID
       FROM inserted JOIN @labels L ON (CAST(inserted.Label AS nvarchar(max)) = L.Label)
```

When the trigger begins the insert operation, it declares a table variable and fills it with the distinct row labels in the inserted virtual table. A simple cursor loop is used to grab the security label ID for each of these. The resulting table variable is used in the actual insert to assign the row label ID for each row that is added to the base table. The INSTEAD OF INSERT trigger in this example uses a helper function from the label policy **usp_GetSecLabelID [25]** , this procedure is used to retrieve the security label ID that corresponds to the user label. This procedure generates the ID if it does not exist already. Then it is a simple matter of inserting to the base table, including the label ID in the INSERT statement. The UPDATE feature works in a simular fashion. The user will click on an ‘Update Patient’ link and the available information will be displayed for the patient. At this moment the information can be changed and updated.


UPDATE

Patient ID	123456 *					
First Name:	George *					
Last Name:	Washington *					
Address:	456 Cherry Tree Lane *					
Phone Number (XXX-XXX-XXXX):	773-555-5555 *					
Email:	gwashington@yahoo.com *					
SS#: (XXX-XX-XXXX):	444-55-3333 *					
Medical Records						
Patient Type	[✓ Oncology] Cardiac Mental Health Regular					
Pharmacy Records	Drug Name	Vicodin	Quantity	30	Directions	One every day
Cardiology	Diagnosis	Bypass	Code	BPASS		
Radiology	Image Type	MRI	Region	Chest		
Dermatology	Area	Head	Type	Face		
General Medicine	Diagnosis	Flu	Code	FL1455		
Psychological Records	Vistit Date	3/9/2015	Psych Evaluation	OCD		
Insurance	Insurance Name	Blue Cross	ID	4	Group	UHEALTH

Update

Figure 6.14: Updating patient information

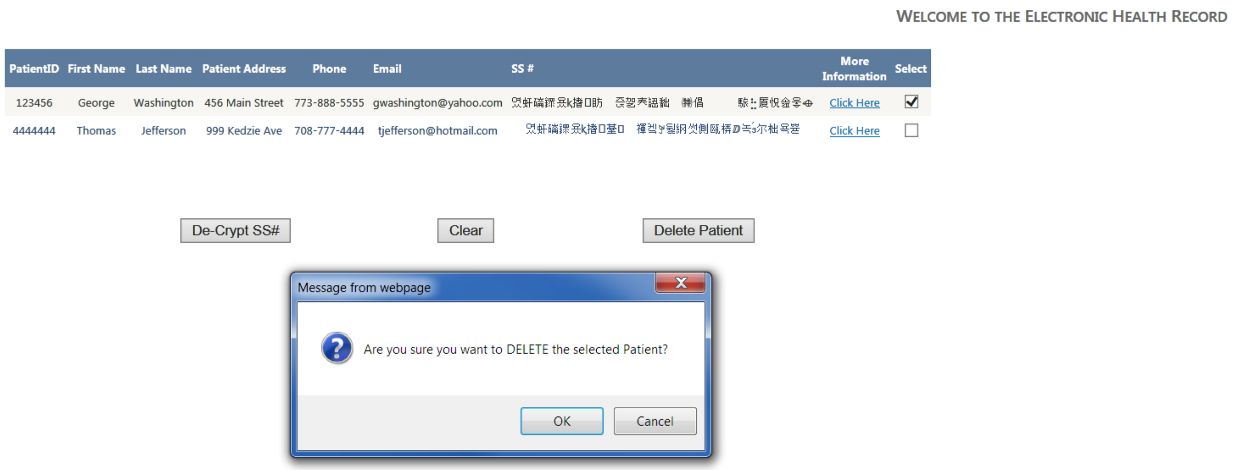
The following code defines one of the INSTEAD OF UPDATE Triggers. There are triggers for each view that correspond to each table. The INSERT Trigger works the same way with a trigger supplied for each view and it’s matching table.

```
INSTEAD OF UPDATE
AS
       SET NOCOUNT ON
       IF NOT UPDATE(Label)
       BEGIN
              UPDATE tblPatientInformation
```

```
        SET FirstName = inserted.FirstName,
               LastName = inserted.LastName,
               PatientAddress = inserted.PatientAddress,
               Phone = inserted.Phone,
               Email = inserted.Email,
               SS#Number = inserted.SS#Number,
               EncryptedSS#Number = inserted.EncryptedSS#Number


        FROM tblPatientInformation, inserted, deleted, vwVisibleLabels
        WHERE inserted.ID = tblPatientInformation.ID
        AND  deleted.ID = tblPatientInformation.ID
        AND  tblPatientInformation.RLS_ID = vwVisibleLabels.ID
END
ELSE
BEGIN
        DECLARE @labels TABLE(Label nvarchar(1024), LabelID int)

        INSERT INTO @labels (Label, LabelID)
        --SELECT Label, NULL FROM inserted
        SELECT DISTINCT CAST(Label AS nvarchar(max)), NULL FROM inserted

        DECLARE @ThisLabel XML(dbo.SecurityLabel)
        DECLARE c_Labels CURSOR FOR
        SELECT Label FROM @labels

        OPEN c_Labels
        FETCH NEXT FROM c_Labels INTO @ThisLabel
        WHILE @@FETCH_STATUS = 0
        BEGIN
               DECLARE @LabelID int
               exec usp_GetSecLabelID @ThisLabel, @LabelID OUTPUT

               UPDATE @labels SET LabelID = @LabelID
               WHERE Label = CAST(@ThisLabel AS nvarchar(max))

               FETCH NEXT FROM c_Labels INTO @ThisLabel
        END
        CLOSE c_Labels
        DEALLOCATE c_Labels

        UPDATE tblPatientInformation
        SET

        FirstName = inserted.FirstName,
               LastName = inserted.LastName,
               PatientAddress = inserted.PatientAddress,
               Phone = inserted.Phone,
               Email = inserted.Email,
               SS#Number = inserted.SS#Number,
               EncryptedSS#Number = inserted.EncryptedSS#Number,
               RLS_ID = L.LabelID

        FROM tblPatientInformation, inserted, @labels L, deleted, vwVisibleLabels
        WHERE
        --inserted.Label = L.Label
        --AND
```

```
                  inserted.ID = tblPatientInformation.ID
                 AND  deleted.ID = tblPatientInformation.ID
                 AND  tblPatientInformation.RLS_ID = vwVisibleLabels.ID
END
```

Lastly users are permitted to DELETE a patients information. The DELETE feature is straightforward, select a patient and click the ‘Delete’ button as shown in the figure.



PatientID	First Name	Last Name	Patient Address	Phone	Email	SS #	More Information	Select
123456	George	Washington	456 Main Street	773-888-5555	gwashington@yahoo.com	[illegible]	Click Here	☑
4444444	Thomas	Jefferson	999 Kedzie Ave	708-777-4444	tjefferson@hotmail.com	[illegible]	Click Here	☐

Are you sure you want to DELETE the selected Patient?

Figure 6.15: Deleting patients

DELETE triggers can be controlled as previous examples have shown – use the deleted virtual table, together with the base table(s) and the Visible Labels view to limit the result of the DELETE operation to rows available to the current user, as in the following code.

```
INSTEAD OF DELETE

AS

       DELETE tblPatientInformation
       FROM tblPatientInformation, vwVisibleLabels, deleted
       WHERE tblPatientInformation.RLS_ID = vwVisibleLabels.ID
        AND tblPatientInformation.ID = deleted.ID
```

The patient information and all subsequent EHR information from the corresponding tables is deleted. No records are availble for that patient after a deletion occurs.

Example Conclusion 6.4

The prototype application demonstrates the functionality of the different INSERT, UPDATE, and DELETE methods available with row level security. The prototype is only one example of the numerous posibilites that can exsist with security labels at the row and cell level.

CHAPTER 7

Future Research 7.1

The electronic health record (EHR) is becoming more of an accepted pratice among health care providers. There are still many concerns about the security and confidentialty of how and even where patients data is kept. Privacy is something quite different from “keeping private”. [8] With the creation of Cloud Computing there are a whole new set of concerns about EHR information. At the same time, cloud computing has attracted a lot of attention because it provides storage-as-a-service and software-as-a-service, by which software service providers can enjoy the virtually infinite and elastic storage and computing resources [13] There's also another much worse security piece of bad news: The nature of mobile apps, with all of their interdependent parts, has opened a huge number of security problems, which have caught many large companies unaware. [30] There are also new frameworks being discussed due to the mobile and cloud technology.

New Frameworks 7.2

With the changes in the Internet the introduction of Cloud computing and the rapid development of mobile application s new frameworks have introduced themselves. Very often the access control mechanism is chosen first (because of platform constraints) and the existing access control requirements for the application are often adjusted to fit into the limitations of the mechanism at hand [10] Some frameworks are actually taking into account the requirements for EHR’s or health networks. A distributed healthcare service infrastructure, however, implies the capability that is required to cope with the administrative burden and the continuous maintenance needs arising from fully functional and networked clinical centers, each of which has its own users, data, access policies, and which assumes that cross-center access is the norm. [37]

Patient-Centric Privacy 7.2.1 In “patient-centric privacy”, we envision that each patient specifies her own privacy policy. [20] One group has proposed a Unified Logical EHR Model. In our model, both the EHR instances and the aggregated virtual composite EHR are uniformly modelled as a labelled hierarchical structure. The nodes represent the clinical data elements that need to be protected for sharing. [16] Another example is the MedVault sharing framework. The framework's authorization module is an attribute-based system, where the patient specifies authorization policies based on specific attributes (with specific values) that the requester must hold in order to gain authorization to access the records.[26]

Broker Based Framework 7.2.2 Other sharing ideas include Health Information Exchanges or Broker-Based approaches using the Cloud. They can reside in a single cloud or multiple clouds (public cloud, private cloud, or hybrid cloud) depending on their deployment needs. [36] The following figure demonstraits an example of a broker-based system and its details.

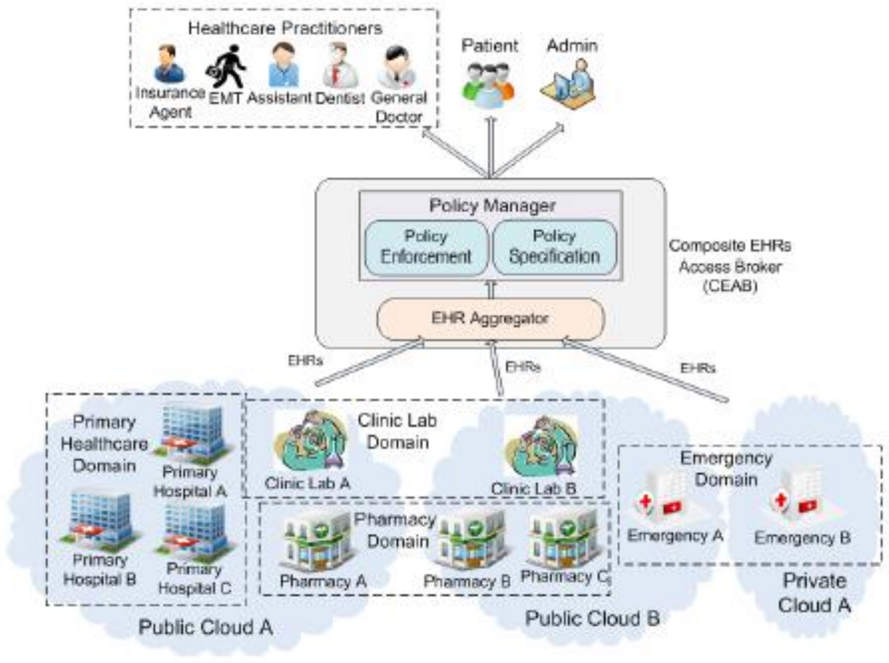


Figure 7.1: A Broker Based Approach [36]

There are methodologies being proposed that will allow Healthcare providers to register their credentials in the UDDI (Universal Description, Discovery and Integration) structure. The UDDI will allow these Healthcare entities to add patient information, services, and data to a broker. Patient metadata stores basic patient information and it maps the original database in healthcare providers with the UDDI which increase broker flexibility to find services according to the data they carry. [18] Another proposal has healthcare using Case-Based Reasoning (CBR) combined with other technology like intelligent agents. CBR uses patient information and case histories, studies, trials and the information and data from their outcomes. Intelligent agents provide an autonomous venue to operate routine tasks and make decisions for users and adapt to their changing needs. It’s a cooperative distributed solution which allows the numerous healthcare actors to share their information and benefit from sub-systems ‘capabilities in open distributed healthcare environments. [5]

Outlook and Vision 7.3

The future of EHR data can be valuable to many different parties. Currently some government agencies are creating EHRSS systems also known as Electronic Health Record Surveilance Systems. Population health surveillance seeks to monitor health across a range of indicators that together embody key characteristics of the health of the population in question. [27] This data is extracted from EHR's where tradtionaly these systems would rely on census, medical studies, and population surveys. Future of Security Labels and access control Access control has grown into a multi-purpose security feature and the limits for design are growing in numbers. Security Labels are at a juncture where more implementation is needed, especialy at the database level. The potential for their use is far from being uttilized and the solutions they offer are not always being considered. Multilevel security or MLS as it is refrered to sometimes is a field that needs to be explored futher considering the vast changes undergoing the Internet with Clould computing and mobile devices.

CHAPTER 8

Conclusion 8.1

Access Control provides a solution for securing Electronic Health Records and systems. Coupled with row and cell level security while instituting encryption, this combination provides a security policy for safeguarding sensitive health information as well as other disciplines. We presented a vision of using this type of security to enact a type of solution for a progressive issue challenging the IT Healthcare frontier. We developed a prototype example to demonstrate the possible solution and preformed actual tests to produce results. The example and results exhibit that by using the principle of Security Labels as an Access Control method that data can and is secured to a level that gives health records a more reliable option.

Bibliography

[1] Abraham, S. E. (2013). Distributed Attribute Based Encryption for Patient Health Record Security under Clouds. (p 297)

[2] Agrawal, R., & Johnson, C. Securing electronic health records without impeding the flow of information. IBM Almaden Research Center (p 1).

[3] Agrawal, R., Kiernan, J., Srikant, R., & Xu, Y. Hippocratic Databases, IBM Almaden Research Center (2002). (p 4).

[4] Alhaqbani, B., & Fidge, C. (2007). Access Control Requirements for Processing Electronic Health Records. (p 3).

[5] Al-Sakran, H. O. (2014). Framework architecture for improving healthcare information systems using agent technology.

[6] Anderson, R. J. A Security Policy Model for Clinical Information Systems. (p 2).

[7] Appari, A., & Johnson, M. E. (2008). Information Security and Privacy in Healthcare: Current State of Research1. (p 3).

[8] AXIOMATICS. Attribute Based Access Control (ABAC) – the Best Cure for Sustainable eHealth Services [White paper]. Retrieved from http://www.axiomatics.com/component/rsfiles/preview.html?path=White%2BPapers%252FXACML%2Bfor%2BeHealth%2BSolutions-final.pdf

[9] Benaloh, J., Chase, M., Horvitz, E., & Lauter, K. (2009). Patient Controlled Encryption: Ensuring Privacy of Electronic Medical Records. (p 1).

[10] Chandramouli, R. Business Process Driven Framework for defining an Access Control Service based on Roles and Rules.

[11] Centers for Disease Control and Prevention. (2015). diagram illustrates how access control and encryption work together to secure data. Retrieved from http://www.cdc.gov/cancer/npcr/tools/security/encryption2.htm

[12] Dekker, M. A. C., & Etalle, S. (2006). Audit-Based Access Control for Electronic Health Records. (p 222).

[13] Fox, A. (2009). Above the Clouds: A Berkeley View of Cloud Computing.

[14] Gajanayake, R., Iannella, R., & Sahama, T. (2012). Privacy Oriented Access Control for Electronic Health Records.

[15] Isode. Access Control using Security Labels & Security Clearance. [White paper]. Retrieved from http://www.isode.com/whitepapers/security-labels-clearance.html

[16] Jin, J., Ahn, G. J., Covington, M. J., & Zhang, X. (2000). Toward an Access Control Model for Sharing Composite Electronic Health Records. (p 1)

[17] Jin, J., Ahn, G. J., Hu, H., Covington, M. J., & Zhang, X. (2009). Patient-centric Authorization Framework for Sharing Electronic Health Records.

[18] Khoury, I., Wu, C. S., Chang, W. C., & Cebeci, S. (2011). e-Healthcare Web Service Broker Infrastructure for Medical Data Exchange. *Oakland university Research Paper, USA.*

[19] Kume, N., Okamoto, K., Kuroda, T., & Yoshihara, H. The Electronic Health Record as a Clinical Study Information Hub. (p 45).

[20] Li, M., Yu, S., Ren, K., & Lou, W. Securing Personal Health Records in Cloud Computing: Patient-Centric and Fine-Grained Data Access Control in Multi-owner Settings. (p 94).

[21] Liu, C., Billard, A., Ozols, M., & Jeremic, N. Access Control Models and Security Labelling. Information Networks Division Defense Science and Technology Organization PO Box 1500, Edinburgh. (p 1).

[22] Liu, Y. A., & Stoller, S. D. (2005). Role-based access control: A simplified specification. Technical Report DAR 05-24, Computer Science Department, SUNY Stony Brook.

[23] Mattsson, U. T. (2005). Database encryption-how to balance security with performance. Available at SSRN 670561. (p 2).

[24] Microsoft. (2012 March). SQL Server Label Security Toolkit Using the Label Security Toolkit. Retrieved from https://sqlserverlst.codeplex.com/documentation

[25] Microsoft. (2012 March). SQL Server Label Security Toolkit Database Developer's Reference, Retrieved from https://sqlserverlst.codeplex.com/documentation

[26] Mohan, A., Bauer, D., Blough, D. M., Ahamad, M., Bamba, B., Krishnan, R., & Palanisamy, B. (2009). A patient-centric, attribute-based, source-verifiable framework for health record sharing.

[27] New York City Department of Health and Mental Hygiene (2013, July) Developing an Electronic Health Record-Based Population Health Surveillance System. Retrieved from http://www.nyc.gov/html/doh/downloads/pdf/data/nyc-macro-report.pdf (p 1).

[28] Health Level Seven International (2013, May) Guide to the HL7 Healthcare Privacy and Security Classification System (HCS). Retrieved from https://www.hl7.org/documentcenter/public_temp_0F371F9D-1C23-BA17-0C52D857D96DB0A2/wg/secure/3.%20HCS%20Guide%20Final%202013%200322%20JMD.pdf (p 9).

[29] Rask, A. Implementing Row and Cell-Level Security in Classified Databases, Microsoft (2012 January).

[30] Schuman, E. (2014). The future of health IT security, 1. Retrieved from http://www.govhealthit.com/news/future-health-it-security

[31] Stallings, W., & Brown, L. (2008). Computer Security. Principles and Practice.

[32] Terry, N. P., & Francis, L. P. (2007). Ensuring the privacy and confidentiality of electronic health records. U. Ill. L. Rev., 683.

[33] Rjaibi, W., Pickel, J., Leffler, J., & Tassi, B. Best Practices. (p 5).

[34] Solomon, M., & Chapple, Mike. (2005). Information Security Illuminated. Burlington, MA: Jones & Bartlett Learning. (p 34).

[35] Weber, H. A. (2003). Role-based access control: the NIST solution. Sans Institute.

[36] Wu, R., Ahn, G. J., & Hu, H. Secure Sharing of Electronic Health Records in Clouds. (p 2).

[37] Xiao, L., Hu, B., Croitoru, M., Lewis, P., & Dasmahapatra, S. A Knowledgeable Security Model for Distributed Health Information Systems.