

SCALABLE AND SECURE SHARING OF PERSONAL HEALTH RECORDS USING ATTRIBUTE-BASED ENCRYPTION (ABE)

Project Report Submitted in partial fulfillment of the requirement

For the degree of

Bachelor of Technology

In

Computer Science Engineering

Under the Supervision of

SUMAN SAHA

By

RUCHI AGGARWAL 101323

To



JAYPEE UNIVERSITY OF INFORMATION AND TECHNOLOGY

WAKNAGHAT, SOLAN-173215, HIMACHAL PRADESH

JAYPEE UNIVERSITY OF INFORMATION AND TECHNOLOGY

WAKNAGHAT, SOLAN-173215, HIMACHAL PRADESH

CERTIFICATE

This is to certify that the work titled “**Scalable and Secure Sharing of Public Health Records using Attribute-Based Encryption**” Submitted By **Ruchi Aggarwal (101323)** in partial fulfillment for the award of degree of B.Tech Computer Science Engineering of Jaypee University of Information Technology, Waknaghat has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

(Signature of Supervisor)

Name of Supervisor: Suman Saha

Designation: Assistant Professor, Dept. of CSE and ICT

Jaypee University of Information Technology

Date:

ACKNOWLEDGEMENT

I would like to express my greatest gratitude to all the people who have helped and supported me throughout the project. I am grateful to my mentor **Suman Saha** for his continuous support for the project, from initial advice and contacts in the early stages of conceptual inception and through ongoing process and to this day.

Thanks and appreciation to the helpful people at college for their support. I would also thank our university and my faculty members without whom, this project would have been a distant reality. I also extend my heartfelt thanks to my family and well wishers.

A special thanks to all the fellow students who helped me in completing the project and exchanged their interesting ideas, thoughts and made this project easy and accurate.

RUCHI AGGARWAL

101323

B.Tech-CSE

DATE:

Table of Contents

Certificate

Acknowledgement

Table of Contents

List of Figures

Chapters

1 Introduction	1-3
1.1 Problem Statement	
1.2 Motivation	
1.3 Objective	
1.4 Scope and Deliverables	
2 Literature Review	4-11
2.1 Established Findings	
2.2 SKC & PKC Based Solutions	
2.3 ABE Based Solution	
2.4 Fine-grained Access Control	
2.5 Secret Sharing Schemes	
2.6 Attribute Based Encryption	
3 Requirement Elicitation and Analysis	12-17
3.1 Requirement Analysis	
3.2 Top level use case and sequence Diagrams	
3.3 Design Specifications	

4 Project Plan and Implementation Details	18-28
4.1 Sharing of PHR	
4.2 Implementation of PHR System	
4.3 Algorithm for PHR System	
4.4 Module Description	
4.5 Block Diagram of PHR System	
4.6 Stepwise Implementation Algorithm	
4.7 Encryption and Decryption	
5 Coding	29-38
6 Advantages and Applications	39
7 Output	40-49
8 Conclusion and Future Scope	50
9 References	51-52

LIST OF FIGURES

- Figure 1 - .NET Framework Stack
- Figure 2 - .NET Framework 3.5
- Figure 3 - Use Case Admin Session
- Figure 4 – Use Case PHR Owner Session
- Figure 5 - Use Case Data Access Control Session
- Figure 6 – Sequence Diagram
- Figure 7 – ER Diagram
- Figure 8 – Flow Chart for PHR System
- Figure 9 – Module Description
- Figure 10 - Access Policy Table
- Figure 11 – Block Diagram Of PHR System
- Figure 12 – Stepwise Algorithm
- Figure 13 – DES Encryption Block Diagram

JAYPEE UNIVERSITY OF INFORMATION AND TECHNOLOGY

WAKNAGHAT, SOLAN-173215, HIMACHAL PRADESH

CERTIFICATE

This is to certify that the work titled "**Scalable and Secure Sharing of Public Health Records using Attribute-Based Encryption**" Submitted By **Ruchi Aggarwal (101323)** in partial fulfillment for the award of degree of B.Tech Computer Science Engineering of Jaypee University of Information Technology, Wagnaghat has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Suman Saha

(Signature of Supervisor)

Name of Supervisor: Suman Saha

Designation: Assistant Professor, Dept. of CSE and ICT

Jaypee University of Information Technology

Date: *13. May. 2014*

JAYPEE UNIVERSITY OF INFORMATION AND TECHNOLOGY
WAKNAGHAT, SOLAN-173215, HIMACHAL PRADESH

CERTIFICATE

This is to certify that the work titled “**Scalable and Secure Sharing of Public Health Records using Attribute-Based Encryption**” Submitted By **Ruchi Aggarwal (101323)** in partial fulfillment for the award of degree of B.Tech Computer Science Engineering of Jaypee University of Information Technology, Wagnaghat has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Suman Saha

(Signature of Supervisor)

Name of Supervisor: Suman Saha

Designation: Assistant Professor, Dept. of CSE and ICT

Jaypee University of Information Technology

Date: *13. May. 2014*

ACKNOWLEDGEMENT

I would like to express my greatest gratitude to all the people who have helped and supported me throughout the project. I am grateful to my mentor **Suman Saha** for his continuous support for the project, from initial advice and contacts in the early stages of conceptual inception and through ongoing process and to this day.

Thanks and appreciation to the helpful people at college for their support. I would also thank our university and my faculty members without whom, this project would have been a distant reality. I also extend my heartfelt thanks to my family and well wishers.

A special thanks to all the fellow students who helped me in completing the project and exchanged their interesting ideas, thoughts and made this project easy and accurate.

Ruchi Aggarwal

RUCHI AGGARWAL

101323

B.Tech-CSE

DATE: 13 May 2014

ACKNOWLEDGEMENT

I would like to express my greatest gratitude to all the people who have helped and supported me throughout the project. I am grateful to my mentor **Suman Saha** for his continuous support for the project, from initial advice and contacts in the early stages of conceptual inception and through ongoing process and to this day.

Thanks and appreciation to the helpful people at college for their support. I would also thank our university and my faculty members without whom, this project would have been a distant reality. I also extend my heartfelt thanks to my family and well wishers.

A special thanks to all the fellow students who helped me in completing the project and exchanged their interesting ideas, thoughts and made this project easy and accurate.

Ruchi Aggarwal

RUCHI AGGARWAL

101323

B.Tech-CSE

DATE: *13 May 2014*

1. INTRODUCTION

A personal health record (PHR) makes it easy to gather and manage medical information in one accessible and secure location. Carrying paper records is a big drawback, rarely the patients have with them when they need. Personal health record systems overcome this problem by making the personal health record accessible anytime via a Web-enabled device, such as computer. Literally, having a personal health record can be a lifesaver. In an emergency patient can quickly give emergency personal vital information about disease, medications and drug allergies. Now a day, personal health record (PHR) has become a patient-centric model of health information exchange. A PHR service allows patients to create, manage and control their personal health data from one place through the web, which has made the storing, retrieving and sharing of the medical information more efficient.

Especially, each patient will have full control of their medical records and can share their health data with different users from different domains which include healthcare providers, family members and friends. Due to high cost of building and maintaining separate data centers, many PHR services are outsourced and provided by third party service providers for example, Microsoft Health Vault (UK). The Health Vault Program of Microsoft will allow users including individuals, health centers, hospitals etc. to gain access to the information on health related issues. The user interface will be simple, that would allow anyone to operate the program easily. But on the other hand, many people do not wish to share their private health records and other information universally through the Health Vault. As the sensitive Personal Health Information (PHI) is highly valuable, the third-party storage servers are the targets of various malicious behaviors which may result in exposure of the PHI.

The main concern is about whether the patients could actually control the sharing of their sensitive PHR and other information, especially when they are stored on a third-party server which people may not fully trust. To ensure patient-centric privacy control over their own PHRs, encryption of data is necessary prior storage. Basically, the PHR owners themselves should decide how to encrypt their files and to allow which set of users to obtain access to each file. The PHR and other files are available to only those users who are given the corresponding decryption key and are confidential to other users. Furthermore, the patient will always have the right to not only to grant, but also to revoke access rights when it is necessary.

This scheme endeavors to study the patient-centric secure sharing of PHRs stored on semi-trusted servers, and focus on addressing the complicated and challenging key management

issues. To protect the personal health data stored on a semi-trusted server, Attribute-Based Encryption (ABE) method is used. Using ABE, files are encrypted under a set of attributes. Access policies are expressed based on the attributes of users or data, which enables patients to selectively share their PHR among a set of users. The complexities per encryption, key generation and decryption are only linear with the number of attributes involved. However, to integrate ABE into a large-scale PHR system, important issues such as key management scalability, dynamic policy updates, and efficient on-demand revocation are non-trivial to solve. To this end, the following are the main contributions:

- An ABE method is proposed for patient-centric secure sharing of PHRs, under the multi-owner settings. To overcome the key management challenges, the users in the system are divided into two categories, namely public domain and personal domain. In particular, large number of professional users is managed by attribute authorities (AA) in the public domain, while each owner only needs to manage the keys of a small number of users in their personal domain.
- In the public domain, multi-authority ABE (MA-ABE) is used to improve the security and avoid key management problem. Each attribute authority (AA) in it governs a disjoint subset of user role attributes. Encryption mechanism allows PHR owners to specify role-based access policies during file encryption. In the personal domain, owners directly assign access rights to personal users and encrypt a PHR file under its data attributes.

The main goal of our framework is to provide secure patient-centric PHR access and efficient key management at the same time. The key idea is to divide the system into multiple security domains (namely, public domains (PUDs) and personal domains (PSDs)) according to the different users' data access requirements. The PUDs consist of users based on their professional roles, such as doctors, nurses and medical researchers. In practice, a PUD can be hospital, government or insurance sector. In case of PSD, its users are personally associated with a data owner (such as family members or close friends), and they access the PHRs based on access rights assigned by the owner.

1.1 PROBLEM STATEMENT

- To describe a new approach that enables secure storage and controlled sharing of patient's health data.
- Explore key policy attribute based encryption and multi-authority attribute based encryption to enforce patient access control policy such that everyone can download the data ,but only authorize user can view the medical records.
- To supports multiple owner scenarios and divides the users in the system into multiple security domains that greatly reduce the key management complexity for owners and users.

1.2 MOTIVATION

What motivated us to choose this project was the concern about the privacy of patients' personal health data and who could gain access to the medical records.

Since patients lose physical control to their own personal health data, directly placing those sensitive data under the control of the servers cannot provide strong privacy assurance at all.

1.3 OBJECTIVES

- Secure and flexible access of information
- Scalability in key management
- Efficient user revocation by the authorized user
- Achieve fine grained and scalable data access control for medical records.

1.4 SCOPE AND DELIVERABLES

- Patient and client health care records document an individual's health evaluation, diagnosis, treatment, care, progress and health outcome.
- For access control of outsourced data, partially trusted servers are often assumed.
- The deliverables include detail design of modules and implementation Packages of the proposed framework.

LITRATURE REVIEW

2. LITERATURE SURVEY

2.1 Established findings

For access control of outsourced data, partially trusted servers are often assumed. With cryptographic techniques, the goal is trying to enforce who has (read) access to which parts of a patient's PHR documents in a fine-grained way.

2.2 Symmetric key cryptography (SKC) based solutions:

Symmetric-key algorithms are a class of algorithms for cryptography that use the same cryptographic keys for both encryption of plaintext and decryption of cipher text. (Solution proposed by Vimercati).

Public key cryptography (PKC) based solutions:

PKC based solutions were proposed due to its ability to separate write and read privileges. (Schemes proposed by J. Benaloh, M. Chase, E. Horvitz, and K. Lauter in their work "Patient controlled encryption: ensuring privacy of electronic medical records").

2.3 Attribute Based Encryption based solutions:

A number of works used ABE to realize fine-grained access control for outsourced data , Especially, there has been an increasing interest in applying ABE to secure electronic healthcare records (EHRs).(Scheme proposed by Narayan and Akinyele).

2.4 Fine-grained Access Control:

Fine-grained access control systems facilitate granting differential access rights to a set of users and allow flexibility in specifying the access rights of individual users. Several techniques are known for implementing fine grained access control. Common to the existing techniques and the references therein is the fact that they employ a trusted server that stores the data in clear. Access

control relies on software checks to ensure that a user can access a piece of data only if he is authorized to do so. This situation is not particularly appealing from a security standpoint. In the event of server compromise, for example, as a result of a software vulnerability exploit, the potential for information theft is immense. Furthermore, there is always a danger of insider attacks where in a person having access to the server steals and leaks the information, for example, for economic gains. Some techniques create user hierarchies and require the users to share a common secret key if they are in a common set in the hierarchy. The data is then classified according to the hierarchy and encrypted under the public key of the set it is meant for. Clearly, such methods have several limitations. If a third party must access the data for a set, a user of that set either needs to act as an intermediary and decrypt all relevant entries for the party or must give the party its private decryption key, and thus let it have access to all entries.

2.5 Secret-Sharing Schemes:

Secret-sharing schemes (SSS) are used to divide a secret among a number of parties. The information given to a party is called the share (of the secret) for that party. Every SSS realizes some access structure that denies the sets of parties who should be able to reconstruct the secret by using their shares. Shamir and Blakley were the first to propose a construction for secret-sharing schemes where the access structure is a threshold gate. That is, if any t or more parties come together, they can reconstruct the secret by using their shares; however, any lesser number of parties does not get any information about the secret. Benaloh extended Shamir's idea to realize any access structure that can be represented as a tree consisting of threshold gates. In SSS, one can specify a tree-access structure where the interior nodes consist of AND and OR gates and the leaves consist of different parties. Any set of parties that satisfy the tree can come together and reconstruct the secret. Therefore in SSS, collusion among different users (or parties) is not only allowed but required. In our construction each user's key is associated with a tree-access structure where the leaves are associated with attributes. A user is able to decrypt a cipher text if the attributes associated with a cipher text satisfy the key's access structure. In our scheme, contrary to SSS, users should be unable to collude in any meaningful way.

The encryption techniques in detail are as follows:

2.6 Attribute-Based Encryption:

Attribute-Based Encryption (ABE), a generalization of identity-based encryption that incorporates attributes as inputs to its cryptographic primitives. Data is encrypted using a set of

attributes so that multiple users who possess proper can decrypt. Attribute-Based Encryption (ABE) not only offers fine-grained access control but also prevents against collusion .One disadvantage of encrypting data is that it severely limits the ability of users to selectively share their encrypted data. Suppose a particular user wants to grant decryption access to a party to all of its Internet track logs for all entries on a particular range of dates that had a source IP address from a particular subnet. The user either needs to act as an intermediary and decrypt all relevant entries for the party or must give the party its decryption key and to have access to all entries. Neither one of these options is not applicable. An important setting where these issues give rise to serious problems is audit logs. Sahai and Waters [5] made some initial steps to solving this problem by introducing the concept of Attributed-Based Encryption (ABE). In an ABE system, a user's keys and ciphertexts are labeled with sets of descriptive attributes and a particular key can decrypt a particular ciphertext only if there is a match between the attributes of the ciphertext and the user's key.

To put it in other words Attribute-based access control is an approach where the access decision is based on attributes (properties) of the resource, the requestor and the environment. This model provides flexibility and scalability that are essential in large distributed open environments where subjects are identified by characteristics. It can be implemented using digital credentials, that is, digitally signed assertions about credential owner by a credential issuer. More precisely, an attribute certificate can be used to support attribute-based systems. Such certificate contains attributes that specify access control information associated with the certificate holder. The decision to access a resource is based on the attributes in requestor's credentials. The attempts to provide a uniform framework for attribute-based access control and enforcement include the works of Bonati and Samarati and Yu et al. [XACML (Xtensible Access Control Markup Language) is an XML specification for expressing policies for data control over the Internet. It is intended to define the representation for rules that specify who, what, when, and how can access information. The ABAC model can be easily implemented using XACML due to its generic authorization architecture. Attributes can be related to the subject, resource and environment. XACML allows using the subject, resource and environmental attributes in the policy evaluation. This language supports the following features:

- Policies can be shared across different applications.
- Policies can be maintained in one or more locations.
- The application environment is isolated from the authorization process.
- Different access control mechanisms are supported, e.g. ABAC, RBAC, etc.

Based on the above features, XACML is regarded as the standard for solving complex access control problems, e.g. the access control in healthcare. The major components and actors of the XACML framework data flow are:

- 1) PEP (policy enforcement point) is responsible for implementing the policy evaluation decision,
- 2) PDP (policy decision point) is responsible for evaluation of the policies,
- 3) PAP (policy administrator point) writes the access control policies and makes them available to the PDP,
- 4) PIP (policy information point) returns the requested attributes to the PDP for the policy evaluation.

Currently the OASIS Cross-Enterprise Security and Privacy Authorization (XSPA) technical committee is developing the XACML profile for healthcare enterprises.

Drawbacks:

Akinyele et al. investigated using ABE to generate self-protecting EMRs, which can either be stored on cell phones or cloud servers so that EMR could be accessed when health provider is offline also. However, there exist several common drawbacks for the above works discussed. First, assuming the use of a single trusted authority (TA) in the system. Single trusted authority (TA) not only creates a load bottleneck, but also have key escrow problem since the TA can access all the encrypted files. This opens the door for potential privacy exposure.

Key- Policy Attribute-Based Encryption:

Yu et al. (YWRL) applied key-policy ABE to secure outsourced data, where there will be single data owner who can encrypt his data and share with multiple authorized users by providing decryption keys to them. Key contains the attribute-based access privileges. Yu et al. (YWRL) also propose a method for the data owner to revoke a user efficiently by delegating the updates of affected cipher texts and user secret keys to the server. KP-ABE is a public key cryptography primitive for one-to-many encryption. In KP-ABE, data are associated with attributes that will have the public key component. The encryptor/owner associates the set of attributes to the message by encrypting it with the corresponding public key components. Each user/clients is assigned an access structure which is usually defined as an access tree that contains the data attributes in which the interior nodes of the access tree are threshold gates and leaf nodes are

associated with attributes. User secret key is defined based on the access structure so that the user is able to decrypt a cipher text if and only if the data attributes satisfy his access structure.

Drawbacks:

In KP-ABE, the key update operations can be aggregated over time, YWRL scheme achieves low amortized overhead. Meanwhile in the YWRL scheme, the data owner is also a TA at the same time. It would not be efficient to apply in the PHR system with multiple data owners and users. Since each user would receive many keys from multiple owners, even if the cryptographic keys contain the same set of attributes.

Cipher text Policy Attribute-Based Encryption:

In the Cipher text Policy Attribute-based Encryption (CPABE), the private key is distributed to users by a trusted central issuer only once. The keys are identified with a set of descriptive attributes, and the encryptor specifies an encryption policy using an access tree so that those with private keys which satisfy it can decrypt the cipher text. Cipher text Policy Attribute-based Encryption (CP-ABE) can be widely applied to realize access control in many applications including medical systems and education systems X.Liang aimed at developing the CP-ABE scheme with efficient revocation. Designing a revocation mechanism for CP-ABE is not a simple task while considering the following aspects: first, system manager only associates user secret keys with different sets of attributes instead of individual characteristics ; second, users' individuality are taken place by several common attributes, and thus revocation on attributes or attribute sets cannot accurately exclude the users with misbehaviors; third, the system must be secure against collusion attack from revoked users even though they share some common attributes with non-revoked users.

In CP-ABE, the data owner encrypts the data according to an access control policy P defined over a set of attributes, and the receiving end can decrypt the encrypted data only if his secret key associated with a set of attributes satisfies P . For example, suppose Alice encrypts her data according to an access policy $P = (a1 \text{ AND } a2) \text{ OR } a3$. Bob can decrypt the encrypted data only if his secret key is associated with a set of attributes that satisfy the access policy. To satisfy P , Bob must have a secret key associated with at least one from the following attribute sets: $(a1, a2)$, $(a3)$ or $(a1, a2, a3)$. In general, CP-ABE scheme consists of the following four algorithms:

- Setup algorithm $(MK, PK) \leftarrow \text{Setup}(1k)$: It is run by the trusted authority or the security administrator. The setup algorithm takes as input a security parameter k and outputs a master secret key MK and a master public key PK .

- Key Generation algorithm (SK) \leftarrow Key Gen (MK, ω): It is run by the trusted authority, and takes as input a set of attributes ω and MK. The algorithm outputs a user secret key SK associated with the attribute set ω .
- Encrypt algorithm (CT) \leftarrow Encrypt (m, PK, P): It is run by the encryptor. The input of the algorithm is a message m, a master public key PK and an access control policy P, the output of the algorithm is a ciphertext CT encrypted under the access control policy P.
- Decrypt algorithm (m) \leftarrow Decrypt (CT, SK): It is run by the decryptor. The input of the algorithm is a ciphertext CT to be decrypted and a user secret key SK. The output of the algorithm is a message m, if the attribute set of the secret key satisfies the access policy P under which the message was encrypted, or an error message if the attribute set of the secret key does not satisfies the access policy P under which the message was encrypted.

Drawbacks:

CP-ABE systems can support only uncontrolled delegation [12] (the delegator cannot prevent the delegate to delegate further his authority), or use a system where attributes are valid within a specific time frame (there is no way to revoke an attribute before the expiration date).

Multiple-Authority Attribute- Based Encryption:

Chase and Chow proposed a multiple-authority ABE (CC MA-ABE) solution in which there will be multiple TAs, each governs a different subset of the system's users' attributes and generate user secret keys collectively. A user needs to obtain one part of his key from each TA. Chase and Chow scheme prevents collusion among at most N-2 TA's. Lin et al. recently proposed a different approach for building a multi-authority ABE scheme without a central authority. However, their construction requires designers to fix a constant m for the system, which directly determines efficiency. The resulting construction is such that any group of m + 1 colluding users will be able to break security of the encryption.

Drawbacks:

In CC MA-ABE access policy is embedded in user keys not in the cipher text and it is non-intuitive approach. Already issued private keys can never be modified until the whole system

crashes and cannot be able to distinguish the same user in different transaction. Data access right could be given based on user's identities and lack of expressibility in access policy.

Distributed Attribute -Based Encryption

Sascha Muller introduced a concept of Distributed Attribute-Based Encryption (DABE). In DABE, there will be an arbitrary number of parties to maintain attributes and their corresponding secret keys. In CP-ABE schemes, where all secret keys are distributed by one central trusted party, there are three different types of entities in a DABE scheme: a master, attribute authorities and users. The master is responsible for the distribution of secret user keys. However, master is not involved in the creation of secret attribute keys. Attribute authorities are responsible to verify whether a user is eligible of a specific attribute; in this case they distribute a secret attribute key to the user. An attribute authority generates a public attribute key for each attribute it maintains; this public key will be available to all the users. Eligible users receive a personalized secret attribute key over an authenticated and trusted channel. Users can encrypt and decrypt messages. To encrypt a message, user should formulate the access policy in Disjunctive Normal Form (DNF). To decrypt a cipher text, a user needs at least access to some set of attributes (and their associated secret keys) which satisfies the access policy.

3. REQUIREMENT ELICITATION AND ANALYSIS

3.1 Requirement Analysis

- **Microsoft Visual Studio**

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop console and graphical user interface applications along with Windows forms or WPF applications, web sites, web applications, and web services in both native code together with managed code for all platforms supported by Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework and Microsoft Silver light.

Visual Studio includes a code editor supporting IntelliSense as well as code refactoring. The integrated debugger works both as a source-level debugger and a machine-level debugger. Other built-in tools include a forms designer for building GUI applications, web designer, class designer, and database schema designer. It accepts plug-ins that enhance the functionality at almost every level—including adding support for source-control systems (like Subversion and Visual SourceSafe) and adding new toolsets like editors and visual designers for domain-specific languages or toolsets for other aspects of the software development lifecycle.

Visual Studio supports different programming languages by means of language services, which allow the code editor and debugger to support (to varying degrees) nearly any programming language, provided a language-specific service exists. Built-in languages include C/C++ (via Visual C++), VB.NET (via Visual Basic .NET), C# (via Visual C#), and F# (as of Visual Studio 2010). Support for other languages such as Python, and Ruby among others is available via language services installed separately. It also supports XML/XSLT, HTML/XHTML, JavaScript and CSS. Individual language-specific versions of Visual Studio also exist which provide more limited language services to the user: Microsoft Visual Basic, Visual J#, Visual C#, and Visual C++.

The IDE provides three services: SVsSolution, which provides the ability to enumerate projects and solutions; SVsUIShell, which provides windowing and UI functionality (including tabs, toolbars and tool windows); and SVsShell, which deals with registration of VSPackages. In addition, the IDE is also responsible for coordinating and enabling communication between services.[8] All editors, designers, project types and other tools are implemented as VSPackages. Visual Studio uses COM to access the VSPackages. The Visual Studio SDK also includes the Managed Package Framework (MPF), which is a set of managed wrappers around the COM-interfaces that allow the Packages to be written in any CLI compliant language. However, MPF

does not provide all the functionality exposed by the Visual Studio COM interfaces. The services can then be consumed for creation of other packages, which add functionality to the Visual Studio IDE.

- **.Net Framework**

Microsoft .NET Framework is a software framework developed by Microsoft that runs primarily on Microsoft Windows. It includes a large library and provides language interoperability (each language can use code written in other languages) across several programming languages. Programs written for .NET Framework execute in a software environment (as contrasted to hardware environment), known as the Common Language Runtime (CLR), an application virtual machine that provides services such as security, memory management, and exception handling. The class library and the CLR together constitute .NET Framework.

Microsoft .NET Framework's Base Class Library provides user interface, data access, database connectivity, cryptography, web application development, numeric algorithms, and network communications. Programmers produce software by combining their own source code with .NET Framework and other libraries. .NET Framework is intended to be used by most new applications created for the Windows platform. Microsoft also produces an integrated development environment largely for .NET software called Visual Studio.

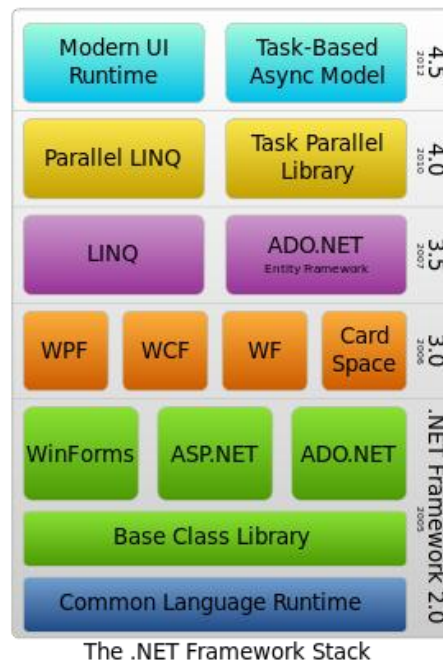


Figure 1

.NET Framework 3.5

.NET Framework version 3.5 uses Common Language Runtime (CLR) 2.0, that is, the same version as .NET Framework version 2.0. In addition, .NET Framework 3.5 also installs .NET Framework 2.0 SP1 and 3.0 SP1 (with the later 3.5 SP1 instead installing 2.0 SP2 and 3.0 SP2), which adds some methods and properties to the BCL classes in version 2.0 which are required for version 3.5 features such as Language Integrated Query (LINQ). These changes do not affect applications written for version 2.0, however.

.NET Compact Framework 3.5 support additional features on Windows Mobile and Windows Embedded CE devices.

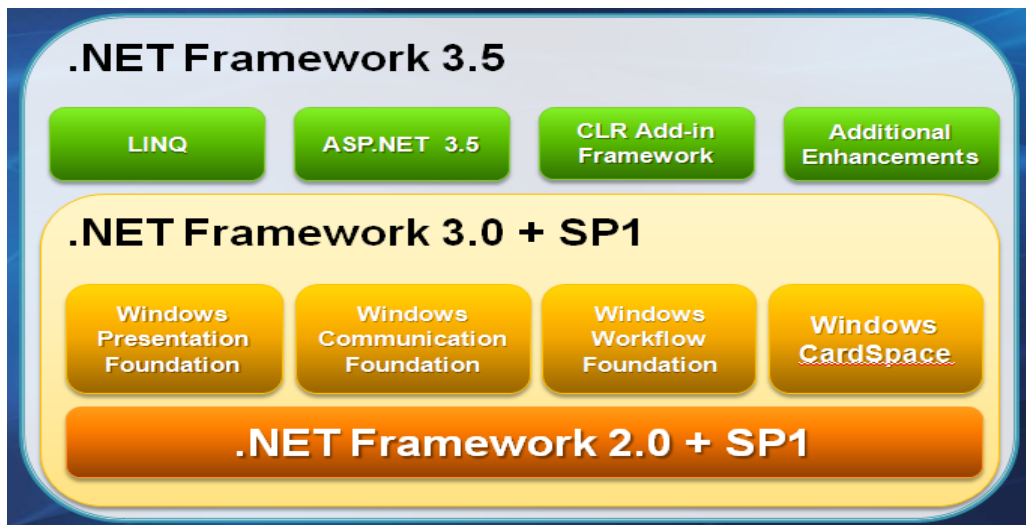


Figure 2

- **MS-ACCESS DATABASE**

Microsoft Access is a database management system from Microsoft that combines the relational Microsoft Jet Database Engine with a graphical user interface and software-development tools. It is a member of the Microsoft Office suite of applications, included in the Professional and higher editions or sold separately.

Microsoft Access stores data in its own format based on the Access Jet Database Engine. It can also import or link directly to data stored in other applications and databases.

Software developers and data architects can use Microsoft Access to develop application software, and "power users" can use it to build software applications. Like other Office applications, Access is supported by Visual Basic for Applications, an object-oriented programming language that can reference a variety of objects including DAO (Data

Access Objects), ActiveX Data Objects, and many other ActiveX components. Visual objects used in forms and reports expose their methods and properties in the VBA programming environment, and VBA code modules may declare and call Windows operating-system functions.

3.2 Top level Use Case and Sequence Diagrams

- **Use case diagrams**

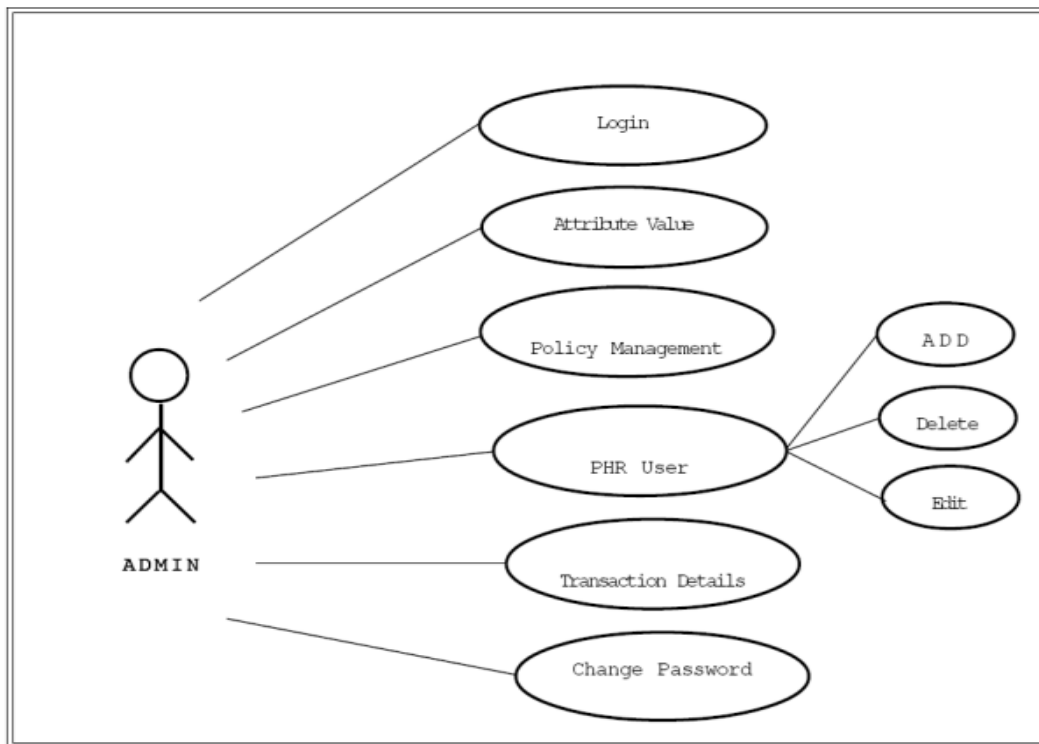


Figure 3 - Admin session

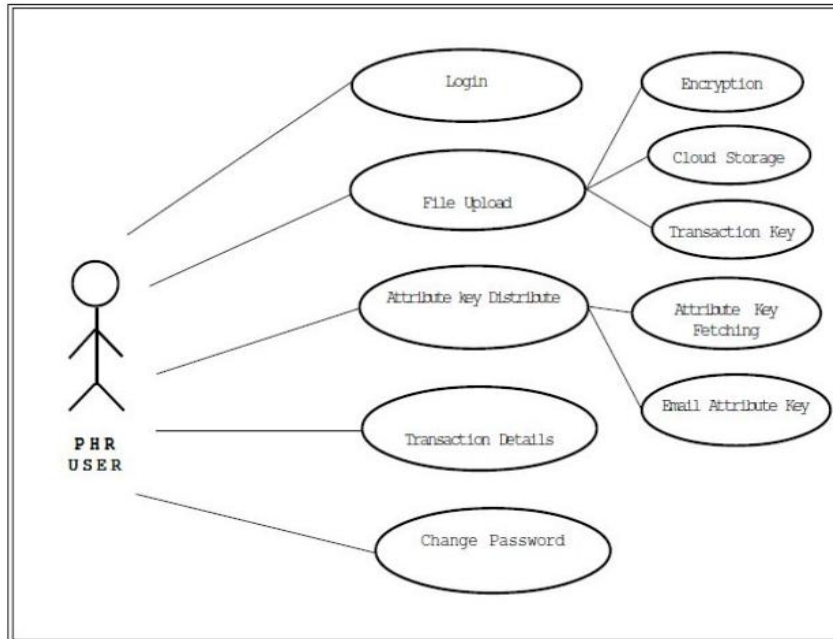


Figure 4 - PHR Owner Session

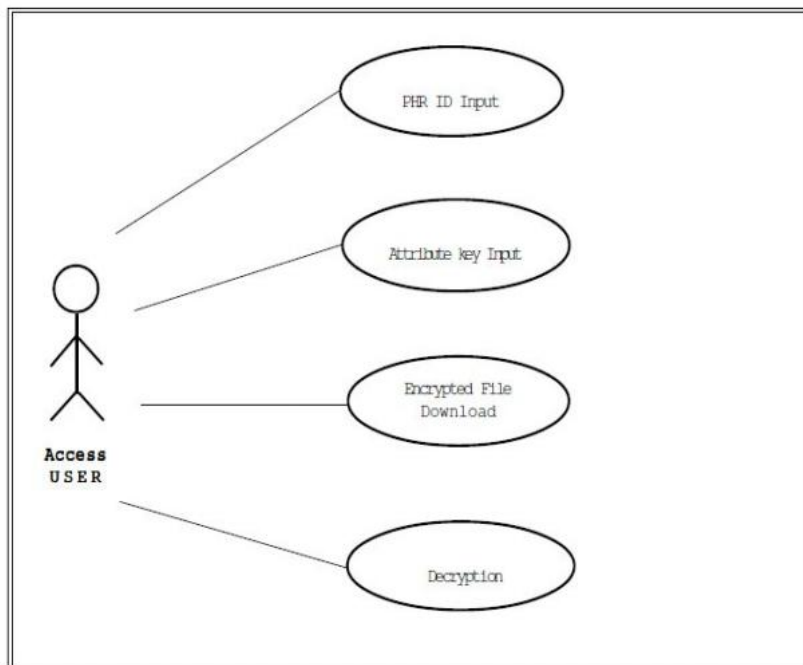


Figure 5- Data access user session

Sequence Diagram

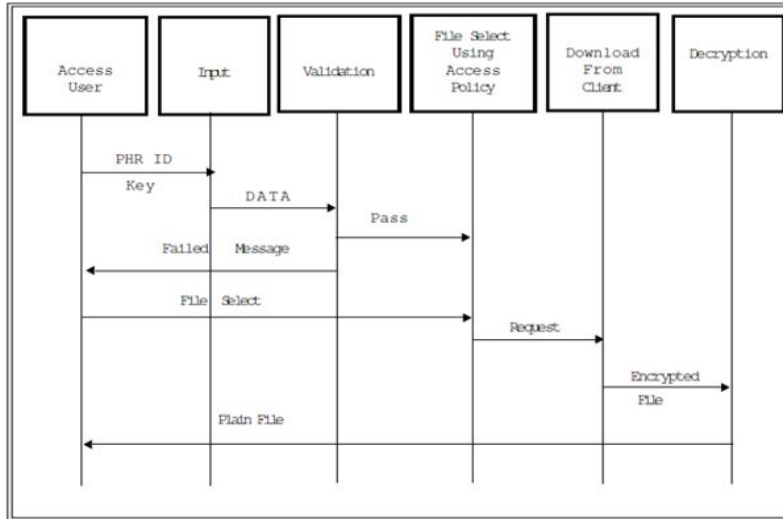


Figure 6

3.3 Design specification ER DIAGRAM

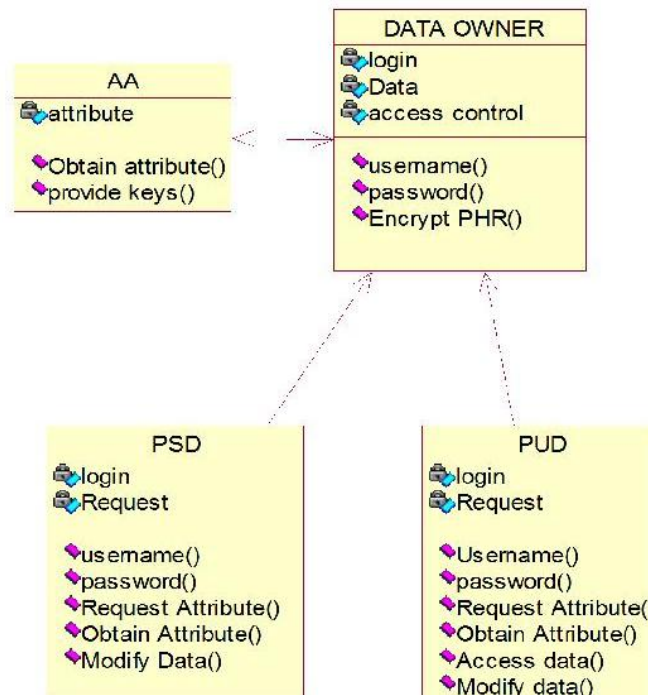


Figure 7

4. Project Plan and Implementation Details

4.1 Sharing of PHR

The main goal of PHR system is to provide patient-centric PHR access with high security and efficient key management. This is achieved by dividing the system into multiple security domains (namely, public domains (PUDs) and personal domains (PSDs)) according to different users' data access requirements. The PUDs consist of users based on their professional roles, such as doctors, nurses and medical researchers. In practice, a PUD can be hospital, government or insurance sector. In case of PSD, its users are personally associated with a data owner (such as family members or close friends) and they access the PHRs based on access rights assigned by the owner. In both types of security domains, we utilize ABE to realize cryptographically enforced, patient-centric PHR access. Especially, in a PUD multi-authority ABE is used, in which there are multiple "attribute authorities" (AAs), each governing a disjoint subset of attributes. Role attributes are defined for PUDs that represent the professional role. The PUD users obtain their attribute-based keys from the respective AAs, without the need to interact directly with the owners. To control access from PUD users, owners can specify role-based access policies for their PHR files. Since the PUDs consist of a large number of users, there is a great reduction in the key management overhead for both the owners and users.

The data owners (e.g., patients) are the trusted authorities of their own PSD and provide the secret keys and access rights to the users in their PSD. Since the users are personally known by the PHR owner, to realize patient-centric access, the owner is at the best position to grant user access rights. For PSD, data attributes are defined which refer to the intrinsic properties of the PHR data, such as the category of a PHR file. For the purpose of PSD access, each PHR file is labeled with its data attributes, while the key size is only linear with the number of file categories a user can access. Since the number of users in a PSD is often small, it reduces the burden for the owner. When encrypting the data for PSD, all that the owner needs to know is the intrinsic data properties. The multi-domain approach provides best models for different user types and access requirements in a PHR system. The use of ABE makes the encrypted PHRs self-protective, i.e., they can be accessed by only authorized users even when storing on a server, and when the owner is not online.

4.2 Implementation of PHR System

The PHR system can be implemented using the hardware and software mentioned in above sections. The system first defines attributes shared by the users in PSD and PUD, such as “personal”, “medical history”, “current exams”, “insurance”, and “sensitive”.

An emergency attribute is also defined for emergency access. Each PHR owner’s client application generates its corresponding master keys. When using the PHR service for the first time, a PHR owner can specify the access privilege of a data reader in PSD and then the application generate and distribute corresponding key to the latter through the mail.

For the PUDs, the system defines role attributes, and a user in a PUD obtains secret key from AAs. The secrete key is sent to the hospital authority or other authorities included in the system through mail. Each user in the PUD obtains attributes from the AAs. In practice, there exist multiple AAs each governing a different subset of role attributes. For example, hospital staffs shall have a different AA from pharmacy specialists.

4.3 Algorithm for the PHR system

1. PHR owner will input the attributes he/she wants to share.
2. Encryption of input attributes is done using 3-DES algorithm.
3. Encrypted attributes are uploaded and distribution of attribute based key.
4. User or data access member input the attribute-based key.
5. Verification of the key for its validity.
6. If the key is invalid, attributes will not decrypt.
7. If the key is valid, attributes will be decrypted.
8. Decrypted attributes will be downloaded into the user’s local system.

Flowchart-for PHR system

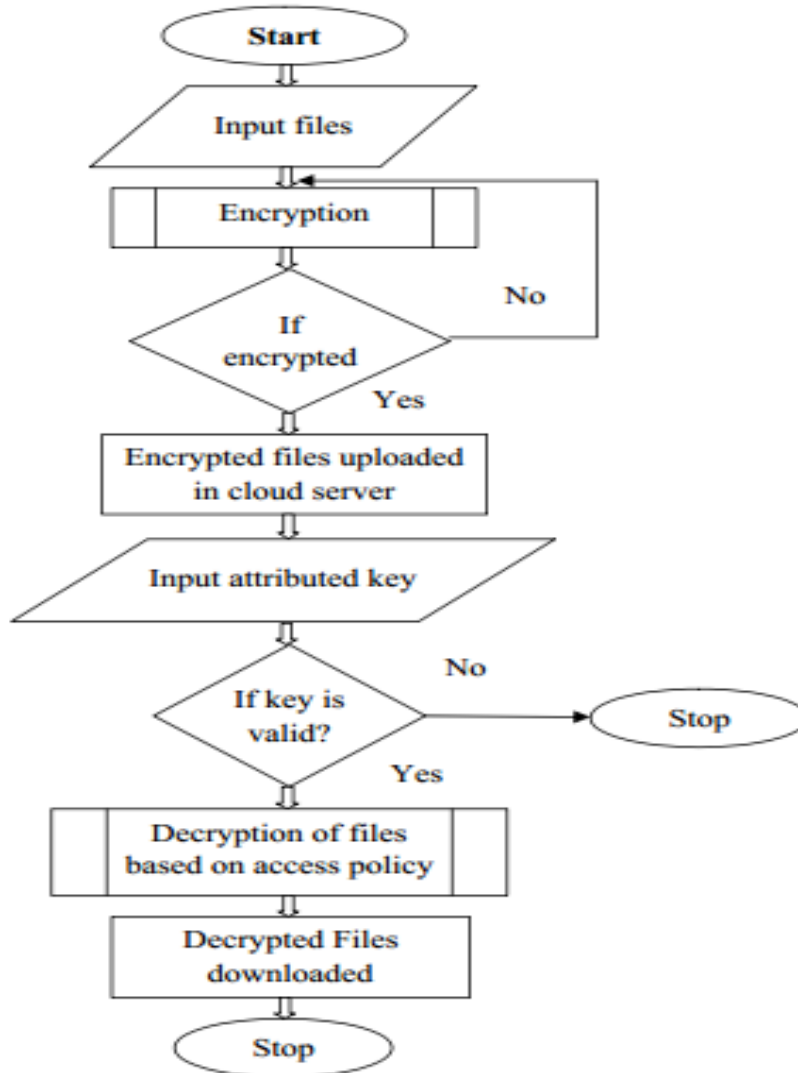


Figure 8

4.4 Module Description

The goal of patient-centric privacy is often in conflict with scalability in a PHR system. The authorized users may either need to access the PHR for personal use or professional purposes. Examples of the former are family member and friends, while the latter can be medical doctors, pharmacists, and researchers, etc. We refer to the two categories of users as personal and professional users, respectively.

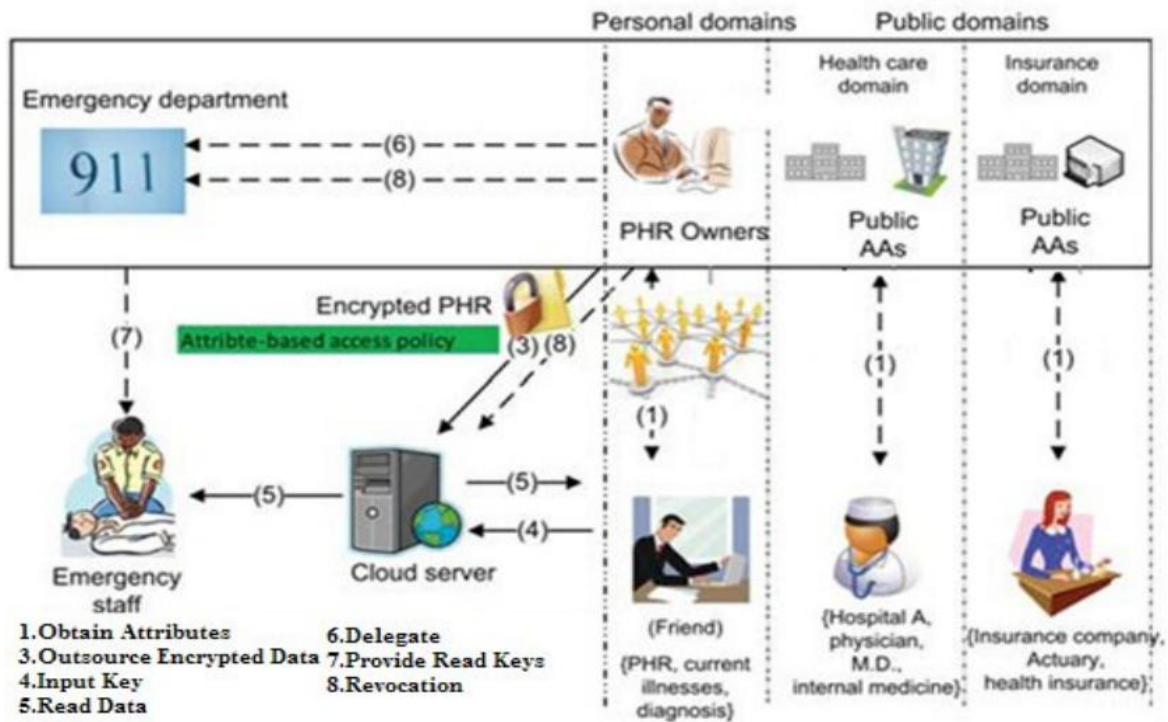


Figure 9

For the PUDs, the system defines role attributes, and a reader in a PUD obtains secret key from AAs. MA-ABE is used to encrypt the data.

PHR Encryption and Access: The owners upload ABE-encrypted PHR files to the server. Each owner's PHR file is encrypted both under a certain fine-grained and role-based access policy for users from the PUD to access, and under a selected set of data attributes that allows access from users in the PSD. Only authorized users can decrypt the PHR files, excluding the server. The data readers download PHR files from the server, and they can decrypt the files only if they have suitable attribute-based keys.

User Revocation: Here we consider revocation of a data reader or attributes/access privileges. There are several possible cases:

- 1) Revocation of one or more role attributes of a public domain user;
- 2) Revocation of a personal domain user's access privileges;
- 3) Revocation of a personal domain user.

These can be initiated through the PHR owner's client application.

Policy Updates: The PHR owners can update their sharing policy for the existing PHR documents by updating the attributes (or access policy). The supported operations include add/delete/change, which can be done by the server on behalf of the user.

Break-glass: When an emergency happens, the regular access policies may no longer be applicable. To handle this situation, break-glass access is needed to access the victim's PHR. In this system, each owner's PHR's access right is also delegated to an emergency Department. In case of emergency, the emergency staff needs to contact the ED to verify the identity and the emergency situation of the patient, and obtain temporary read keys. After the emergency is over, the patient can revoke the emergent access via the ED.

This system considers the server to be semi-trusted, i.e., honest but curious. That is the server will try to find out the secret information in the stored PHR files as much as possible, but they will honestly follow the protocol in general. On the other hand, some users will also try to access the files beyond their rights. For example, a pharmacy may try to obtain the prescriptions of patients for marketing and boosting its profits. To do so, they may collude with other users, or even with the server.

PHR Owners need upload data on server in such a manner that confidentiality of data and access rights of the data in highest point. There are various types of user allowed to access the data based on policy management as given by table below. This system has three types of users Admin, PHR Owner & Data Access Member. Following are the type of files which the PHR owner wishes to share with various types of users. Below which gives the details about the attributes used.

Sample Files used in this system

- Personal File
- Medical History
- Current Medical Examination
- Insurance Details
- Sensitive Details

Attribute Types in this System

- Friends
- Hospitals

- Insurance
- Emergency

Files/Attribute	PHR Owner	Hospitals	Insurance	Emergency
Personal	YES	NO	NO	YES
Medical History	YES	YES	NO	YES
Current Exams	YES	YES	YES	YES
Insurance	YES	YES	YES	YES
Sensitive	YES	NO	NO	YES

Figure 10 - Access Policy Table

4.5 Block diagram of PHR System

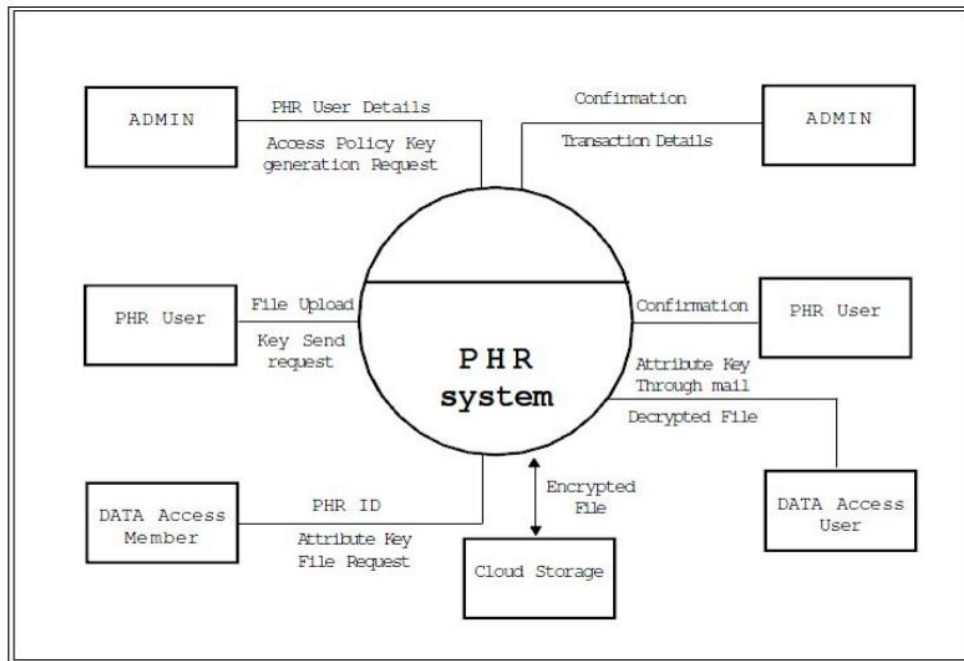


Figure 11

The block diagram of the proposed PHR system is shown above. In this diagram, PHR system is the system in which encryption and decryption of the health records and other files takes place. PHR users or owners upload their files which are encrypted in PHR system and stored in the server. The PHR user will send the attribute based key to the data access users such as hospitals, friends and insurance company using which they will be authorized to access the files according to policy management. Hence the decrypted file will be downloaded into the data access user's system using attribute based key. Different users will be given different keys. Admin will be responsible for managing PHR user details, transaction details, policy management etc.

He can change the access policy any time. PHR user details contain PHR ID, name, city, contact number etc. Transaction details contain the user name, transaction type (uploading or downloading of a file), file name and type (text or image), date and time of transaction.

Deal with Emergency Access

Sometimes medical staffs need to have temporary access to certain parts of the PHR data when an emergency occurs to a patient, who may be unconscious and unable to modify the access policies beforehand. The medical staffs require a temporary authorization (e.g., emergency key) to decrypt those parts of data. This can be easily achieved in this system by allowing each patient delegate their emergency key to an emergency department (ED). At the beginning, each owner defines an “emergency” attribute and includes it into the Personal domain part of the cipher text of each PHR document to allow emergency access. The patient then generates an emergency key and delegates it to the ED who stores it in a database of patient directory. During emergency, the medical staff requests and obtains the corresponding patient's emergency key from the ED and then access the decrypted PHR documents using that key. After recovering from the emergency, the patient can revoke the emergency access by changing the access policy.

4.6 The stepwise implementation algorithm:

1. The trusted authority from the professional domain and the patient from the social domain run the Setup algorithm of their ABE scheme.
2. Next, users from the professional domain get their secret keys related to their attributes they possess from the trusted authority from professional domain.
3. The patient uses a number of healthcare devices and creates measurement data and forwards them to the application hosting device which can be patient's personal computer, mobile phone or any other trusted device.

4. The application hosting device categorizes the measurement data. The hosting device encrypts the data according to an access policy $P=P1 \text{ OR } P2$, which consists from two sub policies P1 and P2. Either P1 or P2 must be satisfied in order to decrypt the cipher text. The first part of the access policy P1 is intended for the social domain, therefore, the patient would be responsible to generate secret keys associated with attributes in P1, and the second part of the policy P2 is intended for the professional domain, therefore Trusted authority from professional domain would be responsible to generate secret keys associated with attributes in P2.

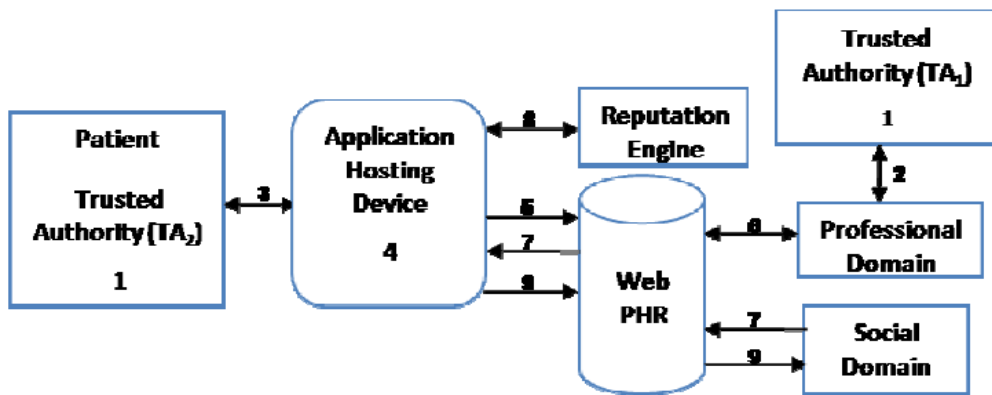


Figure 12

The structure of P1 is as follows:

$$P1 = a^{MD} \text{ OR } a^{DC} \text{ OR } a^{AC}$$

- This implies that in order to access the measurement data, the receiver must have a secret key SKMD (associated with attribute a^{MD}), or a secret key SKDC (associated with attribute a^{DC}), or the secret key SKAC (associated with attribute a^{AC}).
- Where P1 contains attributes related to the resource (In CP-ABE a policy contains attributes which identify the user), in which the attribute a^{MD} identifies the measured data MD, the attribute a^{DC} identifies the data category DC, and the attribute a^{AC} identifies the administrator category AC

The structure of P2 is dynamic and depends on patient preferences and contains attributes associated with users from the professional domain.

5. The encrypted data is sent to the web PHR repository.
6. From the stored data, eventually a patient could see health trends and begin to learn what lifestyle or other behaviors are affecting her glucose levels or blood pressure. When the doctor from the professional domain wants to see patient data, it downloads the encrypted data from the server, and decrypts them locally using her secret key, as shown in step 6th.
7. Next, the patient receives a request from a user from the social domain with whom the patient may have no pre-arranged trust relationship, to see his/her data.
8. Now, the patient makes a decision regarding whether to issue or not the secret key to the requesting user from the social domain. The patient bases his decision on the requester's reputation score generated by the reputation evaluation engine. The reputation evaluation engine may take as input the ratings given by other users, and outputs the reputation of the requester.
9. The patient runs the key generation algorithm to generate the secret key associated with a set of attributes related to the document. The patient could generate different types of secret keys with different decryption power. If the user has high reputation, he will get a secret key with higher decryption power and vice versa. The requesting user uses the secret key to decrypt the encrypted data.

4.7 Encryption and Decryption

A message in its original form (plaintext) is converted (encrypted) into an unintelligible form (cipher text) by a set of procedures known as an encryption algorithm (cipher) and a variable, called a key. The cipher text is transformed (decrypted) back into plaintext using the encryption algorithm and a key. It is a way to enhance the security of a message or file by scrambling the contents so that it can be read only by someone who has the right encryption key to unscramble it.

Data Encryption Standard (DES)

Using Data Encryption Standard (DES) algorithm data is encrypted and decrypted. DES is the block cipher, an algorithm that takes a fixed-length string of plaintext bits and transforms it through a series of complicated operations into another cipher text bit string. In DES, the block size is 64 bits. DES also uses a key to customize the transformation, so that decryption can supposedly only be performed by those who know the particular key used to encrypt. The key

consists of 64 bits; however, only 56 of these are actually used by the algorithm. Eight bits are used solely for checking parity, and are thereafter discarded. Hence the effective key length is 56 bits, and it is always quoted as such. DES uses the two basic techniques of cryptography - confusion and diffusion. At the simplest level, diffusion is achieved through numerous permutations and confusions is achieved through the XOR operation. The mechanism of diffusion seeks to make the relationship between the plaintext and cipher text as complex as possible in order to avoid attempts to deduce the key. On the other hand, confusion seeks to make the relationship between the statistics of the cipher text and the value of the encryption key as complex as possible, again to avoid attempts to discover the key. This is achieved by the use of a complex substitution algorithm.

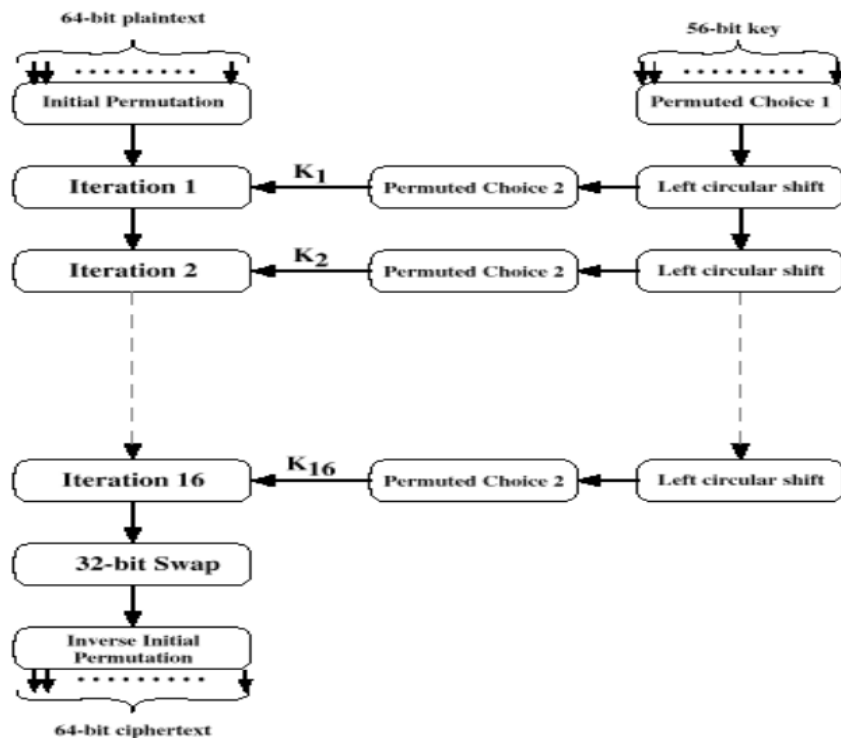


Figure 13

The basic process in enciphering a 64-bit data block and a 56-bit key using the DES consists of:

- An initial permutation (IP)
- 16 rounds of a complex key dependent calculation f
- A final permutation, being the inverse of IP

3-DES (Triple DES)

Triple DES uses a "key bundle" which comprises three DES keys, K_1 , K_2 and K_3 , each of 56 bits. The encryption algorithm is:

$$\text{Cipher text} = E_{K_3}(D_{K_2}(E_{K_1}(\text{plaintext})))$$

i.e., DES encrypts with K_1 , DES *decrypt* with K_2 , then DES encrypt with K_3 .

Decryption is the reverse:

$$\text{Plain text} = D_{K_1}(E_{K_2} D_{K_3}(\text{cipher text}))$$

i.e., decrypt with K_3 , *encrypt* with K_2 , and then decrypt with K_1 .

Each triple encryption encrypts one block of 64 bits of data.

In each case the middle operation is the reverse of the first and last. This improves the strength of the algorithm when using keying option 2, and provides backward compatibility with DES with keying option 3.

Key Options

The standards define three keying options:

- Keying option 1: All three keys are independent.
- Keying option 2: K_1 and K_2 are independent, and $K_3 = K_1$.
- Keying option 3: All three keys are identical, i.e. $K_1 = K_2 = K_3$.

Keying option 1 is the strongest, with $3 \times 56 = 168$ independent key bits.

Keying option 2 provides less security, with $2 \times 56 = 112$ key bits. This option is stronger than simply DES encrypting twice, e.g. with K_1 and K_2 , because it protects against meet-in-the-middle attacks.

Keying option 3 is equivalent to DES, with only 56 key bits. This option provides backward compatibility with DES, because the first and second DES operations cancel out.

Each DES key is nominally stored or transmitted as 8 bytes, each of odd parity, so a key bundle requires 24, 16 or 8 bytes, for keying option 1, 2 or 3 respectively.

5. CODING

Code for Doctor Registration which consists of Encryption and Decryption using triple-DES (3-DES)

```

namespace PersonalHealthRecords
{
    public partial class WebForm2 : System.Web.UI.Page
    {
        private readonly byte[] IVector = new byte[8] { 27, 9, 45, 27, 0, 72,
171, 54 };
        private string Key1 = "";
        protected void Page_Load(object sender, EventArgs e)
        {
            int i=1;
            String str1 = "DOC00";
            TextBox1.Text = str1 + i;
            i++;
            Conn = new OleDbConnection();
            Conn.ConnectionString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\Users\ruchi's\Documents\doctor.accdb";

        }
        OleDbConnection Conn = null;
        OleDbCommand Cmd = null;
        string hour,minute,hour1,minut1,ampm1,ampm3;
        protected void hours_selectedchange(object sender, EventArgs e)
        {
            hour = hours.SelectedItem.Text.ToString();

        }
        public bool TestEmailRegex(string emailAddress)
        {
            string patternLenient = @"^\w+([-+.']\w+)*@\w+([-
.\w+)*\.\w+([-.\w+)*";
            Regex reLenient = new Regex(patternLenient);
            bool isLenientMatch =
reLenient.IsMatch(emailAddress);
            return isLenientMatch;
        }
        protected void Button5_Click(object sender, EventArgs e)
        {
            string abc = Encrypt(RandomUtil.GetRandomString());
            key.Text = abc;
        }
        private string Encrypt(string inputString)
        {

```

```
        byte[] buffer = Encoding.ASCII.GetBytes(inputString);
        TripleDESCryptoServiceProvider tripleDes = new
TripleDESCryptoServiceProvider();
        MD5CryptoServiceProvider MD5 = new MD5CryptoServiceProvider();
        tripleDes.Key =
MD5.ComputeHash(ASCIIEncoding.ASCII.GetBytes(Key1));
        tripleDes.IV = IVector;
        ICryptoTransform ITransform = tripleDes.CreateEncryptor();
        return
Convert.ToBase64String(ITransform.TransformFinalBlock(buffer, 0,
buffer.Length));
    }
    private string Decrypt(string inputString)
    {
        byte[] buffer = Convert.FromBase64String(inputString);
        TripleDESCryptoServiceProvider tripleDes = new
TripleDESCryptoServiceProvider();
        MD5CryptoServiceProvider MD5 = new MD5CryptoServiceProvider();
        tripleDes.Key =
MD5.ComputeHash(ASCIIEncoding.ASCII.GetBytes(Key1));
        tripleDes.IV = IVector;
        ICryptoTransform ITransform = tripleDes.CreateDecryptor();
        return
Encoding.ASCII.GetString(ITransform.TransformFinalBlock(buffer, 0,
buffer.Length));
    }
    static class RandomUtil
    {
        public static string GetRandomString()
        {
            string path = Path.GetRandomFileName();
            path = path.Replace(".", ""); // Remove period.
            return path;
        }
    }
}
```

TripleDESCryptoServiceProvider class is used to encrypt/ decrypt strings which in turn uses 3DES (Triple Data Encryption Standard) algorithm. 3DES algorithm uses three 64-bit long keys to (Encrypt-Decrypt-Encrypt) data.

MD5CryptoServiceProvider class (in System.Security and System.Security.Cryptography namespace) to compute a hash value for encryption/decryption.

MD5.ComputeHash Method computes the hash value for the input data. This member is overloaded.

ICryptoTransform defines the basic operations of cryptographic transformations.

Code for forget Key and then send key to email

```
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
using System.Data.OleDb;
using System.Net.Mail;

namespace PersonalHealthRecords
{
    public partial class ForgorKeyDoctor : System.Web.UI.Page
    {
        OleDbConnection Conn = null;
        OleDbCommand Cmd = null;
        string to;
        protected void Page_Load(object sender, EventArgs e)
        {
            Conn = new OleDbConnection();
            Conn.ConnectionString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\Users\HP\Documents\doctor.accdb";
            to = TextBox2.Text;
        }

        protected void email_send(string strTo, string strSubject, string
strBody)
        {
            try
            {
                string msg = strBody;
                MailMessage Msg = new MailMessage();
                Msg.From = new MailAddress("emailid");
                Msg.To.Add(TextBox2.Text);
                Msg.IsBodyHtml = true;
                Msg.Subject = "Subject";
                Msg.Body = "<html><body>" + msg + "</body></html>";
                Msg.BodyEncoding = System.Text.Encoding.GetEncoding("UTF-8");
                Msg.Priority = System.Net.Mail.MailPriority.High;
                SmtplibClient smtp = new SmtplibClient();
                smtp.Host = "smtp.gmail.com";
                smtp.Port = 587;
            }
        }
    }
}
```

SCALABLE & SECURE SHARING OF PERSONAL HEALTH RECORDS USING ATTRIBUTE BASED ENCRYPTION

```
smtp.Credentials = new System.Net.NetworkCredential("email",
"password");
smtp.EnableSsl = true;
smtp.Send(Msg);
}
catch (Exception ex)
{
    TextBox1.Text = ex.Message;

    if (ex.InnerException != null)
    {
        TextBox1.Text = ex.InnerException.ToString();
    }
    else
    {
        TextBox1.Text = ex.Message;
    }
}
}
string key_generate;
protected void back(object sender, EventArgs e)
{
    Response.Redirect("Login.aspx");
}
protected void home(object sender, EventArgs e)
{
    Response.Redirect("Index.aspx");
}
protected void retrieve(object sender, EventArgs e)
{
    String user = TextBox1.Text.ToString();
    String key = TextBox2.Text;
    try
    {
        OleDbCommand Cmd5 = new OleDbCommand();
        Cmd5.CommandType = CommandType.Text;

        String sql5 = "Select key from doc where docid='" +
TextBox1.Text + "'";
        Cmd5.CommandText = sql5;
        Cmd5.Connection = Conn;

        Conn.Open();
        OleDbDataReader reader1 = Cmd5.ExecuteReader();
        while (reader1.Read())
        {
            key_generate = reader1.GetString(0);
        }
        Conn.Close();
        Cmd = new OleDbCommand();
        Cmd.CommandType = CommandType.Text;
        String sql = "Select docid from doc where docid='" + user +
"' and email ='" + key + "'";
        Cmd.CommandText = sql;
```

```
        Cmd.Connection = Conn;
        Conn.Open();
        Cmd.ExecuteNonQuery();
        OleDbDataAdapter da = new OleDbDataAdapter(sql, Conn);
        DataTable dt = new DataTable();
        da.Fill(dt);
        if (dt.Rows.Count > 0)
        {
            email_send(to, "KEY", key_generate);
            Label3.Visible = true;
            Label3.Text = "Key Sent to your Registered Email ID...";
        }
        else
        {
            Label4.Visible = true;
            Label4.Text = "Not correct... Retry...";
        }
    }
    catch (OleDbException ex)
    {
        Label3.Visible = true;
        Label3.Text = ex.Message;
    }
    finally
    {
        Conn.Close();
    }
}
}
```

Code to show attribute based encryption in phrloggedin

```
public partial class phrownerloggedin : System.Web.UI.Page
{
    string x;
    OleDbConnection Conn = null;
    OleDbConnection Conn1 = null;
    OleDbConnection Conn2 = null;
    OleDbConnection Conn3 = null;
    OleDbConnection Conn5 = null;
    OleDbConnection Conn6 = null;
    OleDbConnection conn = null;
    OleDbCommand cmd = null;
    OleDbCommand Cmd1 = null;
    OleDbConnection Conn4 = null;
    OleDbCommand Cmd4 = null;
    OleDbCommand Cmd6 = null;
    string y, z, a;
    private readonly byte[] IVector = new byte[8] { 27, 9, 45, 27, 0, 72,
171, 54 };
    private string Key1 = "";
```

SCALABLE & SECURE SHARING OF PERSONAL HEALTH RECORDS USING ATTRIBUTE BASED ENCRYPTION

```
protected void Page_Load(object sender, EventArgs e)
{
    x = (string) (Session["Patient ID"]);
    TextBox1.Text = x;
    start1();
    start2();
}
protected void start1()
{
    x = (string) (Session["Patient ID"]);
    conn = new OleDbConnection();
    conn.ConnectionString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\Users\HP\Documents\doctor_details.accdb";
    cmd = new OleDbCommand();
    cmd.CommandType = CommandType.Text;
    String sql = "Select patid from doctor_encrypt where patid='" + x
+ "'";
    cmd.CommandText = sql;
    cmd.Connection = conn;
    conn.Open();
    cmd.ExecuteNonQuery();
    OleDbDataAdapter da = new OleDbDataAdapter(sql, conn);
    DataTable dt = new DataTable();
    da.Fill(dt);
    if (dt.Rows.Count > 0)
    {
        Label3.Visible = true;
        Label3.Text = "User has already viewed his details";
    }
    else
    {
        encryption_doctor();
    }
}
protected void start2()
{
    x = (string) (Session["Patient ID"]);
    conn = new OleDbConnection();
    conn.ConnectionString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\Users\HP\Documents\patient_details.accdb";
    cmd = new OleDbCommand();
    cmd.CommandType = CommandType.Text;
    String sql = "Select patid from patient where patid='" + x + "'";
    cmd.CommandText = sql;
    cmd.Connection = conn;
    conn.Open();
    cmd.ExecuteNonQuery();
    OleDbDataAdapter da = new OleDbDataAdapter(sql, conn);
    DataTable dt = new DataTable();
    da.Fill(dt);
    if (dt.Rows.Count > 0)
    {
        Label3.Visible = true;
```

```
        Label3.Text = "User has already viewed his details";
    }
    else
    {
        encryption_patient();
    }
}
protected void decryption_patient()
{
    x = (string) (Session["Patient ID"]);
    Conn5 = new OleDbConnection();
    Conn5.ConnectionString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\Users\HP\Documents\patient_details.accdb";
    OleDbCommand Cmd5 = new OleDbCommand("Select
patid,disease,pathistory from patient where patid='" + x + "'", Conn5);
    Cmd5.Connection = Conn5;
    Conn5.Open();
    OleDbDataReader reader = Cmd5.ExecuteReader();
    while (reader.Read())
    {
        y = reader.GetString(0);
        z = reader.GetString(1);
        a = reader.GetString(2);
    }
    String d = Decrypt(z);
    String h = Decrypt(a);
    Conn6 = new OleDbConnection();
    Conn6.ConnectionString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\Users\HP\Documents\patient_details1.accdb";
    Cmd6 = new OleDbCommand();
    Cmd6.CommandType = CommandType.Text;
    Cmd6.Connection = Conn6;
    Cmd6.CommandText = "insert into patient values('" + x + "','" + d
+ "','" + h + "')";
    Conn6.Open();
    int result = Cmd6.ExecuteNonQuery();
}
protected void decryption_doctor()
{
    x = (string) (Session["Patient ID"]);
    Conn5 = new OleDbConnection();
    Conn5.ConnectionString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\Users\HP\Documents\doctor_details.accdb";
    OleDbCommand Cmd5 = new OleDbCommand("Select
patid,prescription,solution from doctor_encrypt where patid='" + x + "'",
Conn5);
    Cmd5.Connection = Conn5;
    Conn5.Open();
    OleDbDataReader reader = Cmd5.ExecuteReader();
    while (reader.Read())
    {
        y = reader.GetString(0);
        z = reader.GetString(1);
    }
}
```

```
        a = reader.GetString(2);
    }
    String d = Decrypt(z);
    String h = Decrypt(a);
    Conn6 = new OleDbConnection();
    Conn6.ConnectionString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\Users\HP\Documents\doctor_details1.accdb";
    Cmd6 = new OleDbCommand();
    Cmd6.CommandType = CommandType.Text;
    Cmd6.Connection = Conn6;
    Cmd6.CommandText = "insert into doctor_details1 values('" + x +
"', '" + d + "', '" + h + "')";
    Conn6.Open();
    int result = Cmd6.ExecuteNonQuery();
}
protected void Button1_Click(object sender, EventArgs e)
{
    GridView2.Visible = true;
    GridView1.Visible = true;
}
protected void encryption_patient()
{
    x = (string) (Session["Patient ID"]);
    Conn = new OleDbConnection();
    Conn.ConnectionString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\Users\HP\Documents\patient.accdb";
    OleDbCommand Cmd = new OleDbCommand("Select
patid,disease,patienthistory from pat where patid='" + x + "'", Conn);
    Cmd.Connection = Conn;
    Conn.Open();
    OleDbDataReader reader = Cmd.ExecuteReader();
    while (reader.Read())
    {
        y = reader.GetString(0);
        z = reader.GetString(1);
        a = reader.GetString(2);
    }
    string d = Encrypt(z);
    string h = Encrypt(a);
    Conn1 = new OleDbConnection();
    Conn1.ConnectionString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\Users\HP\Documents\patient_details.accdb";
    Cmd1 = new OleDbCommand();
    Cmd1.CommandType = CommandType.Text;
    Cmd1.Connection = Conn1;
    Cmd1.CommandText = "insert into patient values('" + x + "', '" + d
+ "', '" + h + "')";
    Conn1.Open();
    int result = Cmd1.ExecuteNonQuery();
}
protected void encryption_doctor()
{
    x = (string) (Session["Patient ID"]);
```

SCALABLE & SECURE SHARING OF PERSONAL HEALTH RECORDS USING ATTRIBUTE BASED ENCRYPTION

```
        Conn2 = new OleDbConnection();
        Conn2.ConnectionString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\Users\HP\Documents\prescription.accdb";
        OleDbCommand Cmd2 = new OleDbCommand("Select patid,prescription
from doctor_prescribed where patid='" + x + "'", Conn2);
        Cmd2.Connection = Conn2;
        Conn2.Open();
        OleDbDataReader reader = Cmd2.ExecuteReader();
        while (reader.Read())
        {
            y = reader.GetString(0);
            z = reader.GetString(1);
        }
        Conn2.Close();
        Conn3 = new OleDbConnection();
        Conn3.ConnectionString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\Users\HP\Documents\surgerydetails.accdb";
        OleDbCommand Cmd3 = new OleDbCommand("Select patid,solution from
surgerydetails where patid='" + x + "'", Conn3);
        Cmd3.Connection = Conn3;
        Conn3.Open();
        OleDbDataReader reader1 = Cmd3.ExecuteReader();
        while (reader1.Read())
        {
            a = reader1.GetString(1);
        }
        string d = Encrypt(z);
        string h = Encrypt(a);
        Conn4 = new OleDbConnection();
        Conn4.ConnectionString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\Users\HP\Documents\doctor_details.accdb";
        Cmd4 = new OleDbCommand();
        Cmd4.CommandType = CommandType.Text;
        Cmd4.Connection = Conn4;
        Cmd4.CommandText = "insert into doctor_encrypt values('" + x +
"' ,'" + d + "' ,'" + h + "')";
        Conn4.Open();
        int result = Cmd4.ExecuteNonQuery();
    }
    private string Encrypt(string inputString)
    {
        byte[] buffer = Encoding.ASCII.GetBytes(inputString);
        TripleDESCryptoServiceProvider tripleDes = new
TripleDESCryptoServiceProvider();
        MD5CryptoServiceProvider MD5 = new MD5CryptoServiceProvider();
        tripleDes.Key =
MD5.ComputeHash(ASCIIEncoding.ASCII.GetBytes(Key1));
        tripleDes.IV = IVector;
        ICryptoTransform ITransform = tripleDes.CreateEncryptor();
        return
Convert.ToBase64String(ITransform.TransformFinalBlock(buffer, 0,
buffer.Length));
    }
}
```

SCALABLE & SECURE SHARING OF PERSONAL HEALTH RECORDS USING ATTRIBUTE BASED ENCRYPTION

```
private string Decrypt(string inputString)
{
    byte[] buffer = Convert.FromBase64String(inputString);
    TripleDESCryptoServiceProvider tripleDes = new
TripleDESCryptoServiceProvider();
    MD5CryptoServiceProvider MD5 = new MD5CryptoServiceProvider();
    tripleDes.Key =
MD5.ComputeHash(ASCIIEncoding.ASCII.GetBytes(Key1));
    tripleDes.IV = IVector;
    ICryptoTransform ITransform = tripleDes.CreateDecryptor();
    return
Encoding.ASCII.GetString(ITransform.TransformFinalBlock(buffer, 0,
buffer.Length));
}
protected void Button2_Click(object sender, EventArgs e)
{
    if (TextBox2.Text == "")
    {
        Label3.Visible = true;
        Label3.Text = "Please Enter the key";
    }
    else
    {
        Conn1 = new OleDbConnection();
        Conn1.ConnectionString =
@"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\Users\HP\Documents\patient.accdb";
        string input_key = TextBox2.Text;
        x = (string)(Session["Patient ID"]);
        Cmd1 = new OleDbCommand();
        Cmd1.CommandType = CommandType.Text;
        String sql = "Select patid,key from pat where patid='" + x +
" ' and key='" + input_key + "'";
        Cmd1.CommandText = sql;
        Cmd1.Connection = Conn1;
        Conn1.Open();
        Cmd1.ExecuteNonQuery();
        OleDbDataAdapter da = new OleDbDataAdapter(sql, Conn1);
        DataTable dt = new DataTable();
        da.Fill(dt);
        if (dt.Rows.Count > 0)
        {
            Button3.Visible = true;
            Label3.Text = "Valid Key...";
            decryption_patient();
            decryption_doctor();
        }
    }
}
}
```


6. ADVANTAGES AND APPLICATIONS

Advantages

- There is policy management for file access, data access member can able to access the files which they have rights set by the policy.
- Files stored are in encrypted form and there is no chance of others to view the file content.
- There is a structured way to access the file for personal & professional purpose through attribute policies and attribute based encryption and decryption.

Applications

- Online personal health record (PHR) enables patients to manage their own medical records in a centralized way, which greatly facilitates the storage, access and sharing of personal health data.
- Easy ways for health providers, insurance companies, hospitals and patients to all access the information.
- Comprehensive solutions for parents who are managing their children's records.
- Eliminate communication barriers and allows documentation flow between patients and clinicians in a timely fashion can save time consumed by face-to-face meetings and telephone communication.
- In the case of an emergency a PHR can quickly provide critical information to proper diagnosis or treatment.

7. OUTPUT

Snapshot 1: This is the home page of the PHR System.



Snapshot 2: On home page, when register button is clicked, then the Register page is loaded in which we have registration for different users like doctor, patient, insurance agent and pathologist Registration.



Snapshot 3: When doctor registration is clicked then the page for registering a new doctor is loaded where a doctor registers itself with a unique doctor id. Then he will ask for the key using Get Key button and after copying it, then the record is added successfully.

DOCTOR REGISTRATION

Doctor Id: DOC001 Date Of Birth: 7/31/1992
Doctor Name: Manas Kapoor Qualification: MBBS
Gender: Male Female Joining Date: 10/31/2013
Marital Status: Married Unmarried Timings: 05:00 PM TO 10:00 PM
Phone No: 8679363058
Mobile: 9811263520
Email: mansaskapoor31@gmail.com
Address: H15B-52A Parmar Bhawa
Specialization: General Physician

7ZTjP009b4JVPcyyqJdQ== Get Key Add Cancel Home Back

Snapshot 4: When patient registration is clicked then the page for registering a new patient in the PHR System is loaded where a patient registers itself with a unique patient id. Then he will ask for the key using Get Key button and after copying it, then the record is added successfully.

PATIENT REGISTRATION

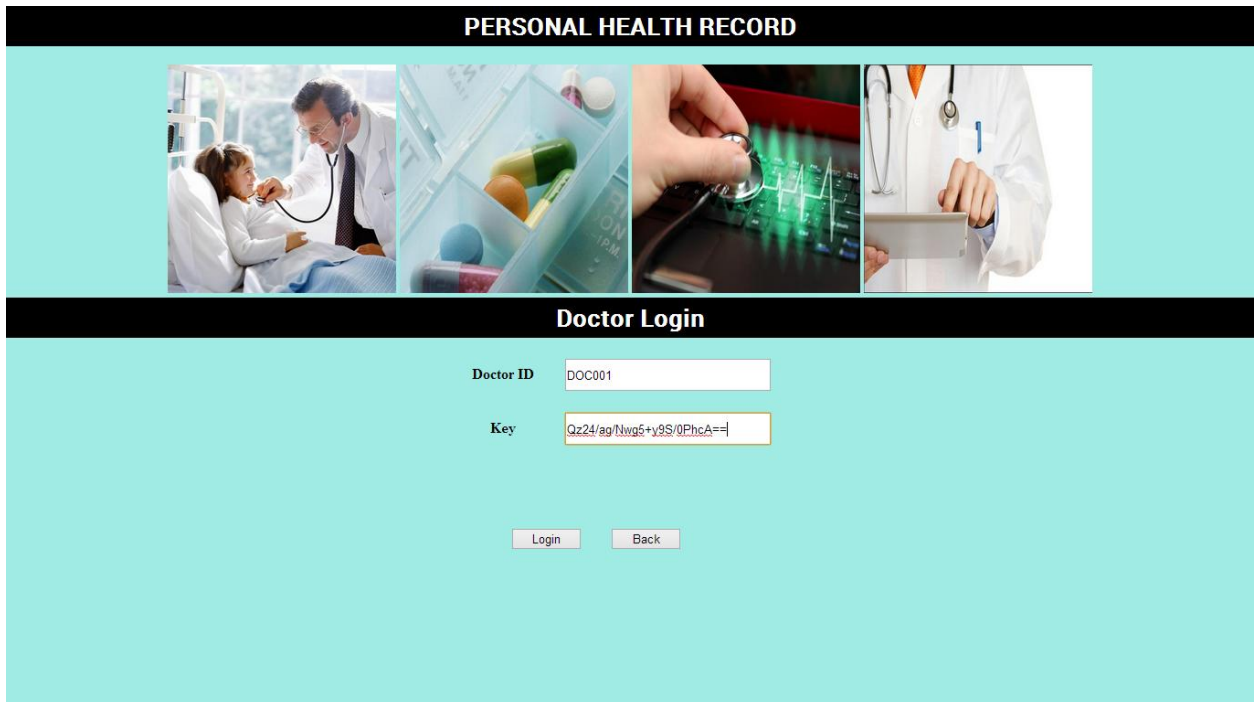
Patient ID: PAT001
Patient Name: Ruchi Aggarwal
Date Of Birth: 9/1/1992
Gender: Male Female
Marital Status: Married Unmarried
Phone No: 9736553508
Email: ruchia75@gmail.com
Address: IP Extension, Delhi
Disease: Measels
Patient History: No past history

5f3Qe2NpjwkiZ6Nj3yV6Q== Get Key Submit Cancel Home Back

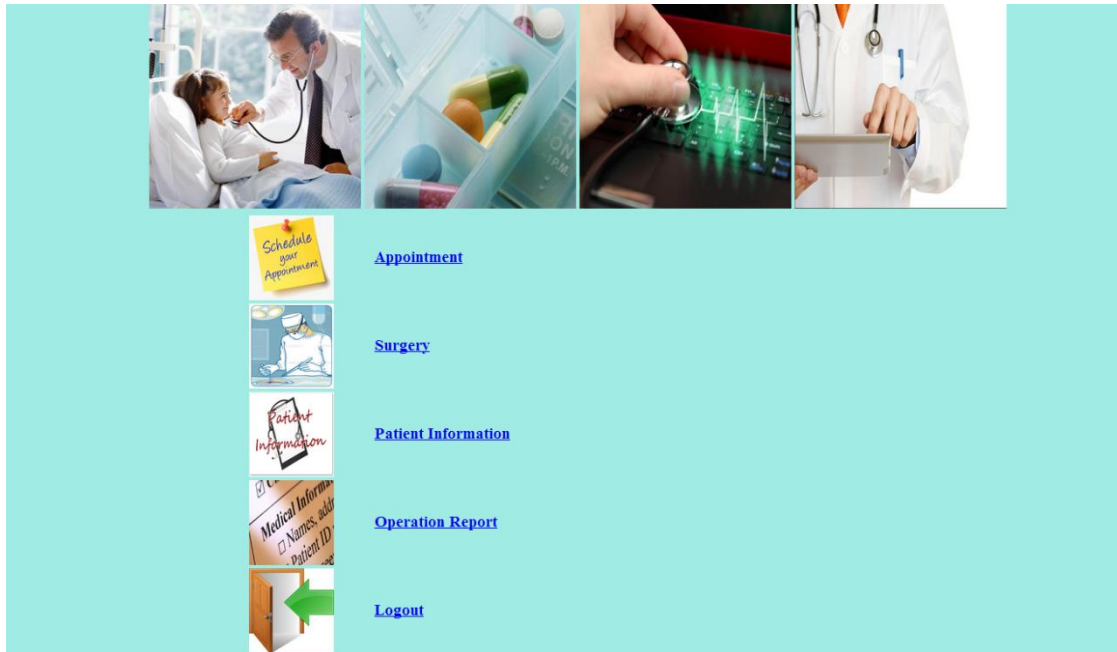
Snapshot 5: After registration, the user will click on access control, where there is a link for patient as well as doctor login.



Snapshot 6: Now the Doctor will Login using its ID and the Encrypted Key which he/she copies at the registration time.



Snapshot 7: Now the doctor login is clicked and the home page for doctor who has been already registered is displayed where there are links for scheduling an appointment for the patient, surgery details of a particular patient, patient Information required by the doctor, operation report and logout.



Snapshot 8: Here the doctor can confirm or cancel the appointment of the patient which he can view in his part.



Snapshot 9: The doctor clicks on Surgery and update the patient record.

PERSONAL HEALTH RECORD

Doctor Surgery

Patient ID: PAT001

Name: Ruchi Aggarwal

Problem: Measels

Solution:

Surgery Date: 12/3/2013

Surgery Timings: 09:00 PM TO 09:30 PM

Update Back Logout

Snapshot 10: To get any patients information, the doctor will click on patient info and that record will be displayed.

PERSONAL HEALTH RECORD

Patient Information

Patient ID: PAT001

Patient Name	Date of Birth	Gender	Phone No.	Email ID	Disease	Patient History
Ruchi Aggarwal	9/1/1992	Female	9736553508	ruchia75@gmail.com	Measels	No past history

Snapshot 11: After Surgery or an operation, the doctor will update the patient record whether it was successful or not.

The screenshot shows a web interface titled "PERSONAL HEALTH RECORD". At the top, there is a "Patient ID" dropdown menu with "PAT001" selected. Below this, there are two radio buttons: "Success" (which is selected) and "Failure". At the bottom of the form, there are three buttons: "Update", "Back", and "Log Out". The background of the form is a light teal color.

Snapshot 12: Now if the user is a patient that is personal user then he/she will login using his/her ID and the encrypted key.

The screenshot shows a web interface titled "PERSONAL HEALTH RECORD" with a "Patient Login" section. The top of the page features a banner with four images: a doctor examining a patient, a pill organizer, a hand using a stethoscope, and a doctor holding a tablet. Below the banner, the "Patient Login" section has two input fields: "Patient ID" with "PAT001" entered, and "Key" with "5f3Qe2NpIwKirZ6Nj3yV6Q==" entered. At the bottom, there are two buttons: "Login" and "Back". The background of the form is a light teal color.

Snapshot 13: After the successful login, the patient's Home page will be opened where we have scheduling of an appointment, Doctor Prescription, surgery details and the logout option.



Snapshot 14: This is the page when the patient clicks the button for appointment details and ask for the appointment time by the doctor.



Snapshot 15: The patient can view his prescribed solution given to him by the doctor.




PERSONAL HEALTH RECORD

Doctor Prescription

Patient ID

patid	prescription
PAT001	Take proper bed rest.

Snapshot 16: Firstly PHR user is able to view data in encrypted form. But when he is verified using his secret key, he is able to view the data in decrypted data.



Patient ID

Enter Key

Attribute Based Encryption

Patient ID	Disease	Patient History	Patient ID	Solution	Prescription
PAT001	jttTO1/iSSs=6dDvhcrDHBIAOnA9AS8ZSA==		PAT001	P7gtthyJ0rlMnuHtVz6CA86ve8VICM19	Gwx29ehRPxJasRh5x31JYA==


Attribute Based Decryption

User has already viewed his details

patid	disease	pasthistory	prescription	solution
PAT001	Measels	No past history	Take dispirin	Take proper bed rest.

Break Down Glass Access

Snapshot 17: Doctor is able to view patient details in an emergency but in decrypted form.



Emergency Details


Emergency ID:

Patient ID:

Patient Name:

patid	disease	pathistory	prescription	solution
PAT001	jttTO1iSSs=6dDvchrDHBIAOnA9AS8ZSA==		Gwx29ehRPxJasRh5x3lJYA==	P7gthyJ0rlMnuHtVz6CA86ve8VICM9

Snapshot 18: Doctor wants to receive secret key to decrypt the data which is sent to the mail.




Retrieving Secret Key

Emergency ID:

Patient ID:

Mail ID:

Snapshot 19: After receiving the key on the mail id the doctor can then decrypt the data of the patient and view it in an emergency situation.



Retrieving Information

Patient ID

Key

patid	disease	pasthistory
PAT001	Measels	No past history

prescription	solution
Take dispirin	Take proper bed rest.

8. CONCLUSION AND FUTURE SCOPE

Conclusion

In the proposed scheme, it is possible to achieve secure sharing of personal health records and other files. Patients can have complete control of their own privacy through encrypting their Personal Health Record (PHR) and other data to allow access to selective users. The unique challenge introduced by multiple PHR owners and users such as security and key management complexities are greatly reduced by using DES encryption algorithm that has a key size of 56-bits. An Attribute Based Encryption (ABE) is used to encrypt the PHR data, so that patients can allow access not only to personal users, but also to various users from public domains with different professional roles. On-demand user revocation with security is also achieved. Through implementation and simulation, we show that our solution is scalable.

Future Scope

This project work is on simulation, while the extension of this work is that it can be deployed in real time applications for the secure sharing of files. More security can be achieved through other complicated algorithms with a larger key size.

The records can be shared with users other than the attributes included in this work.

9. REFERENCES

Books:

1. New Directions of Modern Cryptography, 2012 by Zhenfu Cao
2. Identity- Based cryptography by Marc Joye, Gregory Neven, IOS Press, 2009
3. Access Control, Authentication, and Public Key Infrastructure, Bill Ballard, Tricia Ballard, Erin Banks, Jones & Bartlett Publishers.
4. Introduction to Identity Based encryption by Luther Martin, Artech House, 2008
5. Cryptography with network security, William Stallings, Pearson Education
6. Secure Multi-Party Computation, ManojPrabhakaran, AmitSahai, IOS Press
7. A Practical Guide to Managing Information Security, Steve Purser, Artech house
8. Modern Cryptography: Theory and Practice, Mao, Pearson Education

Research Papers:

1. J. Benaloh, M. Chase, E. Horvitz, and K. Lauter, "Patient controlled encryption: ensuring privacy of electronic medical records," in CCSW '09, 2009, pp. 103–114.
2. V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in CCS '06, 2006, pp. 89–98.
3. A. Akinyele, C.U. Lehmann, M.D. Green, M.W. Pagano, Z.N.J. Peterson, and A.D. Rubin. Self-protecting electronic medical records using attribute-based encryption on mobile device. Technical report, Cryptology ePrint Archive, Report 2010/565, 2010. <http://eprint.iacr.org/2010/565>.
4. S. Narayan, M. Gagné, and R. Safavi-Naini, "Privacy preserving ehr system using attribute-based infrastructure," ser. CCSW '10, 2010, pp. 47–52.
5. L. Ibraimi, M. Petkovic, S. Nikova, P. Hartel, and W. Jonker, "Ciphertext-policy attribute-based threshold decryption with flexible delegation and revocation of user attributes," 2009.

6. S. D. C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, “Over-encryption: management of access control evolution on outsourced data,” in VLDB ’07, 2007, pp. 123–134.
7. A. Sahai and B. Waters. Fuzzy identity-based encryption. Advances in Cryptology {EUROCRYPT 2005, pages 457-473, 2005.
8. Angelo De Caro and Vincenzo Iovino, “jPBC: Java Pairing Based Cryptography” Computers and Communications (ISCC), 2011 IEEE Symposium on Digital Object Identifier: 10.1109/ISCC.2011.5983948 Publication Year: 2011 , Page(s): 850 – 855
9. M. Chase and S. S. Chow, “Improving privacy and security in multi-authority attribute-based encryption,” in CCS ’09, 2009, pp. 121–130.

Web:

1. www.google.com
2. www.wikipedia.org
3. www.w3schools.com
4. www.tutorialaspnet.com
5. www.stackoverflow.com
6. msdn.microsoft.com